Movie and Video Recommendation System using Machine learning

Shaun Chiang Kum Wah
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Technology Park Malaysia
57000 Kuala Lumpur, Malaysia.
tp062483@mail.apu.edu.my

Adeline Sneha J
Senior Lecturer/School of Computing
Asia Pacific University of Technology and Innovation (APU)
Technology Park Malaysia
57000 Kuala Lumpur, Malaysia.
adeline.john@apu.edu.my

Saif Ahmad Saif Salem Ali
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Technology Park Malaysia
57000 Kuala Lumpur, Malaysia.
tp072853@mail.apu.edu.my

Leong Jia Jun
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Technology Park Malaysia
57000 Kuala Lumpur, Malaysia.
tp065960@mail.apu.edu.my

Kamalanathan Shanmugam
Senior Lecturer / School of Technology
Asia Pacific University of Technology and Innovation (APU)
Technology Park Malaysia
57000 Kuala Lumpur, Malaysia.
kamalanathan@apu.edu.my

Mohamed Faroug Mohamed Abdelrahman
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Technology Park Malaysia
57000 Kuala Lumpur, Malaysia.
tp071240@mail.apu.edu.my

***Abstract*— The rise of recommendation systems being implemented in all aspects of our lives requires us to have a basic understanding regarding these algorithms. This study focuses on an item-based collaborative filtering recommendation system used for movies and show recommendation will provide insight into the parameters used for said recommendation such as genre, rating, and director of the movie/show.**

Keywords*— Machine Learning, Recommendation System, Item-Based Collaborative Filtering

I. INTRODUCTION

With the ever-increasing amount of options found on video streaming services it has become common practice to employ a recommendation system to help users make an informed selection based on their previous choices and preferences. This system does come with its own issues however, new users of the system would face the cold start problem where there is little to no data for the algorithm to help make an informed decision for the user. This paper is using the Netflix Movies and TV Shows dataset to simulate an environment with data that has been rated by other users along with using the Item-Based collaborative filtering so new users can still use the recommendation system. The purpose of this research is to compare several selections of parameters to find the most suitable for movie and video recommendations.

II. LITERATURE REVIEW

A. Similar Projects

Multiple research papers have used recommendation systems in other fields considering a varying amount of parameters using ML algorithms. This section reviews a few studies using recommendation systems to find similarities of algorithms used.

Soumya et al. (Sri Attaluri, K Batcha, & Mafas, 2020) had used recommendation systems and feature extraction to recommend crops for farmers to plant. Feature extraction is a process where features of a data entry in a dataset is isolated to better filter or sort entries that contains specific features for recommendation or removal. The paper provided evidence that historical and forecasted data is able for us to make reasonably accurate recommendations.

Joanne et al. (Lim Jo En, Kae Li, Davina, Vern Jian, & Arabee Bin Abdul Salam, 2024) conducted a research paper where they used recommendation systems and SVD to recommend products. Their paper goes in depth on how to mitigate the cold start problem that naturally occurs in all recommendation systems by comparing different approaches that tackle the problem along with the performance of collaborative filtering systems when SVD matrix's latent features are manipulated.

Smith Johnson (Smith, 2024). conducted a comprehensive study on the application of hybrid recommendation systems in enhancing user engagement and accuracy. Their research delves into the integration of content-based and collaborative filtering techniques, with a special focus on addressing the sparsity problem prevalent in user-item interaction data. By comparing various hybrid models, the study provides insights into the effectiveness of combining model-based and memory-based approaches. Moreover, the research explores the impact of incorporating contextual information and user feedback loops on the performance of recommendation systems. The paper provides a detailed analysis of the scalability of hybrid systems and their ability to adapt to dynamic user preferences, thereby offering a significant contribution to the field of personalized recommendation systems.

B. Methodology/Approach

This part will explore previous research and studies that has used Machine Learning algorithms to approach parameter selections for their respective recommendation systems.

Dimitris et al. (Kalimeris, Kalyanaraman, Bhagat, & Weinsberg, 2021)has determined that repeated used of a recommendation system by a user can cause the user to narrow down the recommendations through preference amplification which can lead to increased engagement but also lack of diversity and echo chambers. Through the selection of different parameters, it can be used to mitigate these issues and reduce potentially objectionable content by the user to increase their engagement.

C. Conclusion/Recommendation

Item-based collaborative filtering and similarity-based collaborative filtering has been proved to provide accurate recommendations for the user. To achieve a high accuracy on recommendations several parameter sets are used to find the selection that gives the most accurate recommendations. For this paper, 2 selections of data features will be used for the experiment done in this paper for reference of parameters in item-based collaborative filtering.

III. Materials and Methods

A. Machine Learning Algorithms

- Item-Based Collaborative Filtering
- Similarity-based Collaborative Filtering

Item-Based collaborative filtering and similarity-based collaborative filtering are techniques in machine learning algorithms that analyze item parameters and generate relevant recommendations. The source code used in this paper was created by User Erin Ward in Kaggle. We have modified the parameters of this code for this work. Similarity-Based Collaborative filtering recommends similar videos that share similar parameter types selected by us and recommends the top 10 videos based on said selected parameters.

B. Software Requirements

- Python
- Google Collab

The source code has been imported from Kaggle to Google Collab to be run by Google's provided servers for Collab. Running it on Google Collab also gives us the opportunity to edit the parameters from anywhere.

IV. Algorithm Implementation

A. Purpose

This paper was conducted to investigate how item-based collaborative filtering and similarity-based collaborative filtering is used to predict recommendations for the user. Using the selected parameters, the code recommends other movies/shows when they have matching parameters. Our goal here is to find out which sets of parameters would give us the best accuracy for recommendations.

B. Parameters

- Recommendation system using country, cast, director, ratings, and genre.

  The first parameter set takes account of the country of origin, the members of the cast, the director of the film/show, user ratings of the show and the genre it belongs in. This gives the algorithm a good set of parameters to find recommendations for the user.

- Recommendation system using show description.

  This second parameter set only uses the description provided by the show's page to find relevant shows to recommend to the user.

V. Results and Discussion

A. Discusion on implementation

```
import numpy as np
import pandas as pd
import re

import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.tokenize import word_tokenize
```

Figure 1. code to import necessary libraries

We first setup the experiment by importing the necessary packages such as NumPy, pandas, re and nltk.

```
data = pd.read_csv('../input/netflix-shows/netflix_titles.csv')
data.head()
```

Figure 2. loading the dataset and shows the top 10 entries

Next, we have to load the dataset which is "Netflix_titles.csv". This dataset holds thousands of entries of movies and shows hosted on the Netflix streaming service.

```
data = data.dropna(subset=['cast', 'country', 'rating'])
```

Figure 3. code that drops empty entries

Before we continue with building the recommendation engine, we first need to drop any null values found in the dataset.

```
movies = data[data['type'] == 'Movie'].reset_index()
movies = movies.drop(['index', 'show_id', 'type', 'date_added', 'release_year', 'duration', 'description'], axis=1)
movies.head()
```

	title	director	cast	country	rating	listed_in
0	7:19	Jorge Michel Grau	Demián Bichir, Héctor Bonilla, Oscar Serrano, ...	Mexico	TV-MA	Dramas, International Movies
1	23:59	Gilbert Chan	Tedd Chan, Stella Chung, Henley Hii, Lawrence ...	Singapore	R	Horror Movies, International Movies
2	9	Shane Acker	Elijah Wood, John C. Reilly, Jennifer Connelly...	United States	PG-13	Action & Adventure, Independent Movies, Sci-Fi...
3	21	Robert Luketic	Jim Sturgess, Kevin Spacey, Kate Bosworth, Aar...	United States	PG-13	Dramas
4	122	Yasir Al Yasiri	Amina Khalil, Ahmed Dawood, Tarek Lotfy, Ahmed...	Egypt	TV-MA	Horror Movies, International Movies

Figure 4. code that drops unwanted parameters

```
actors = []

for i in movies['cast']:
    actor = re.split(r', \s*', i)
    actors.append(actor)

flat_list = []
for sublist in actors:
    for item in sublist:
        flat_list.append(item)

actors_list = sorted(set(flat_list))

binary_actors = [[0] * 0 for i in range(len(set(flat_list)))]

for i in movies['cast']:
    k = 0
    for j in actors_list:
        if j in i:
            binary_actors[k].append(1.0)
        else:
            binary_actors[k].append(0.0)
        k+=1

binary_actors = pd.DataFrame(binary_actors).transpose()
```

Figure 5. code segment to sort actors

For the recommendation engine using cast, director, country, rating, and genre all parameters except the selected parameters are dropped to load the dataset faster. Now we read all actors and count them into the "binary_actors" list to better read the data as it goes through the recommender, all other parameters go through the same process and placed in their respective lists.

```
def recommender(search):
    cs_list = []
    binary_list = []
    if search in movies['title'].values:
        idx = movies[movies['title'] == search].index.item()
        for i in binary.iloc[idx]:
            binary_list.append(i)
        point1 = np.array(binary_list).reshape(1, -1)
        point1 = [val for sublist in point1 for val in sublist]
        for j in range(len(movies)):
            binary_list2 = []
            for k in binary.iloc[j]:
                binary_list2.append(k)
            point2 = np.array(binary_list2).reshape(1, -1)
            point2 = [val for sublist in point2 for val in sublist]
            dot_product = np.dot(point1, point2)
            norm_1 = np.linalg.norm(point1)
            norm_2 = np.linalg.norm(point2)
            cos_sim = dot_product / (norm_1 * norm_2)
            cs_list.append(cos_sim)
        movies_copy = movies.copy()
        movies_copy['cos_sim'] = cs_list
        results = movies_copy.sort_values('cos_sim', ascending=False)
        results = results[results['title'] != search]
        top_results = results.head(5)
        return(top_results)
```

Figure 6. the code block for the recommender function

Once all of the parameters are properly prepared, it runs through the recommender function where the Item-based collaborative filtering is coded in. the entry typed in the search argument of the function is then compared with the rest of the dataset and the top 5 results are then showed in a table as recommendations for the user.

```
movies_des = data[data['type'] == 'Movie'].reset_index()
movies_des = movies_des[['title', 'description']]
movies_des.head()
```

	title	description
0	7:19	After a devastating earthquake hits Mexico Cit...
1	23:59	When an army recruit is found dead, his fellow...
2	9	In a postapocalyptic world, rag-doll robots hi...
3	21	A brilliant group of students become card-coun...
4	122	After an awful accident, a couple admitted to ...

Figure 7. extraction of title and description

For the recommendation system using the description of the shows, only the title and description parameters are saved into the list for further use in the system.

```
filtered_movies = []
movies_words = []

for text in movies_des['description']:
    text_tokens = word_tokenize(text)
    tokens_without_sw = [word.lower() for word in text_tokens if not word in stopwords.words()]
    movies_words.append(tokens_without_sw)
    filtered = (" ").join(tokens_without_sw)
    filtered_movies.append(filtered)

movies_words = [val for sublist in movies_words for val in sublist]
movies_words = sorted(set(movies_words))
movies_des['description_filtered'] = filtered_movies
movies_des.head()
```

	title	description	description_filtered
0	7:19	After a devastating earthquake hits Mexico Cit...	after devastating earthquake hits mexico city ...
1	23:59	When an army recruit is found dead, his fellow...	when army recruit found dead , fellow soldiers...
2	9	In a postapocalyptic world, rag-doll robots hi...	in postapocalyptic world , rag-doll robots hid...
3	21	A brilliant group of students become card-coun...	a brilliant group students card-counting exper...
4	122	After an awful accident, a couple admitted to ...	after awful accident , couple admitted grisly ...

Figure 8. description is put in lowercase and each word is given a token.

The dataset is further processed by putting all works in the description to lowercase and then any unique words is given a token for the recommender system to read.

```
def recommender2(search):
    cs_list = []
    binary_list = []
    if search in movies_des['title'].values:
        idx = movies_des[movies_des['title'] == search].index.item()
        for i in movie_word_binary.iloc[idx]:
            binary_list.append(i)
        point1 = np.array(binary_list).reshape(1, -1)
        point1 = [val for sublist in point1 for val in sublist]
        for j in range(len(movies_des)):
            binary_list2 = []
            for k in movie_word_binary.iloc[j]:
                binary_list2.append(k)
            point2 = np.array(binary_list2).reshape(1, -1)
            point2 = [val for sublist in point2 for val in sublist]
            dot_product = np.dot(point1, point2)
            norm_1 = np.linalg.norm(point1)
            norm_2 = np.linalg.norm(point2)
            cos_sim = dot_product / (norm_1 * norm_2)
            cs_list.append(cos_sim)
        movies_copy = movies_des.copy()
        movies_copy['cos_sim'] = cs_list
        results = movies_copy.sort_values('cos_sim', ascending=False)
        results = results[results['title'] != search]
        top_results = results.head(5)
        return(top_results)
```

Figure 9. The code block for the description recommender system

Once the dataset is prepared, it is run through the recommender system comparing the tokens of the description with the data entry selected for the function's argument to tally up the amount of tokens that match with the selected entry before displaying the top 5 in a table for the user to get their recommendations from.

B. *Results*

```
recommender('The Conjuring')
```

	title	director	cast	country	rating	listed_in
1868	Insidious	James Wan	Patrick Wilson, Rose Byrne, Lin Shaye, Ty Simp...	United States, Canada, United Kingdom	PG-13	Horror Movies, Thrillers
968	Creep	Patrick Brice	Mark Duplass, Patrick Brice	United States	R	Horror Movies, Independent Movies, Thrillers
1844	In the Tall Grass	Vincenzo Natali	Patrick Wilson, Laysla De Oliveira, Avery Whit...	Canada, United States	TV-MA	Horror Movies, Thrillers
969	Creep 2	Patrick Brice	Mark Duplass, Desiree Akhavan, Karan Soni	United States	TV-MA	Horror Movies, Independent Movies, Thrillers
1077	Desolation	Sam Patton	Jaimi Paige, Alyshia Ochse, Toby Nichols,	United States	TV-MA	Horror Movies, Thrillers

Figure 10. results of the recommender system on the movie "The Conjuring"

The results shown in figure 4 are from the recommendation system using the 5 parameters. The algorithm took around 5 minutes to generate the recommendations. Manually reviewing the top recommendations shown at figure 4 that they share the thriller genre tag thus making the recommendation valid.

```
[15] recommender2("Dr. Seuss' The Cat in the Hat")
```

	title	description	description_filtered	cos_sim
1016	Dark Light	Implicated in her daughter's disappearance, a ...	implicated daughter 's disappearance , mother ...	0.500343
1660	He's Out There	While vacationing at a remote lake house, a mo...	while vacationing remote lake house , mother d...	0.480769
3808	The Breadwinner	A courageous 11-year-old Afghan girl disguises...	a courageous 11-year-old afghan girl disguises...	0.471782
1853	INDIA	A man buys a young girl, code-names her "Doll"...	a buys young girl , code-names `` doll '' send...	0.463281
766	Pygoy Re Pygoy	A mother's plan to find her bachelor son a mat...	a mother 's plan find bachelor match derails p...	0.462968

Figure 11. results of the description recommender system on the move "Dr. Suess' The Cat in the Hat

The results here in Figure 11 show the recommendations of the description recommender system. As it currently stands, this version of the recommender system does not have any code to conduct semantic analysis thus taking each unique word as the same meaning leading to unreliable recommendations like the one shown in figure 11 where it recommended horror genre movies when a movie under the kids genre is used for the search.

VI. CONCLUSIONS

In conclusion, the recommendation system is able to be used with a wide selection of parameters but might become unreliable if given complex parameters such as descriptions. Our testing has shown that when given set values in parameters such as genre, ratings, or country. Complex parameters such as descriptions need to be processed by hybrid recommendation systems that incorporate other algorithms such as the Natural Language Processing (NLP) model. In future works, we would attempt to make such advanced recommendation systems to further our understanding of the Recommendation system algorithm.

ACKNOWLEDGEMENTS

We would like to give our thanks to User Erin Ward on Kaggle who has provided us with the source code that allowed us to conduct experiments by changing the parameters of the code. We would also like to give our heartfelt appreciation towards the researchers that have been cited in our research project as they provided us with an immense amount of insight regarding the topic we have selected.

VII. REFERENCES

Kalimeris, D., Kalyanaraman, S., Bhagat, S., & Weinsberg, U. (2021). Preference Amplification in Recommender Systems. *KDD '21: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 805-815.

Lim Jo En, J., Kae Li, C., Davina, L. Z., Vern Jian, C., & Arabee Bin Abdul Salam, Z. (2024). Product Recommendation System using SVD and Machine Learning techniques. *Journal of Applied Technology and Innovation (e -ISSN: 2600-7304) vol. 8, no.1*, 21-24.

Smith, J. (2024). "Enhancing User Engagement and Accuracy in Recommendation Systems: A Comparative Study of Hybrid Approaches". *"Journal of Advanced Computational Intelligence and Informatics"*.

Sri Attaluri, S., K Batcha, N., & Mafas, R. (2020). Crop Plantation Recommendation using Feature Extraction and Machine Learning Techniques. *Journal of Applied Technology and Innovation (e -ISSN: 2600-7304) vol. 4, no. 4*, 4.