CS419(Project)

Movie Recommendation system

Team ID-12

TEAM	-Shivam Kumar	210020128
	- Kunal Mishra	210020066
	-Aman kushwah	210040014
	-Aryan Adinath Popalghat	210020088

Description

Given a dataset containing movies name,users and their rating to some movies,we are deciding whether a movie not watched by a particular user should be recommended to that user or not.

Literature survey

There are many approaches to this problem used by different prople.Some of them are:

- **Model-Based collaborative filtering**: Remembering the matrix is not required here. From the matrix, we try to learn how a specific user or an item behaves. We compress the large interaction matrix using dimensional Reduction or using clustering algorithms.There are several methods for this approach:-

1. 1. Clustering algorithms: They normally use simple clustering Algorithms like K-Nearest Neighbours to find the K closest neighbors or embeddings given a user or an item embedding based on the similarity metrics used.

2. Matrix Factorization based algorithm: we generate the factor matrices as feature matrices for users and items. These feature matrices serve as embeddings for each user and item
3. Deep Learning methods

- **Content-based filtering:** Content-based filtering uses item features to recommend other items similar to what the user likes, based on their

previous actions or explicit feedback.
(reference:https://developers.google.com/machine-learning/recommendation/content-based/basics)

- **Evaluation metrics:** Evaluation metrics for recommender systems have evolved; initially accuracy of predicted ratings was used as an evaluation metric for recommender systems. Netflix even started a competition to find the algorithm which could improve upon their accuracy by 10 percent for one million dollars. (reference: Recommendation System Evaluation Metrics | by Rishabh Bhatia | Medium)

- **Deep learning approaches**

Problem Statement:

Use the given dataset of movies and their ratings. (Given in the form of csv files). By using the following two methods recommend the list of movies to the user. Recommended movies should be those movies for which the user hasn't given any rating (rating=null/void). Means the user hasn't seen that movie. The output should contain movies recommended in decreasing order- means first is highly recommended and last is least recommended.
Method 1:
User-to-User collaborative filtering
Method 2:
Item-to-Item collaborative filtering

Approach:

1. **User-to-User collaborative filtering:**

   We used a **cosine similarity function** to find the most similar user.

   Cosine similarity between two vectors u and v: $Similarity(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{||\vec{A}|| * ||\vec{B}||}$

We will find the similarity between target users with all the other users and will eventually find the most similar user to the targeted user and will recommend the movies which are not watched by the targeted user and have been rated more than 3 by the most similar user.

Other's people approach

For calculating similarity,we can use different similarity functions.Some of them are:-

1.Euclidean distance:The element wise squared distance between two vectors.

2.Pearson Similarity: It is a coefficient given by:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

2. Item-to-Item collaborative filtering:

1. **First step:** Build the model by finding similarity between all the item pairs. Most common method to find similarity between item pairs is by using cosine similarity (which we have used)
2. **Formula for cosine similarity:** $Similarity(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{||\vec{A}|| * ||\vec{B}||}$
3. **Second step (prediction computation):** To execute the recommendation system. It uses the items (already rated by the user) that are most similar to the missing item to generate ratings. We try to generate predictions based on the ratings of similar products. We compute this using the following formula. This formula computes the rating of a particular item using a weighted sum of the ratings of other similar products.

$$rating(U, I_i) = \frac{\sum_j rating(U, I_j) * s_{ij}}{\sum_j s_{ij}}$$

4. In our case, the item means movie. In the above formula, rating(U , I_i) means rating given by user U to i^{th} item (again here item is movie) and s_{ij} means the similarity between the i^{th} and j^{th} movie.

Source of data:

https://www.kaggle.com/datasets/shubhammehta21/movie-lens-small-latest-dataset?select=movies.csv

Data explanation

movies.csv-> Contains 3 columns named: movieId , title , genres
movieId:Ids of different movies
title: Name of the movies

Dimension of the dataset:9742 rows × 3 columns

ratings.csv ->Contains 4 columns:
userId: Ids of different users
movieId: Ids of different movies
rating: rating given by a particular user to a particular movie
Timestamp:time spent in watching a particular movie by a particular user

Dimension of the dataset:(100836 rows × 4 columns)

Functions used in the User-to-User Approach code

1- similar_users(userID,user_ratings) ->

- Takes two arguments: **user_ratings** (Dataframe containing the movies in columns and userid's in rows) and **userID** (person id for which we have to recommend movies)
- **user_ratings[i][j]**=rating of movie 'j' by user having id 'i'.
- This calculates the cosine similarity between a user having an userID with all the other users.
- Returns **similar_users** (a 2d array containing **similarity value** and the **user id**)

2- movie_recommend(userID,user_ratings)->

- Takes two arguments **userID** (targeted user) and **user_ratings**(dataframe containing movie ratings) and returns the final list of recommendations.

Function used in the Item-to-Item Approach code

1-recommend(userID,user_ratings)->

- Takes two arguments: **user_ratings** (Dataframe containing the movies in columns and userid's in rows) and **userID** (person id for which we have to recommend movies).

- This function calculates a similarity matrix and stores it in the similiarty_matrix variable.
- Then using similarity matrix and user _ratings,we predict the rating given by the userID to the not watched movies using this formula:

$$rating(U, I_i) = \frac{\sum_j rating(U,I_j) * s_{ij}}{\sum_j s_{ij}}$$

2-movies_name(userID,user_ratings)->

- Calls function recommend to calculate the not watched movie ratings and stores it in recommended movies array if rating>=3
- Returns the list of movies which has predicted rating>=3

Results:

We will pass the data frame(containing userID as row labels and movies name as column labels) and the userID for which we are recommending the movies to movie_recommend function.This function will return the list of movies recommended to user having id=userID.

For userID 2 in the datasets ,

the movies recommended by User to User approach are->

'burbs, The (1989)'',
'(500) Days of Summer (2009)',
'10 Cloverfield Lane (2016)',
'10 Things I Hate About You (1999)',
'10,000 BC (2008)',
'101 Dalmatians (1996)',
'101 Dalmatians (One Hundred and One Dalmatians) (1961)',
'12 Angry Men (1957)',
'12 Years a Slave (2013)',
'127 Hours (2010)',
'13 Going on 30 (2004)',
'13th Warrior, The (1999)',
'1408 (2007)',
'15 Minutes (2001)',
'17 Again (2009)',
'1984 (Nineteen Eighty-Four) (1984)',
'2 Days in the Valley (1996)',
'2 Fast 2 Furious (Fast and the Furious 2, The) (2003)',
'20,000 Leagues Under the Sea (1954)',
'2001: A Space Odyssey (1968)',
'2010: The Year We Make Contact (1984)',
'2012 (2009)',
'21 (2008)',
'21 Grams (2003)',
'21 Jump Street (2012)',
'22 Jump Street (2014)',

'25th Hour (2002)',
'27 Dresses (2008)',
'28 Days (2000)

The movies recommended by item to item approach are->

['Inside Job (2010)', 5.0],
['Mad Max: Fury Road (2015)', 5.0],
['Step Brothers (2008)', 5.0],
['Warrior (2011)', 5.0],
['Wolf of Wall Street, The (2013)', 5.0],
['Dark Knight, The (2008)', 4.5],
['Good Will Hunting (1997)', 4.5],
['Inglourious Basterds (2009)', 4.5],
['Town, The (2010)', 4.5],
['Departed, The (2006)', 4.0],
['Gladiator (2000)', 4.0],
['Inception (2010)', 4.0],
['Kill Bill: Vol. 1 (2003)', 4.0],
['Shutter Island (2010)', 4.0],
['Talladega Nights: The Ballad of Ricky Bobby (2006)', 4.0],
['Tommy Boy (1995)', 4.0],
['Whiplash (2014)', 4.0],
['Collateral (2004)', 3.5],
['Dark Knight Rises, The (2012)', 3.5],
['Django Unchained (2012)', 3.5],
['Ex Machina (2015)', 3.5],
['Exit Through the Gift Shop (2010)', 3.0],
['Interstellar (2014)', 3.0],
['Shawshank Redemption, The (1994)', 3.0],
['Zombieland (2009)', 3.0]

Resources referred for this project:

https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109

https://www.geeksforgeeks.org/item-to-item-based-collaborative-filtering/

Build a Recommendation Engine With Collaborative Filtering – Real Python

https://towardsdatascience.com/introduction-to-recommender-systems-1-971bd274f421