Project Report

Spring 2022

Introduction to Data Mining

Professor Huang

Group Members
Robert Doxey
Aloysius Arno Wiputra
Jordan Fuchs
Dylan McKenna
Brian Suriel

Problem Introduction

Deciding which movie to watch can sometimes be a tough decision. You may waste hours just searching for the right movie, leaving your night ruined. In some cases, you may even find yourself watching the beginning of a movie only to realize 30 minutes into it that you made the wrong decision. It would be nice to have an accurate and reliable movie recommender system that movie watchers can use when in search for their next movie.

Our project aims to solve the aforementioned problem in two ways. One solution is to accurately recommend movies based on someone's past watched movies. The other solution is to recommend movies based on similar titles & descriptions to a movie they have watched. Both ways use different algorithms to solve the problem. These algorithms will be explained in detail further into this report.

Formalization

The data set we selected contains data for over 9,000 movies. The data it contains for each movie are organized under the following attributes: 'Release_Date,' 'Title,' 'Overview,' 'Popularity', 'Vote_Count', 'Vote_Average', 'Original_Language,' 'Genre,' and 'Poster_URL.' Initially, all data in this dataset is sorted based on popularity. It is also worth noting that the dataset is in .csv format, which is a very common and workable file format.

This dataset is the brain of our project, as it is what contains all of the information needed in order to actually recommend movies. Most importantly, with this data and our algorithms put together, we can provide the end-user with a functional movie recommender system. As you can see, our project's function is clearly a recommendation system due to the fact that our program will recommend movies to users based on certain criteria related to the user.

Algorithms

Algorithm 1

The first algorithm we implemented is a solution based on computing Jaccard similarity. For this part, we had written a script to separate and convert the genre values in the dataset from one mixed entry into separate columns and another script to fill those columns with 1s and 0s based on whether the genre is valid for that movie or not. The dataframe containing the encoded genres were then transposed and saved to another csv so we can reference that whenever we needed after. Since we had done a sorting and reset the index, we were confident that this new dataset matched the original one, but we also did checks to ensure that that was the case.

The recommender algorithm then takes in a user input or data and checks whether the movie we had queried is in the dataframe or not. If it is, then we iterated through the dataframe and compared the genres of the movie we had searched with the genres of all the 9000+ movies in the dataset. We then kept the ones that are similar up to a certain threshold based on the argument we had passed, and also kept the max rating. If a movie had at that point the same similarity rating, we compared the popularity and chose the more popular one to be recommended.

The output is then saved into a new result dataframe, and outputted to the sure based on max popularity.

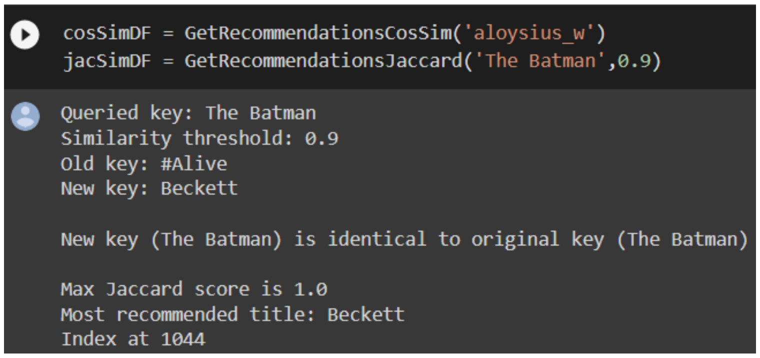
Algorithm 2

The second algorithm uses TF-IDF and Cosine Similarity instead of jaccard similarity. For this approach, we tested it with a sample user dataframe. First we combined the title of the movie and the plot summary of the movie into a new column, and used a python package to vectorize it. It was then fit into a TF-IDF matrix and applied the cosine similarity function to the matrix.

We then created a function to take in the user_id and grabbed the movies that the user have watched from the sample dataframe. The result is then passed into the method and we calculated the cosine similarity score of that movie and compared it with the movies in the dataset. For each entry that we found to have a high similarity score, we then outputted into a result dataframe showing what movies we recommend based on what movie we had queried.

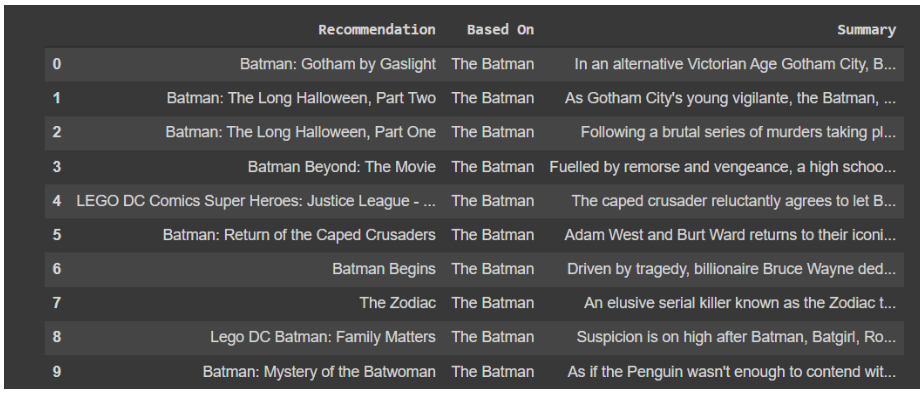
Experiment Results

To show how the two algorithms do head-to-head, they were both queried using the same movie, 2022's *The Batman* to see what they would recommend. The only difference here is that the Cosine Similarity was queried via the user file, whereas Jaccard was queried with the name of the movie with a similarity threshold of 0.9.



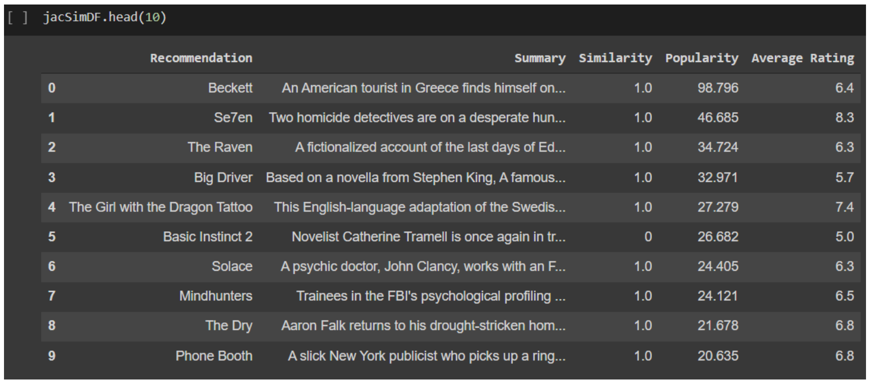
```
cosSimDF = GetRecommendationsCosSim('aloysius_w')
jacSimDF = GetRecommendationsJaccard('The Batman',0.9)
```

```
Queried key: The Batman
Similarity threshold: 0.9
Old key: #Alive
New key: Beckett

New key (The Batman) is identical to original key (The Batman)

Max Jaccard score is 1.0
Most recommended title: Beckett
Index at 1044
```



	Recommendation	Based On	Summary
0	Batman: Gotham by Gaslight	The Batman	In an alternative Victorian Age Gotham City, B...
1	Batman: The Long Halloween, Part Two	The Batman	As Gotham City's young vigilante, the Batman, ...
2	Batman: The Long Halloween, Part One	The Batman	Following a brutal series of murders taking pl...
3	Batman Beyond: The Movie	The Batman	Fuelled by remorse and vengeance, a high schoo...
4	LEGO DC Comics Super Heroes: Justice League - ...	The Batman	The caped crusader reluctantly agrees to let B...
5	Batman: Return of the Caped Crusaders	The Batman	Adam West and Burt Ward returns to their iconi...
6	Batman Begins	The Batman	Driven by tragedy, billionaire Bruce Wayne ded...
7	The Zodiac	The Batman	An elusive serial killer known as the Zodiac t...
8	Lego DC Batman: Family Matters	The Batman	Suspicion is on high after Batman, Batgirl, Ro...
9	Batman: Mystery of the Batwoman	The Batman	As if the Penguin wasn't enough to contend wit...

Above are the results of the cosine similarity algorithm, which recommends movies with a similar title, all but one of which features the word Batman. This makes sense because the algorithm uses TF-IDF and that would focus on a keyword like Batman. Not only that, but the user who was taken as input for the query, in this case, 'aloysius,' has watched a lot of Batman movies. It would be a fairly reasonable assumption that someone who has seen a lot of Batman movies would want to watch more of them.



```
[ ] jacSimDF.head(10)
```

	Recommendation	Summary	Similarity	Popularity	Average Rating
0	Beckett	An American tourist in Greece finds himself on...	1.0	98.796	6.4
1	Se7en	Two homicide detectives are on a desperate hun...	1.0	46.685	8.3
2	The Raven	A fictionalized account of the last days of Ed...	1.0	34.724	6.3
3	Big Driver	Based on a novella from Stephen King, A famous...	1.0	32.971	5.7
4	The Girl with the Dragon Tattoo	This English-language adaptation of the Swedis...	1.0	27.279	7.4
5	Basic Instinct 2	Novelist Catherine Tramell is once again in tr...	0	26.682	5.0
6	Solace	A psychic doctor, John Clancy, works with an F...	1.0	24.405	6.3
7	Mindhunters	Trainees in the FBI's psychological profiling ...	1.0	24.121	6.5
8	The Dry	Aaron Falk returns to his drought-stricken hom...	1.0	21.678	6.8
9	Phone Booth	A slick New York publicist who picks up a ring...	1.0	20.635	6.8

In the screenshot above are the results of the Jaccard algorithm. Instead of focusing on the words in the title or summary of the movie and looking for matching phrases, it looks for movies that have similar genres. It looks more for gritty, action-y films with lots of darker themes instead of straight matches. While it may be a safe bet to assume a Batman fan would want to watch another Batman movie, that may not be the *only* thing they ever want to watch, sometimes they may want to mix it up. One could eat ice cream for every meal, but that doesn't make it a good idea.

Task Table

Tasks	People
1. Group Leader - assigned tasks and made sure everything got done on time	Robert
2. Proposal	Robert, Dylan, Brian, Arno, Jordan
3. Collecting and Preprocessing Data	Arno, Robert
4. Implementing Algorithm 1	Arno, Robert
5. Implementing Algorithm 2	Arno, Dylan
6. Evaluating and Comparing Algorithms	Arno, Dylan
7. Writing Report	Robert, Dylan, Arno, Brian
8. Creating Presentation Slides	Robert, Dylan, Arno, Brian
9. Presented	Robert, Dylan, Arno, Brian, Jordan