Movie Recommendation System

Lalitha Sumedha Pothineni
Computer and Information Sciences
University of Michigan-Dearborn
Dearborn MI US
sumedhap@umich.edu

Jaya Shree Jilkara
Computer and Information Sciences
University of Michigan-Dearborn
Dearborn MI US
jshree@umich.edu

Aishwarya Dongari
Computer and Information Sciences
University of Michigan-Dearborn
Dearborn MI US
aishredy@umich.edu

Abstract

We live in a world where technology changes every day, and due to this advancement, we are surrounded by massive data that is unstructured. Data analytics is changing the movie trend; for example, Recommendation systems are one of the fascinating applications of machine learning. Sites like YouTube and Netflix use the recommendation systems extensively to recommend those videos and movies that one may want to watch. For this project, we have implemented a similar recommendation system for movies.

We have built a Simple recommendation system using the dataset based on the popularity of the movie. However, this system gives the same recommendations to everyone. And so, we have implemented content-based filtering. We have done this by taking specific metrics such as the movies that are liked by that user into consideration. This content-based filtering also suffers from several limitations. And so, we have explored them and implemented a Collaborative Filtering to personalize the recommendations. We have also implemented a Hybrid Recommendation system which is a collaboration of both content and collaborative filtering.

Introduction

Recommendation system helps users discover content based on their unique needs and preferences. So, it's all about customization and making it tailored to a person's needs and preferences.

A recommendation system learns from a customer and recommends products or data that he/she will find most valuable or might be interested in. It helps in predicting which movie a customer would like, depending on the ratings from him/her and other similar users, for example: If user1 watches same two movies that user 2 has watched, user2 will be recommended the next movie that user1 watches and vice versa.

This project focuses on predicting and recommend movies to users. It is because of these kinds of recommendation systems that people watch movies that they themselves don't know that they like them until an AI, or a machine learning interface uses an algorithm and provides them with that recommendation.

Background/Motivation

Dataset

The dataset we are using contains metadata of nearly 45,000 movies which includes cast, crew, languages, countries, TMBD votes and its averages. This particular dataset also contains 26 million ratings on a scale of 1-5 for all the movies by nearly 270,000 users.

Libraries

- Anaconda is a distribution of python. Additionally, we are using libraries and tools such as NumPy, SciPy, Pandas, Matplotlib etc.

- Jupyter is a development environment where we are writing the code, exploring and analyzing the data

```
%matplotlib inline
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.corpus import stopwords
from scipy import stats
from wordcloud import WordCloud,STOPWORDS
from ast import literal_eval
from sklearn.feature_extraction.text import TfidfVectorizer, CountVector
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate
import warnings; warnings.simplefilter('ignore')
```

Related Work

We have taken the paper '*Towards the next generation of recommender systems: a survey of the state-of-the-art and possible extensions*' [1] as a reference for this project. We have learnt about the recommendation systems that are currently in use from this paper. In addition to this, it also gave us an insight on how to improve these recommendation systems in near future.

We also got to know about the most widely used recommendation system from the paper by *F. Hdioud, B. Frikh and B. Ouhbi [2]*. In this paper, the author has discussed all the algorithms used in developing recommendation systems besides its shortcomings. Furthermore, the main idea to do this project came from the lectures that were discussed in class on the Topic: *Recommendation Systems [3]*. We got intrigued and have decided to dig deeper into the concept.

Methodology

Simple Recommendation System

It is a recommendation system that recommends to every user based on how popular the movie is. The idea is that it is usually assumed that the more popular movies will be liked by the majority of the people. We have implemented this by sorting our dataset based on ratings as well as popularity. Then we have displayed the movies that topped our list. We have also used TMBD ratings and IMBDs rating formula for constructing the chart.

$$\mathrm{WR} = \frac{v}{v+m} \cdot R + \frac{m}{v+m} \cdot C$$

R= Average
v =total no of votes
m = min no of votes (95% in our case)
C = mean of votes

Evaluation (Simple Recommendation System)

We have evaluated our system by displaying the movies based on a particular genre. In the output below, one can find all the movies having Thriller or Action as their genre. Also, most of the movies have the director in common. Example: 'Inception', 'The Dark Knight', 'Interstellar' has been directed by Christopher Nolan

Out[22]:

	title	year	vote_count	vote_average	popularity	genres	wr
15480	Inception	2010	14075	8	29.1081	[Action, Thriller, Science Fiction, Mystery, A...	7.917588
12481	The Dark Knight	2008	12269	8	123.167	[Drama, Action, Crime, Thriller]	7.905871
22879	Interstellar	2014	11187	8	32.2135	[Adventure, Drama, Science Fiction]	7.897107
2843	Fight Club	1999	9678	8	63.8696	[Drama]	7.881753
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.0707	[Adventure, Fantasy, Action]	7.871787
292	Pulp Fiction	1994	8670	8	140.95	[Thriller, Crime]	7.868660
314	The Shawshank Redemption	1994	8358	8	51.6454	[Drama, Crime]	7.864000
7000	The Lord of the Rings: The Return of the King	2003	8226	8	29.3244	[Adventure, Fantasy, Action]	7.861927
351	Forrest Gump	1994	8147	8	48.3072	[Comedy, Drama,	7.860656

Content-Based Recommendation System

We have realized that the simple recommendation system recommends the same irrespective of the users, and it may not be liked by many. To overcome this, we have implemented the content-based recommendation system that computes the similarity between movies and recommends the films that are similar to the movies that the user has liked before. To achieve this, we have used the movie descriptions and taglines.

To calculate the similarity between the two movies, we have used cosine similarity.

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

Evaluation (Content-Based)

We have evaluated this system based on the type of movie. For example: In the output below you can see all the movies related to the Batman saga on trying to get recommendations for the movie 'Batman Forever'

```
In [46]: getRecommendation('Batman Forever').head(20)

Out[46]: 7931                        The Dark Knight Rises
         2579                  Batman: Mask of the Phantasm
         6900                              The Dark Knight
         6144                                Batman Begins
         8165     Batman: The Dark Knight Returns, Part 1
         524                                        Batman
         1240                              Batman & Robin
         1113                               Batman Returns
         7565                    Batman: Under the Red Hood
         7901                             Batman: Year One
         8227     Batman: The Dark Knight Returns, Part 2
         681                            Eyes Without a Face
         6206                                     Cry_Wolf
         1135                       Night Falls on Manhattan
         2075                               Open Your Eyes
         149                                       Hackers
         8917           Batman v Superman: Dawn of Justice
         2696                                          JFK
         8680                            The Young Savages
         7242                      The File on Thelma Jordon
         Name: title, dtype: object
```

Collaborative Filtering

All the content-based recommendation system does is capturing the movies that are similar to the films the user has liked before. However, it does not take the preference of the user into consideration. In Collaborative Filtering, it doesn't really care what the movie is. It will recommend movies based on how similar other users have predicted it. For this purpose, we have implemented the collaborative filtering in which the tastes of similar users are taken into account.

We have used an algorithm called SVD (Singular Value Decomposition). SVD is a built-in recommendation system that is known to produce very accurate results. It was used widely by Netflix.

Evaluation (Collaborative Filtering)

We have evaluated this using the Root Mean Square Error. For this, the RMSE obtained was 0.89, which is less than 1.

```
Out[80]: {'test_rmse': array([0.89707477, 0.88324853, 0.90921328, 0.89043196, 0
.90354828]),
 'test_mae': array([0.69212425, 0.68049989, 0.69980372, 0.68921543, 0.
69242386]),
 'fit_time': (5.8468077182769775,
  6.20166802406311,
  5.724970817565918,
  5.793152093887329,
  5.855086088180542),
```

Hybrid Recommendation System

We have combined the content and collaborative filtering which recommends movies based on the ratings calculated for that user.

Evaluation (Hybrid System)

In the output below, you can see that although the movie is the same, the movies recommended to the users are different.

```
In [92]: hybridRecommendation(1, 'The Terminator')

Out[92]:
```

	title	vote_count	vote_average	year	id	est
7488	Avatar	12114.0	7.2	2009	19995	3.018819
974	Aliens	3282.0	7.7	1986	679	2.981203
522	Terminator 2: Judgment Day	4274.0	7.7	1991	280	2.954099
6967	Doomsday	374.0	5.8	2008	13460	2.927908
6394	District B13	572.0	6.5	2004	10045	2.819424
1376	Titanic	7770.0	7.5	1997	597	2.747625
7403	Gamer	778.0	5.6	2009	18501	2.706491
7502	The Book of Eli	2207.0	6.6	2010	20504	2.694188
5296	Zardoz	106.0	5.8	1974	4923	2.683079
7991	In Time	3512.0	6.7	2011	49530	2.658243

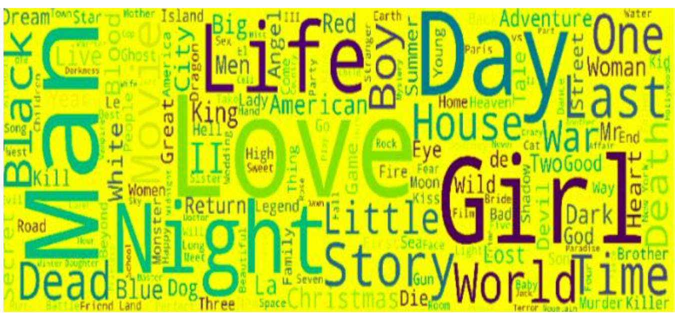
```
In [93]: hybridRecommendation(100, 'The Terminator')

Out[93]:
```

	title	vote_count	vote_average	year	id	est
974	Aliens	3282.0	7.7	1986	679	3.899735
7991	In Time	3512.0	6.7	2011	49530	3.727733
7502	The Book of Eli	2207.0	6.6	2010	20504	3.707291
522	Terminator 2: Judgment Day	4274.0	7.7	1991	280	3.657294
922	The Abyss	822.0	7.1	1989	2756	3.618038
6622	Children of Men	2120.0	7.4	2006	9693	3.599097
6394	District B13	572.0	6.5	2004	10045	3.511455
7296	Terminator Salvation	2496.0	5.9	2009	534	3.448091
344	True Lies	1138.0	6.8	1994	36955	3.436736
2412	RoboCop	1494.0	7.1	1987	5548	3.411205

Experimental Discussion

We have performed exploratory data analysis on the dataset and played with the data by exploring which word is repeated frequently. We have done this by using the term frequency and inverse document frequency.

The easiest way to analyze data is by representing it in the form of a cloud. The following figure depicts a word cloud that describes the titles for a movie.



In order to know the relationship between multiple variables, we have used seaborn, a library built on top of matplotlib and is integrated with pandas.

Individual Contribution

- Jayashree Jilkara – Performed Exploratory Data analysis, Implemented Collaborative Filtering System, and Hybrid Recommendation System, written the report
- Lalitha Sumedha Pothineni – Implemented Content-Based Recommendation and contributed to the report
- Aishwarya Dongari – Implemented Simple Recommendation System as well as contributed to the report

Conclusion

We have successfully implemented all the recommendation systems we were taught in class based on different ideas and algorithms.

- Simple Recommender system: based on the popularity of the movie.
- content-based filtering: taking specific metrics such as the movies that are liked by that user
- Collaborative Filtering: to personalize the recommendations using SVD
- Hybrid: Combined content and collaborative

References

1. Gediminas Adomavicius, Alexander Tuzhilin - "*Towards the next generation of recommender systems: a survey of the state-of-the-art and possible extensions*," IEEE transactions on knowledge and data engineering, 2005
2. F. Hdioud, B. Frikh and B. Ouhbi, "*A comparison study of some algorithms in Recommender Systems*," 2012 Colloquium in Information Science and Technology, Fez, 2012, pp. 130-135.doi: 10.1109/CIST.2012.6388076
3. Abouelenien, Mohamed- CIS 5570 Introduction to Big Data Lecture (Chapter 9: *Recommendation Systems*), 2020