Introduction

In this project (Recome), we have developed a website based on collaborative filtering recommender (CFR) system for recommending movies .

The basic idea of CFR systems is that, if two users share the same interests in the past, e.g. they liked the same book or the same movie, they will also have similar tastes in the future. If, for example, user A and user B have a similar purchase history and user A recently bought a book that user B has not yet seen, the basic idea is to propose this book to user B.

The collaborative filtering approach considers only user preferences and does not take into account the features or contents of the items (books or movies) being recommended. In this project, in order to recommend movies we used a large set of users preferences towards the movies from a publicly available movie rating dataset.

Dataset

The dataset used was from MovieLens, which consisted of 100005 ratings given by 671 users for 9084 movies. Each user in this dataset has rated atleast 20 movies. This dataset was last updated on 10/2016. The link for the dataset is http://files.grouplens.org/datasets/movielens/ml-latest-small.zip
The files we used from this dataset are links.csv, movies.csv, ratings.csv

Links dataframe overview:

MovieId	ImdbId	TmdbId
1	114709	862
2	113497	8844
3	113228	15602
4	114885	31357
5	113041	11862

Movies dataframe overview:

MovieId	Title	Genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy

Ratings dataframe overview:

UserId	MovieId	Rating
1	31	2.5
1	1029	3
1	1061	3
1	1129	2
1	1172	4

Software Requirements

1. Python 3.x
   Libraries Used: sklearn, numpy, pymongo, pandas
   Web Framework: Flask
2. MongoDB

Project Implementation

Web Scraping:

On our website, we want to provide users total information about our movie. So we used tmdb links for each movie to gather information about it. This is done by using **bs4** library in python. After web scraping, we have got all the information about the movie.

Example:

TmdbId: 862

MovieName: Toy Story

Year: 1995

Crew: {'Screenplay': ['Alec Sokolow', 'Joel Cohen', 'Joss Whedon', 'Andrew Stanton'], 'Director': ['John Lasseter']}

Cast: ['Tom Hanks', 'Tim Allen', 'Don Rickles', 'Jim Varney', 'Wallace Shawn']

Description: Led by Woody, Andy's toys live happily in his room until Andy's birthday brings Buzz Lightyear onto the scene. Afraid of losing his place in Andy's heart, Woody plots against Buzz. But when circumstances separate Buzz and Woody from their owner, the duo eventually learns to put aside their differences.

ReleaseDate: October 2, 2009

Runtime: 1h 21min

Language: English

ImgLink: https://image.tmdb.org/t/p/w300_and_h450_bestv2/rhIRbceoE9lR4veEXuwCC2wARtG.jpg

MovieId: 1

ImdbId: 114709

Genres: Adventure|Animation|Children|Comedy|Fantasy

Data Preprocessing:

The data we have collected through webscraping has some errors in it. First we dropped the duplicates with same tmdb id. Then, we eliminated the movies from the dataset whose tmdb link is not responding.

Then we have created a dictionary with genres. For every movie, we have filled this dictionary with value 1, if movie belong to that genre and with 0, if movie doesn't belong to it.

Example:

For Toy Story, the dictionary will be:

{'Comedy': 1, 'Horror': 0, 'Children': 1, 'Action': 0, 'Adventure': 1, 'Film-Noir': 0, 'Western': 0, 'Drama': 0, 'Fantasy': 1, 'Crime': 0, 'Romance': 0, 'IMAX': 0, 'Animation': 1, 'Mystery': 0, 'Documentary': 0, 'Thriller': 0, 'War': 0, 'Musical': 0, 'Sci-Fi': 0}

<u>Structure of Data in MongoDB:</u>

Database Name: test

<u>Collections:</u>

1. movies:
   ['_id', 'movieId', 'movieName', 'description', 'releaseDate', 'runTime', 'language', 'cast', 'crew', 'imgLink', 'tmdbId', 'imdbId', 'genres', 'avgRating']
2. normalized_matrix:
   ['_id', 'userId', 'ratings']
3. movie_indexes:
   ['_id', 'col', 'index']
4. users:
   ['_id', 'first_name', 'last_name', 'userId', 'password', 'email', 'ratings', 'interests']
5. ratings_by_movie:
   ['_id', 'movieId', 'ratings']
6. ratings_by_user:
   ['_id', 'userId', 'ratings', 'genres']

Approaches Used:

Content Based:

In this approach, we tried to calculate the user interest on each genre. For a particular user, his interest on a particular genre is given by

$$g_i = (\sum_{j=1}^{n} r_j)/n$$

g_i is the amount how much user like this genre.

r_j is the rating user i has given for this movie.

The RMSE(Root Mean Square Error) error we got on test data set is 0.688483320746.

Drawback:

While predicting the ratings for a user, movies with same genre are always getting the same predicted rating. As movies with same genres are getting equal prediction, this is not giving the result we expected, so we stopped proceeding in this approach.

Matrix Factorization:

Using Matrix factorization, as there are 19 genres, we have done matrix factorization using k=19. Based on the speed of our laptops we have taken steps=100, as it is taking very large time.

As steps we have taken are very less, the RMSE error is 3.60134741628. As the error is large, and on performance of our cpu's, we discarded this approach.

Cosine Similarity:

For every user we normalized his ratings by subtracting his average rating from given ratings. This makes average rating of every user to 0. So it removes the effect of users giving high (or low) ratings to all their movies might bias the results.

Then we find the cosine similarity between every pair of users. Based on the cosine similarity we get the nearest neighbours for every user.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

<u>Sample of ratings for 5 users and 5 movies:</u>

	MOVIE-1	MOVIE-2	MOVIE-3	MOVIE-4	MOVIE-5
USER-1	0.5		3.5	2	
USER-2		3	1	1	1
USER-3	4	3.5	4.5		4.5
USER-4		4		2	
USER-5	4	3		3.5	4

Normalizing the matrix:

	MOVIE-1	MOVIE-2	MOVIE-3	MOVIE-4	MOVIE-5
USER-1	-1.5		1.5	0	
USER-2		1.5	-0.5	-0.5	-0.5
USER-3	-0.125	-0.625	0.375		0.375
USER-4		1		-1	
USER-5	0.375	-0.625		-0.125	0.375

Cosine Similarity between every pair of users:

	USER-1	USER-2	USER-3	USER-4	USER-5
USER-1	1	0.204	0.426	0	-0.319
USER-2	0.204	1	-0.914	-0.426	-0.769
USER-3	0.426	-0.914	1	-0.533	0.705
USER-4	0	-0.426	-0.533	1	-0.426
USER-5	-0.319	-0.769	0.705	-0.426	1

From the above information we can see that most similar to user 3. So while giving predictions for user5, we can use ratings given by user 3.

Why do we normalize:

	MOVIE-1	MOVIE-2
USER-1	1	3
USER-2	5	3
USER-3	2.5	4.5

In the above table, we can see that user 2 interests are opposite to user3. But if we calculate cosine of angle, the value will be: 0.86618 .

After normalizing

	MOVIE-1	MOVIE-2
USER-1	-1	1
USER-2	1	-1
USER-3	-1	1

Now if we measure cosine of angle between user 2 and user 3, the value will be:

Collaborative Filtering:

By using the cosine similarities, we found for every pair of users, we get the nearest neighbours for every user. Then for every movie of a user, we get the normalized ratings of users who are most similar to him and watched that particular movie. Then we use the following formula to calculate the predicted rating.

Predicted rating for movie k of user i =

((∑ normalized rating of user j who watched movie k * similarity of user i and j)/∑ similarity of users) + average rating of user i .

(We calculated predicted from 40 most similar users)

Note: The predicted rating may be greater than the 5 or less than 5 while calculating predicted rating.

So we scaled them to the range of 0-5.

We got an RMSE error of: 0.95140684173

So accuracy in predicting is: 80.97%

Limitations:

1. As there is a case where the most nearest neighbours may not see the particular movie, so that the deviation will be high which effects the user predicted rating for that movie.
2. For a new user, we don't have enough information to know his behaviour. So, we are actually asking him to give his interested genres. After he has rated 15 movies, we will calculate the predict ratings. Otherwise, recommended will be the top rated in his interested genre.