Algorithms for Information Retrieval and Intelligence Web: Assignment-2

Anime Recommender System

Submitted By:

Varun H Ramurs	PES1UG20CS487
Thamankar Pracheth Prashant	PES1UG20CS469
Vaibhav Vijay	PES1UG20CS479

Table of Contents

Problem Statement

To design and develop a personalised anime recommendation system that accurately suggests anime titles based on the user's viewing history and preferences, as well as incorporating additional data such as genre, rating, and popularity, to improve the user's overall viewing experience and satisfaction.

Introduction

Anime, or Japanese animated productions, have gained widespread popularity and have become an important part of popular culture worldwide. With so many anime titles available, it can be overwhelming for viewers to find their next favourite show. This is where an anime recommender system comes in. An anime recommender system is an intelligent tool that suggests anime titles to users based on their viewing history, preferences, and other data points such as genre, rating, and popularity. By using machine learning algorithms and predictive analytics, these systems provide personalised recommendations that can help users discover new anime shows and improve their overall viewing experience.

Dataset Description

This data set contains information on user preference data from 73,516 users on 12,294 anime. Each user is able to add anime to their completed list and give it a rating and this data set is a compilation of those ratings.

1. anime.csv:
   anime_id - myanimelist.net's unique id identifying an anime.
   name - full name of anime.
   genre - comma separated list of genres for this anime.
   type - movie, TV, OVA, etc.
   episodes - how many episodes in this show. (1 if movie).
   rating - average rating out of 10 for this anime.
   members - number of community members that are in this anime's "group".

2. rating.csv:
   user_id - non identifiable randomly generated user id.
   anime_id - the anime that this user has rated.
   rating - rating out of 10 this user has assigned (-1 if the user watched it but didn't assign a rating).

EDA and Pre-processing

1. Handling NaN values: First we have to take care of the NaN values, as this revolves around ratings, a user who hasn't given any ratings has added no value to the engine. So let's drop the NaN values.

2. Filtering: Filter based on user_id. There are users who has rated only once, even if they have rated it 5, it can't be considered a valuable record for recommendation. So we have considered minimum 200 ratings by the user as threshold value.

3. Pivot Table: Consists of rows as title and columns as user id, this will help us to create sparse matrix which can be very helpful in finding the cosine similarity.

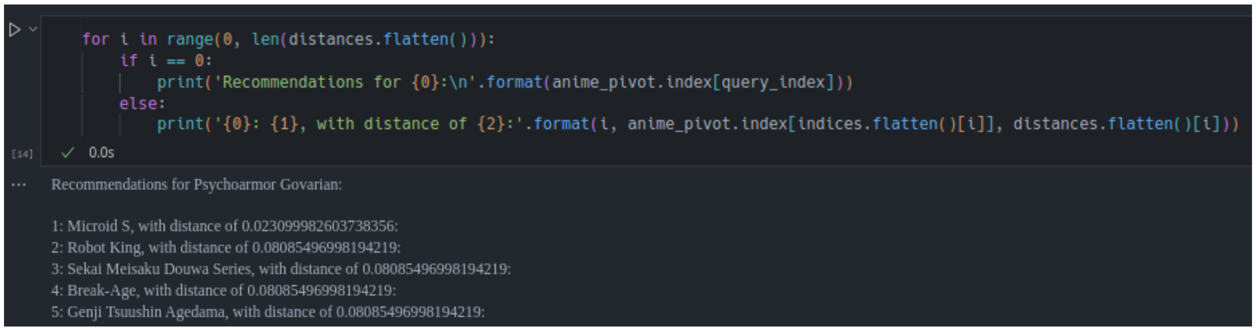
Collaborative Filtering

Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users. It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user.

Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.

We have fitted the sparse matrix(pivot table), let's get a random anime title and find recommendation for it. We are returning the distances and indices of 5 neighbours through KNN from the randomly chosen index(anime_title) those will be our recommended anime's.

Results of Collaborative Filtering using KNN



```
for i in range(0, len(distances.flatten())):
    if i == 0:
        print('Recommendations for {0}:\n'.format(anime_pivot.index[query_index]))
    else:
        print('{0}: {1}, with distance of {2}:'.format(i, anime_pivot.index[indices.flatten()[i]], distances.flatten()[i]))
```

```
[14] ✓ 0.0s
```

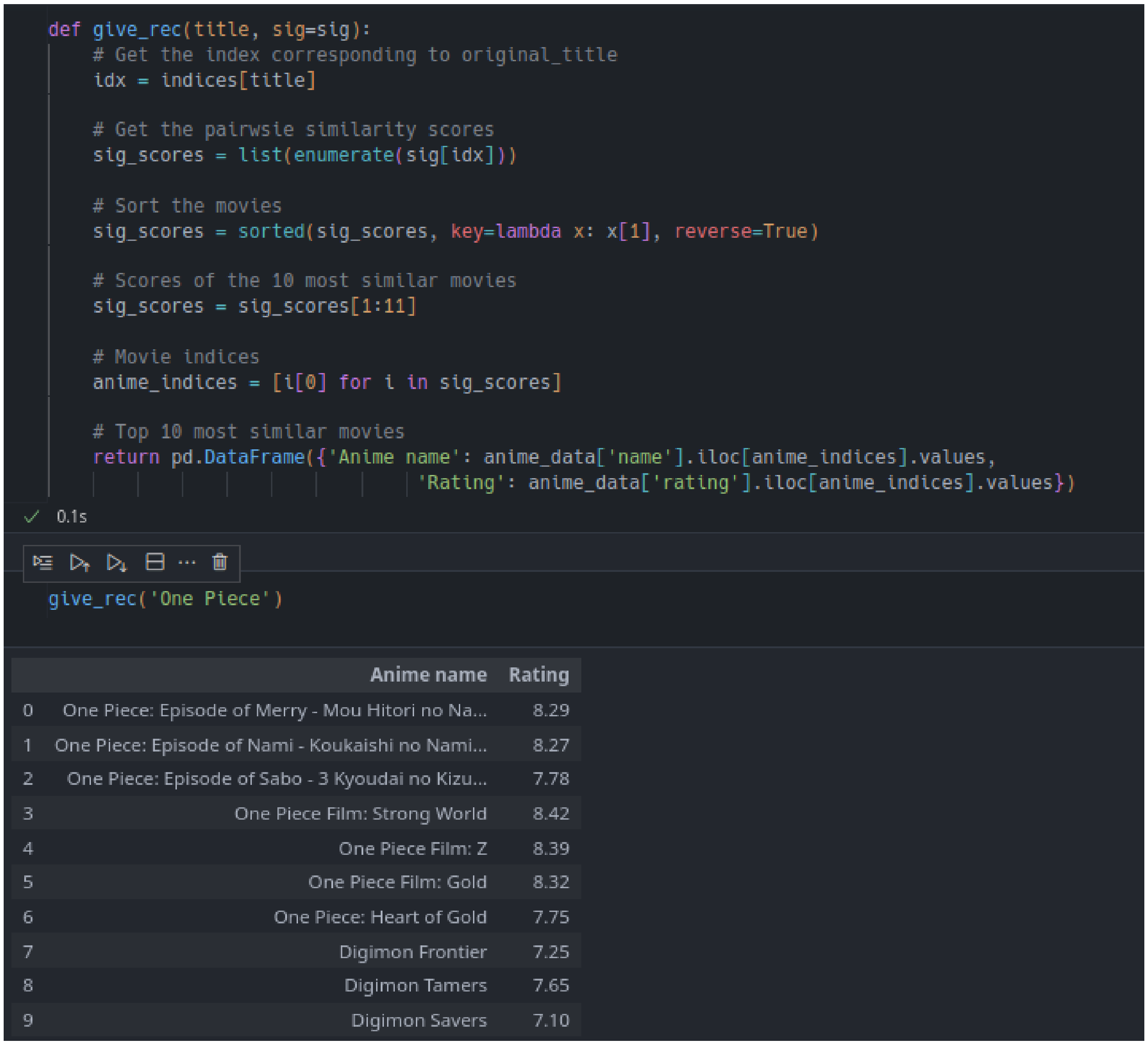
```
Recommendations for Psychoarmor Govarian:

1: Microid S, with distance of 0.023099982603738356:
2: Robot King, with distance of 0.08085496998194219:
3: Sekai Meisaku Douwa Series, with distance of 0.08085496998194219:
4: Break-Age, with distance of 0.08085496998194219:
5: Genji Tsuushin Agedama, with distance of 0.08085496998194219:
```

Content Based Filtering

Content-based filtering uses other users ratings to give recommendations about which anime’s similar to the inputted anime title. We first clean the title and then apply ultimate tf-idf to recommend us the next anime. Scikit-learn provides pairwise metrics that work for both dense and sparse representations of vector collections. Here we need to assign 1 for recommended anime and 0 for not recommended anime. So we are using sigmoid kernel.

Recommendation function: here we create the function for getting the recommendation for an anime. We turn the similarity scores into lists using enumerate function, sort the list and select the top 10 score for recommendation.

Results of Content Based Filtering



```python
def give_rec(title, sig=sig):
    # Get the index corresponding to original_title
    idx = indices[title]

    # Get the pairwsie similarity scores
    sig_scores = list(enumerate(sig[idx]))

    # Sort the movies
    sig_scores = sorted(sig_scores, key=lambda x: x[1], reverse=True)

    # Scores of the 10 most similar movies
    sig_scores = sig_scores[1:11]

    # Movie indices
    anime_indices = [i[0] for i in sig_scores]

    # Top 10 most similar movies
    return pd.DataFrame({'Anime name': anime_data['name'].iloc[anime_indices].values,
                         'Rating': anime_data['rating'].iloc[anime_indices].values})
```

✓ 0.1s

```python
give_rec('One Piece')
```

	Anime name	Rating
0	One Piece: Episode of Merry - Mou Hitori no Na...	8.29
1	One Piece: Episode of Nami - Koukaishi no Nami...	8.27
2	One Piece: Episode of Sabo - 3 Kyoudai no Kizu...	7.78
3	One Piece Film: Strong World	8.42
4	One Piece Film: Z	8.39
5	One Piece Film: Gold	8.32
6	One Piece: Heart of Gold	7.75
7	Digimon Frontier	7.25
8	Digimon Tamers	7.65
9	Digimon Savers	7.10

Conclusions

Ultimately, the effectiveness of a recommendation system relies heavily on the specific dataset and the desired objective. Depending on the situation, collaborative filtering or content-based filtering may be more effective. Collaborative filtering is most beneficial when there is a robust user-item interaction history, whereas content-based filtering is advantageous when there is an abundance of item features available. Hybrid recommendation systems can blend both techniques to produce more accurate and varied recommendations. It is critical to thoroughly analyze the dataset and the desired outcome before selecting a recommendation technique or utilizing multiple approaches.

The success of a recommendation system hinges on the quality and usefulness of the recommendations to the end-users.