Movie Recommendation System Using Machine Learning Techniques

Tejeshwar Reddy Induri
Master of Science in Computer Science
Dakota State University
Madison, USA
Tejeshwarreddy.induri@trojans.dsu.edu

Bhhaanu Pirange
Master of Science in Computer Science
Dakota State University
Madison, USA
Bhhaanu.pirange@trojans.dsu.edu

Harika Battu
Master of Science in Computer Science
Dakota State University
Madison, USA
Harika.battu@trojans.dsu.edu

Abstract— Movies are synonymous to most people currently as the go to entertainment service. Moreover, watching movies with relevant recommendations is a popular idea so, we as a team planned to produce a recommendation system using our knowledge of machine learning techniques. Although we have different types of recommender systems in the market, our system's goal is to provide clustering of these movies into different categories to help users select their desired ones. As part of this project, we will use TMDB 5000 movie dataset collected from Kaggle.

I. INTRODUCTION

Machine learning uses two techniques: Unsupervised learning makes use of internal structures or hidden patterns in the input data, whereas supervised learning uses known input and output data to train a model to predict future outputs. This project uses unsupervised learning to recommend movies based on similar interests and cluster them as top, average, and underrated movies with unlabeled data.

Presently, recommender systems support online social networks and e-commerce sites by providing users with helpful recommendations based on the utilization of networked databases. This is a recommender system that suggests a list of popular movies and moreover movies with similar interests based on cast, crew, genre etc., Our dataset contains information in different formats. As the first step of Analysis, we perform merging of two datasets (i.e., two csv files) that we opted to work on and start preprocessing to remove the noisy data from our dataset to get the best data for our project with Exploratory Data Analysis (EDA) techniques by investigating the data and outlining the key insights using graphs or python functions.

We have two sets of data such as credits and movies in which the data will be combined to one to maintain consistency. The initial features in our dataset will include Title, Cast, Crew, Genre, Overview and Popularity whereas the original dataset contains 4813 entries and 24 columns. Once we deal with preprocessing and applying demographic, content-based filtering algorithms for appropriate recommendation, we plan to group the movies by similarity matrix vector values and movie popularity scores w.r.t movie using k-means clustering algorithm for grouping the data. This algorithm falls under unsupervised learning and clusters unlabeled data. K-means clustering is an iterative process of assigning each data point to the groups and slowly data points get clustered based on features.

This project will use Google Colab as a computing platform through python programming. As part of python programming, we also use some of the libraries such as pandas, NumPy, plotly.express and sklearn for reading the CSV files, manipulating data, visualizing the patterns and modeling the algorithms.

II. METHODS

A recommendation system is a subclass of information filtering systems that aims to foresee the rating or similar interests of a user may assign to a certain item. By this we understood that a recommender system deals with filtering, and which is what we have as our methods to filter the data. However, by filtering we get our required information from the dataset but to show these filtered movies into categories we use a well know clustering technique called "k-means algorithm".

A. Demographic Filtering

Demographic filtering (DF) based recommender system categorize customers based on their demographic data and suggest services in line with that classification. The creation of user profiles in DF involves categorizing users according to stereotyped descriptions that describe the characteristics of different user classes. While demographic techniques use different data than collaborative methods, they still create "people-to-people" relationships. A user rating history, unlike the kind needed by the demographic approach, is required by a collaborative and content-based technique.

This filtering mechanism ultimately produces data for users to determine which items may be, appropriate for those users for recommendation, hence this helped us to achieve top 10 popular

movies overall based on vote average and vote count values from the dataset by setting the vote count threshold to 8 (which will be minimum) to be considered as an eligible movie. Using this we have calculated the weight rating formula as shown below,

$$W= (v/(v+m) * R) + (m/(m+v) * C)$$

Here,

W is the weighted rating.

R is the vote average given in the dataset.

v is the number of votes given for the movie.

m is the minimum number of votes required to be considered for recommendation.

C is the mean vote across the given vote average data.

```
def rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    return (v/(v+m) * R) + (m/(m+v) * C)
```

Fig. 1 Weighted Rating Formula

Once the eligible movies are filtered out, we use these to further process with another level of filtering called "Content-Based" to pick the similarities of every movie with one another.

B. Content-Based Filtering

Information filtering research has evolved into content-based filtering (CBF), which is a continuation of it. In a CBF system, the associated features define the items of interest. A content-based recommender builds a profile of the user's interests based on the qualities present in the objects that the user has evaluated.

In this context, based on the user's prior interests, content-based filtering makes recommendations for other movies that are similar to what they already enjoyed. For this purpose, we are using features like keywords, genre, cast, crew, and title columns from the data frame. The datatype of each column is in the format of list of dictionaries other than title columns from the data frame. The datatype of each column is in the format of list of dictionaries other than title column. We need to convert all other columns to a string format as we are trying to find similarities of words using cosine similarity.

The dataset that was used had to go through preprocessing such as dropping the columns that were not needed for processing and excluding the extra spaces between words that helps in cutting down the computational time and cost. We extracted a unique name from columns like 'cast', 'keywords', 'genres' as they add value to our data and replace these values with existing features. The outcome will be a list of string names now. Similarly, from 'crew' column we filtered out name of the person who has a role as 'director' because considering it has more impact. Once the list of string names is extracted, the resulted list must be converted into a sentence of strings, for which we extract each value from list and store them as a combination of strings using join(). A Count Vectorizer is used to transform a vector of token counts as machine can't understand text data which can convert the text data to numerical representation.

Further, to measure the similarity between the vectors produced by the count vectorizer we use cosine similarity. Cosine similarity is a measure of similarities in each product space. This is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. Once the similarity score are obtained a data frame is used to represent these cosine similarity values in count matrix. Finally, a user-friendly function is created which takes title of a specific movie and cosine similarity values as parameters to recommend top movies similar to the parameterized movie for any user. "Fig. 2" below illustrates the recommendation function used.

```
def recommend(title, cosine_sim):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    movie_indices = [i[0] for i in sim_scores]
    return df2['title'].iloc[movie_indices]
```

Fig. 2 Recommend Function

After Demographic and Content based filtering as our first steps of process to filter out the best/top movies from the given data based on scoring, we plan to group the movies by similarity matrix vector values and movie popularity scores w.r.t movie using k-means clustering algorithm for grouping the data.

C. K-means Clustering Algorithm

As a general method for locating groups or clusters in multivariate data, clustering is the most extensively used method. The fact that the ideal number of clusters for various dataset types is rarely known in practice, which is one of the key challenges for cluster analysis. To explore the inherited grouping or partition of data objects according to a known number of clusters, however, is the main purpose of most clustering algorithms. Thus, even though it must overcome numerous operational difficulties, determining the number of clusters is a crucial task for each clustering problem in practice.

For clustering, several algorithms have been put out in the literature. Due to its simplicity, the k-means clustering algorithm is the most used. One of the most straightforward unsupervised learning algorithms to handle the well-known clustering problem is K-Means. The process uses a predetermined number of clusters (let's assume k clusters).

A) Process:

The primary concept is to define k centroids, one for each cluster. Because different locations produce varied results, these centroids should be positioned

deftly. Therefore, it is preferable to situate them as far apart from one another as you can. The next step is to take each point in each data collection and connect it to the closest centroid. The first group is finished once there are no open points. At this stage, k new centroids that serve as the centers of the clusters formed in the previous phase must be recalculated. The identical data points must be bound once more to the subsequent k new centroids that are closest to them.

`

B) Algorithm:

1. Add k points to the area that the objects in the cluster represent. These positions are the centroids of the first group.
2. Assign each item to the group whose centroid is closest to it.
3. Recalculate the locations of the k centroids once all the items have been assigned.
4. Continue Steps 2 and 3 until the centroids stop moving.

Our main goal of using K-means is to cluster the given recommended movies list into top rated, average, and underrated clusters.

After several runs of K-Means, there have been numerous recommendations in the literature for selecting the appropriate K value. However, we have picked the Elbow method which is visual. Starting with K=2, the goal is to increase it by 1 in each step as you calculate your clusters and the cost of the training. The cost starts to decline significantly at a certain number for K, and as you raise K more, it then approaches a plateau. At this point you have your K value. The same was applied to our project and found out the K value to be three. “Fig. 3” illustrates the elbow method and the bend at point 3 that doesn’t change drastically going forward.

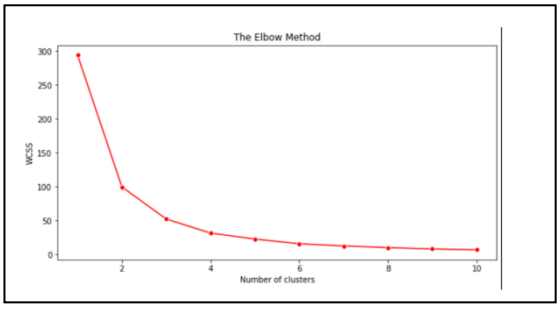

Fig. 3 Elbow Method Graph

The K-means clustering algorithm is used to find groups which have not been explicitly labeled in the data. We are using the recommended movies similarity scores and rating scores as features, where this data depends on the movie name, that is used to get recommendations. Hence the data changes for every new movie recommendation.

Once the movies are clustered into three categories, we show the movies of a particular cluster and give a count of the number of movies for each cluster.

III. Results

A. *Demographic Filtering*

Firstly, we derived the eligible movies to be recommended by setting a criteria as minimum percentage of votes to be at 80. This gave us a count of movies that can be considered for recommendation. “Fig. 3” shows the same on the eligible movies.

```
eligible_movies=eligible_movies.sort_values(by=['score'],ascending=False)
eligible_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
```

	title	vote_count	vote_average	score
1887	The Shawshank Redemption	8205	8.5	8.248201
662	Fight Club	9413	8.3	8.096011
3342	The Godfather	5893	8.4	8.077220
3237	Pulp Fiction	8428	8.3	8.074604
65	The Dark Knight	12002	8.2	8.044155
809	Forrest Gump	7927	8.2	7.972682
96	Inception	13752	8.1	7.969210
95	Interstellar	10867	8.1	7.937302
1996	The Empire Strikes Back	5879	8.2	7.904593
1824	Schindler's List	4329	8.3	7.899869

Fig. 4 Eligible Movies

The same eligible movies are represented in pictorial form using bar graph. “Fig. 4” shows top ten movies that are eligible for recommendation.

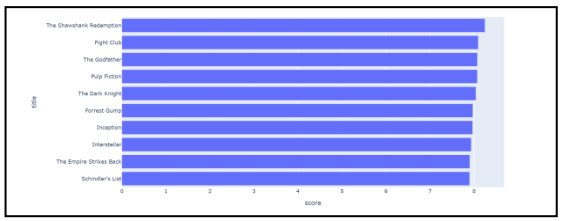

Fig. 5 Top Ten Eligible Movies

B. *Content-Based Filtering*

This filtering technique is processed with data containing features like keywords, genres, cast, crew, and title and these are prepossessed to further generate similarity scores and finally recommends based on a given movie/title name using the recommend function. “Fig. 5” shows the results of recommendation for Newlyweds movie from the given data set.

We only consider the features of keywords, genres, cast, crew, title from the actual data set to improve our computation time over unnecessary processing.

```
recommend('Newlyweds', cosine_sim).head(900)

4755                    The Brothers McMullen
3844                            She's the One
3812                           Purple Violets
4254                  Me You and Five Bucks
1261              Life or Something Like It
                          ...
1613                             Midnight Run
1622    What's the Worst That Could Happen?
1739                       The Spy Next Door
1938                                  Sheena
1946                                 Carnage
Name: title, Length: 900, dtype: object
```

Fig. 6 Recommendations For a movie

C. *K-means Clustering*

K-means uses the new recommendations data with similarity scores and groups them into top, average, and underrated movies. "Fig. 7" shows the clusters formation where the blue color indicates top rated, green indicates average and yellow indicting underrated movies.

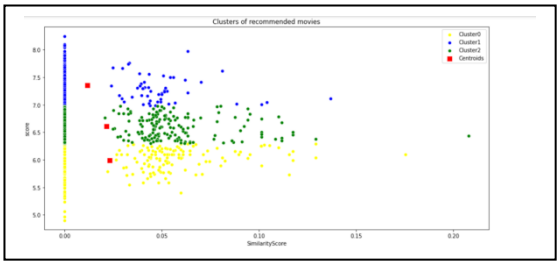

Fig. 7 Clusters

As mentioned earlier, while clusters are not showing the movie/title name "Fig. 8" shows the derived movie names with its cluster groups.

	sno	Index	title	SimScore	score	Cluster
0	0	395	The Holiday	0.207950	6.433345	2
1	1	1566	27 Dresses	0.175412	6.096295	0
2	2	1701	Aladdin	0.136931	7.113539	1
3	3	815	Hitch	0.129099	6.287667	0
4	4	817	American Wedding	0.129099	6.042477	0
...	...	...	...	...	...	...
969	969	4502	The Blair Witch Project	0.000000	6.201234	0
970	970	4583	Paranormal Activity	0.000000	5.981123	0
971	971	4608	12 Angry Men	0.000000	7.534800	1
972	972	4645	Cube	0.000000	6.518282	2
973	973	4676	Mad Max	0.000000	6.376009	2

Fig. 8 Movies and Their Cluster Group

All these results together prove our aim to achieve the top, average, and underrated movies from the dataset of 5000 movies.

IV. Conclusion

Companies that deploy recommender systems may gain tactical advantages, and they may also boost consumer loyalty and satisfaction. If businesses do not apply the recommender system, they run a greater risk of being driven out of the market. When first adopting the recommender system, it's critical to increase accuracy because poor user recommendations might lessen the impact that the recommender system has on sales. All these results together prove our aim to achieve the top, average, and underrated movies from the dataset of 5000 movies. Considering this in our work, we cover two alternative recommender system types, but it is impossible to say that one recommender system type is superior to another because basic systems can also be cheaper to develop even though they have slower accuracy.

References

[1] https://www.ijert.org/research/recommender-systems-types-of-filtering-techniques-IJERTV3IS110197.pdf

[2] https://www.researchgate.net/profile/Trupti-Kodinariya/publication/313554124_Review_on_Determining_of_Cluster_in_K-means_Clustering/links/5789fda408ae59aa667931d2/Review-on-Determining-of-Cluster-in-K-means-Clustering.pdf

[3] https://towardsdatascience.com/visualising-similarity-clusters-with-interactive-graphs-20a4b2a18534

[4] https://cemsarier.github.io/algorithm/recommender%20system/analytics/movie_recommendation/

[5] https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata?select=tmdb_5000_credits.csv

[6] https://asdkazmi.medium.com/ai-movies-recommendation-system-with-clustering-based-k-means-algorithm-f04467e02fcd

[7] https://realpython.com/k-means-clustering-python/

[8] https://towardsai.net/p/l/all-about-k-means-clustering

[9] https://linguisticmaz.medium.com/k-means-clustering-in-python-9825a280f9cb