**Movie Recommendation**

**DSC 680 – Project1 Milestone 2**

Mehtre Vijaykumar

06/19/2024

## Introduction

This project focus on recommendation system for video content providers to predict whether someone will enjoy a movie based on how much they liked or disliked other movies. Recommendation systems are a type of **information filtering system** designed to enhance search results by providing items that are more relevant to the search query or related to the user's search history.

We will do so by following a sequence of steps needed to implement recommendation system. We will start with preprocessing and cleaning of the raw text of the movies, author and ratings. Then we will explore the cleaned text and try to get some intuition about the context of the movies and ratings. After that, we will extract numerical features from the data and finally use these feature sets to train models and identify the movie which someone will enjoy.

## Business Problem

Major companies like YouTube, Amazon, Netflix use recommendation systems in social and e-commerce sites use recommendation system for its users to suggest for an individual according to their requirement more precise and accurate item.

These online content and service providers have a huge amount of content so the problem which arises is which data is required for whom so the problem of providing apposite content frequently. This project represents the overview and approaches of techniques generated in a recommendation system.

### There are basically three types of recommender systems: -

* **Demographic Filtering**- offers users with similar demographic background the similar movies that are popular and well-rated regardless of the genre or any other factors. Therefore, since it does not consider the individual taste of each person, it provides a simple result but easy to be implemented. The System recommends the same movies to users with similar demographic features. Since each user is different, this approach is too simple. The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience.
* **Content Based Filtering**- consider the object’s contents, this system uses item metadata, such as genre, director, description, actors, etc. for movies, to make these recommendations, it will give users the movie recommendation more closely to the individual’s preference. They suggest similar items based on a particular item. The general idea behind these recommender systems is that if a person liked a particular item, he or she will also like an item that is like it.
* **Collaborative Filtering**- focuses on user’s preference data and recommend movies based on it through matching with other users’ historical movies that have a similar preference as well and does not require movies’ metadata. This system matches persons with similar interests and provides recommendations based on this matching. Collaborative filters do not require item metadata like its content-based counterparts.

Content based systems works based on the label or genre of an item. If a user watched a movie so it recommends similar movies based on director, a genre, and many more aspects. The theory behind the collaborative filtering is that if user’s ‘A’ and ‘B’ have rated correspondingly in the past, then there will be an assumption that they will rate correspondingly in the future.

Diagram

Description automatically generated

Figure 1 – Recommendation System

## Data Explanation

I have used movie-lens dataset from: https://grouplens.org/datasets/movielens/latest/.

**Data description**

The dataset contains 100k+ ratings and 3k+ tag applications across 9k+ movies. The data was captured for 600+ users between 1996 and 2018. This dataset was generated in September 2018.

We are going to use 3 dataset , movie , credit and rating to implement movie recommendation system .

The **credits** dataset contains the following features: -

* movie\_id - A unique identifier for each movie.
* cast - The name of lead and supporting actors.
* crew - The name of Director, Editor, Composer, Writer etc.

A screenshot of a computer

Description automatically generated

Figure 2 – Credit Data snapshot

The **Movie** dataset has the following features: -

* budget - The budget in which the movie was made.
* genre - The genre of the movie, Action, Comedy ,Thriller etc.
* homepage - A link to the homepage of the movie.
* id - This is infact the movie\_id as in the first dataset.
* keywords - The keywords or tags related to the movie.
* original\_language - The language in which the movie was made.
* original\_title - The title of the movie before translation or adaptation.
* overview - A brief description of the movie.
* popularity - A numeric quantity specifying the movie popularity.
* production\_companies - The production house of the movie.
* production\_countries - The country in which it was produced.
* release\_date - The date on which it was released.
* revenue - The worldwide revenue generated by the movie.
* runtime - The running time of the movie in minutes.
* status - "Released" or "Rumored".
* tagline - Movie's tagline.
* title - Title of the movie.
* vote\_average - average ratings the movie recieved.
* vote\_count - the count of votes receive

A screen shot of a computer

Description automatically generated

Figure 3 – Movie Data snapshot

The **Rating** dataset has the following features: -

* userId - A unique identifier for user who submitted the rating
* movieId - A unique identifier for each movie.
* rating - rating for movie from 0 to 5
* timestamp – rating timestamp

A table with numbers and text

Description automatically generated

Figure 4 – Rating Data snapshot

## Data Analysis

**Demographic Filtering**

Before getting started with this -

* we need a metric to score or rate movie.
* Calculate the score for every movie.
* Sort the scores and recommend the best rated movie to the users.

We can use the average ratings of the movie as the score but using this won't be fair enough since a movie with 9 average rating and only 10 votes cannot be considered better than the movie with 8 as as average rating but 60 votes. So, we will be using IMDB's weighted rating (wr) which is given as :-

A math equation with numbers and symbols

Description automatically generated

were,

* v is the number of votes for the movie.
* m is the minimum votes required to be listed in the chart.
* R is the average rating of the movie; And
* C is the mean vote across the whole report

We already have v(**vote count**) and R (**vote average**) and C can be calculated as

**Weighted Rating**

We need to calculate our metric for each qualified movie. To do this, we will define a function, **weighted\_rating()** and define a new feature **score**, of which we'll calculate the value by applying this function to our DataFrame of qualified movies:

*def weighted\_rating(x, m=m, C=C):*

*v = x['vote\_count']*

*R = x['vote\_average']*

*# Calculation based on the IMDB formula*

*return (v/(v+m) \* R) + (m/(m+v) \* C)*

**Content Based Filtering**

A diagram of a movie

Description automatically generatedWe will compute pairwise similarity scores for all movies based on their plot descriptions and recommend movies based on that similarity score.

We will compute Term Frequency-Inverse Document Frequency (TF-IDF) vectors for each overview. Term Frequency: **(term instances/total instances)**. Inverse Document Frequency: **log(number of documents/documents with term)** The overall importance of each word to the documents in which they appear is equal to **TF \* IDF**

This will give us a matrix where each column represents a word in the overview vocabulary and each row represents a movie. This is done to reduce the importance of words that occur frequently in plot overviews and therefore, their significance in computing the final similarity score.

We will use the library scikit-learn which gives us a built-in TfIdfVectorizer class that produces the TF-IDF matrix.

These are the following steps we'll follow to define our recommender system:-

* Get the index of the movie given its title.
* Get the list of cosine similarity scores for that movie with all movies. Convert it into a list of tuples where the first element is its position and the second is the similarity score.
* Sort the mentioned list of tuples based on the similarity scores; that is, the second element.
* Get the top 10 elements of this list. Ignore the first element as it refers to self (the movie most similar to a particular movie is the movie itself).
* Return the titles corresponding to the indices of the top elements.

A screenshot of a computer

Description automatically generated

## Conclusion

As we see, the quality of recommendations is not that great. "The Dark Knight Rises" returns all Batman movies while it is more likely that the people who liked that movie are more inclined to enjoy other Christopher Nolan movies. This is something that cannot be captured by the present system.

## Challenges

**New User**: A newly released movie cannot be recommended to the user until it gets some ratings. A new user or item added based problem is difficult to handle as it is impossible to obtain a similar user without knowing previous interest or preferences.

**Synonymy**: arises when a single item is represented with two or more different names or listings of items having similar meanings, in such condition, the recommendation system can’t recognize whether the terms show various items or the same item.

## Questions

1. **New User:** A newly released movie cannot be recommended to the user until it gets some ratings. A new user or item added based problem is difficult to handle as it is impossible to obtain a similar user without knowing previous interest or preferences. How to handle this scenario?
2. **Synonymy** arises when a single item is represented with two or more different names or listings of items having similar meanings, in such condition, the recommendation system can’t recognize whether the terms show various items or the same item. How can we address this issue?
3. Scalability of the model ?
4. Drawbacks and limitations of Collaborative Filtering?
5. Drawbacks and limitations of Content Based Filtering?
6. Drawbacks and limitations of Demographic Filtering?
7. Which is the best Algorithm for Recommendation?
8. What are the factors that are taken into consideration while recommending a movie to the user, for eg. Age, demography, ethnicity, interests, language etc.
9. How much data is enough to predict recommendations for a user or does these changes with the amount of data we process?
10. How soon this recommender system needs to be re-trained?

## Implementation Plan

As we have tried the content-based filter and it’s not so much effective, so we are planning we implement “**Collaborative Filtering**”.

* Find the K-nearest neighbors (KNN) to the user a, using a similarity function w to measure the distance between each pair of users:
* Predict the rating that user a will give to all items the k neighbors have consumed but a has not. We Look for the item j with the best predicted rating. In other words, we are creating a User-Item Matrix, predicting the ratings on items the active user has not seen, based on the other similar users. This technique is memory-based.

## Ethical Consideration

Users for all the reviews in the dataset were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided.

## Reference

[1]Herlocker, J, Konstan, J., Terveen, L., and Riedl, J. Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems 22 (2004), ACM Press, 5-53.

[2] Koren, Yehuda. “Factorization meets the neighborhood: a multifaceted collaborative filtering model.” In Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 426–434. ACM, 2008.

[3] <https://www.mygreatlearning.com/blog/masterclass-on-movie-recommendation-system/>

[4] <https://docs.microsoft.com/en-us/dotnet/machine-learning/tutorials/movie-recommendation>

[5] MovieLens 2018 Introduction-to-Machine-Learning

https://github.com/codeheroku/Introduction-toMachineLearning/tree/master/CollaborativeFiltering/dataset

## Appendix

|  |
| --- |
| Figure 1 – Recommendation System – Page 4 |
| Figure 2 – Credit Data snapshot – Page 5 |
| Figure 3 – Movie Data snapshot – Page 6 |
| Figure 4 – Rating Data snapshot – Page 7 |