|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Kingdom of Saudi Arabia**  **Ministry of Education**  **University of Jeddah**  **College of Computer Science and Engineering**  **Department of Computer Science and Artificial Intelligence** | Logo, company name  Description automatically generated | **المملكة العربية السعودية**  **وزارة التعليم**  **جامعة جدّة**  **كلية علوم وهندسة الحاسب**  **قسم علوم الحاسب والذكاء الاصطناعي** | |  |  |

|  |
| --- |
| **Lab 2** |
| **CCAI 422 Recommender Systems** |
| **Second Trimester 2024**   |  |  | | --- | --- | | **Lab Date/Time: 31th January 2024.**  **Lab assignment submission Date/Time: at the end of the lab** |  | | **Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Student ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | |

|  |  |
| --- | --- |
| **Instructor Name** | **Section** |
|  |  |

**Instructions**:

The lab assignments must be submitted before the allocated Date/Time.

The lab assignments must by uploaded on LMS / sent by email to teacher@uj.edu.sa.

Plagiarism will be punished according to university rules.

|  |  |
| --- | --- |
| **PLO/CLO** | **SO** |
| **PLO S2 (CLO 2):** Demonstrate the ability of applying tools, techniques and practices required for problem solving in the domain of recommender systems | **SO 2:** Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program’s discipline |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Max Score** | **Student Score** |
| **PLO S2 / CLO 2 / SO 2** | **Task 1** | **5** |  |
| **Total** | |  |  |

**Lab description:** In this lab, you will build a simple non-personalised recommender. Building the simple recommender is fairly straightforward. The steps are as follows:

1. Choose a metric (or score) to rate the movies on
2. Decide on the prerequisites for the movie to be featured on the chart
3. Calculate the score for every movie that satisfies the conditions
4. Output the list of movies in decreasing order of their scores

The metric is the numeric quantity based on which we rank movies. A movie is considered to be better than another movie if it has a higher metric score than the other movie. It is very important that we have a robust and a reliable metric to build our chart upon to ensure a good quality of recommendations.

The choice of a metric is arbitrary. One of the simplest metrics that can be used is the movie rating. However, this suffers from a variety of disadvantages. In the first place, the movie rating does not take the popularity of a movie into consideration. Therefore, a movie rated 9 by 100,000 users will be placed below a movie rated 9.5 by 100 users.

This is not desirable as it is highly likely that a movie watched and rated only by 100 people caters to a very specific niche and may not appeal as much to the average person as the former.

It is also a well-known fact that as the number of voters increase, the rating of a movie normalizes and it approaches a value that is reflective of the movie's quality and popularity with the general populace. To put it another way, movies with very few ratings are not very reliable. A movie rated 10/10 by five users doesn't necessarily mean that it's a good movie.

Therefore, what we need is a metric that can, to an extent, take into account the movie rating and the number of votes it has garnered (a proxy for popularity). This would give a greater preference to a blockbuster movie rated 8 by 100,000 users over an art house movie rated 9 by 100 users.

Fortunately, we do not have to brainstorm a mathematical formula for the metric. As the title of this chapter states, we are building an IMDB top 250 clone. Therefore, we shall use IMDB's weighted rating formula as our metric. Mathematically, it can be represented as follows:

Weighted Rating (WR) =

Shape

Description automatically generated with medium confidence

The following apply:

* *v* is the number of votes garnered by the movie
* *m* is the minimum number of votes required for the movie to be in the chart (the prerequisite)
* *R* is the mean rating of the movie
* *C* is the mean rating of all the movies in the dataset

We already have the values for v and R for every movie in the form of the vote\_count and vote\_average features respectively. Calculating C is extremely trivial, as we have already seen in the previous chapter.

**The Pre-Reqs:** The IMDB weighted formula also has a variable m , which it requires to compute its score. This variable is in place to make sure that only movies that are above a certain threshold of popularity are considered for the rankings. Therefore, the value of m determines the movies that qualify to be in the chart and also, by being part of the formula, determines the final value of the score.

Just like the metric, the choice of the value of m is arbitrary. In other words, there is no right value for m. It is a good idea to experiment with different values of m and then choose the one that you (and your audience) think gives the best recommendations. The only thing to be kept in mind is that the higher the value of m, the higher the emphasis on the popularity of a movie, and therefore the higher the selectivity.

**Part 1: [PLO S2 / CLO 2 / SO 2] [5 marks]**

1. The first step in building our simple recommender is setting up our workspace. Let's create a new directory named Lab02. Create a Jupyter Notebook in this directory named Simple Recommender and open it in the browser. You can download the dataset used for this lab from Blackboard or from the link below:

<https://www.kaggle.com/rounakbanik/the-movies-dataset?select=movies_metadata.csv>

**Note:** The link the book seems outdated and is not working.

1. Import data into Python as below:

import pandas as pd

import numpy as np

#Load the dataset into a pandas dataframe

df = pd.read\_csv('../data/movies\_')

#Display the first five movies in the dataframe

df.head()

Upon running the cell, you should see a familiar table-like structure output in the notebook.

1. **The metric:** For our recommender, we will use the number of votes garnered by the 80th percentile movie as our value for m. In other words, for a movie to be considered in the rankings, it must have garnered more votes than at least 80% of the movies present in our dataset. Additionally, the number of votes garnered by the 80th percentile movie is used in the weighted formula described previously to come up with the value for the scores.

Let us now calculate the value of m:

#Calculate the number of votes garnered by the 80th percentile movie

m = df['vote\_count'].quantile(0.80)

m

OUTPUT:

50.0

We can see that only 20% of the movies have gained more than 50 votes. Therefore, our value of m is 50.

1. Another prerequisite that we want in place is the runtime. We will only consider movies that are greater than 45 minutes and less than 300 minutes in length. Let us define a new DataFrame, q\_movies, which will hold all the movies that qualify to appear in the chart:

#Only consider movies longer than 45 minutes and shorter than 300 minutes

q\_movies = df[(df['runtime'] >= 45) & (df['runtime'] <= 300)]

#Only consider movies that have garnered more than m votes

q\_movies = q\_movies[q\_movies['vote\_count'] >= m]

#Inspect the number of movies that made the cut

q\_movies.shape

OUTPUT:

(8963, 24)

We see that from our dataset of 45,000 movies approximately 9,000 movies (or 20%) made the cut.

1. **Calculating the score:** The final value that we need to discover before we calculate our scores is C, the mean rating for all the movies in the dataset:

# Calculate C

C = df['vote\_average'].mean()

C

OUTPUT:

5.6182072151341851

We can see that the average rating of a movie is approximately 5.6/10. It seems that IMDB happens to be particularly strict with their ratings. Now that we have the value of C, we can go about calculating our score for each movie.

1. First, let us define a function that computes the rating for a movie, given its features and the values of m and C:

# Function to compute the IMDB weighted rating for each movie

def weighted\_rating(x, m=m, C=C):

v = x['vote\_count']

R = x['vote\_average']

# Compute the weighted score

return (v/(v+m) \* R) + (m/(m+v) \* C)

1. Next, we will use the familiar apply function on our q\_movies DataFrame to construct a new feature score. Since the calculation is done for every row, we will set the axis to 1 to denote row-wise operation:

# Compute the score using the weighted\_rating function defined above

q\_movies['score'] = q\_movies.apply(weighted\_rating, axis=1)

1. **Sorting and output:** There is just one step left. We now need to sort our DataFrame on the basis of the score we just computed and output the list of top movies:

#Sort movies in descending order of their scores

q\_movies = q\_movies.sort\_values('score', ascending=False)

#Print the top 25 movies

1. Choose a revenue threshold and filter movies with revenue above that threshold.
2. Check for missing values in the dataframe
3. Visualize the distribution of movie ratings

Table

Description automatically generatedq\_movies[['title', 'vote\_count', 'vote\_average', 'score', 'runtime']].head(25)

And voila! You have just built your very first recommender. Congratulations!

We can see that the Bollywood film Dilwale Dulhania Le Jayenge figures at the top of the list. We can also see that it has a noticeably smaller number of votes than the other Top 25 movies. This strongly suggests that we should probably explore a higher value of m. This is left as an exercise for the reader; experiment with different values of m and observe how the movies in the chart change.