CSE 515 : Phase 1 – Analysis on IMDB+MovieLens Dataset - Group 3

Arun Srivatsa Ramesh, Achyutha Sreenivasa Bharadwaj, Kushal Bhatt, Ninad Jadhav, Pranav De

Abstract

1 Introduction

2 Terminology

2.1 Actor

Actor/Actress who has played a role in any movie.

2.2 Genre

Categories under which various movies are classified.

2.3 User

A person who rates and tags movies after he/she has watched them.

2.4 TF

Term Frequency - The frequency with which terms appear in a given document.

2.5 IDF

Inverse Document Frequency - This factor is to determine how important a term is in a document given the corpus.

3 Goal description

Given a dataset consisting of actors, movies, genres and user tags for movies with timestamps, the goal is to represent actors, genres and users in a tag vector space. Additionally, to provide a way of finding the most discriminating features(tags) give two genres.

4 Assumptions

4.1 General Assumptions

1. The data for user rating/tagging is accurate.
2. There exists no duplicate data in the given dataset.
3. Inputs will be given as expected in the specified format.

4.2 Task 1

1. One document is considered as one (actor,movie) pair.
2. If a tag t does not occur, the $tf_{actor,t}$ is assumed to be 0.

4.3 Task 2

1. One document is considered as one (genre,movie) pair.
2. If a tag t does not occur, the $tf_{genre,t}$ is assumed to be 0.

4.4 Task 3

1. One document is considered as one (user,movie) pair.
2. If a tag t does not occur, the $tf_{user,t}$ is assumed to be 0.
3. Users who have watched the movie include users who have rated the movie or users who have tagged the movie.

4.5 Task 4

1. TF is same as Task 2 TF. Only IDF is recomputed.
2. Smoothing technique to handle edge cases (from Section 12.4) is used for P-DIFF1 and P-DIFF2.

5 Description of the solution/implementation

In this project, we focus on representing actors, genres and users in a tag vector space and find a differentiating tag vector for a given genre with respect to another genre. For this process to be facilitated, MySQL database is used to store the dataset after preprocessing it.

5.1 Preprocessing

The first step of preprocessing would be the conversion of the given timestamps for tags into milliseconds. The given timestamps are in the format "YYYY-MM-DD HH-MM-SS". The conversion to milliseconds is done by taking the UNIX Epoch time as reference. After converting to milliseconds, a standard normalization is performed on the timestamps to fit in the range (0,1). This is performed to normalize the timestamps of the mltags.csv file containing the userid,movieid,tagid,timestamps.
In the mlmovies.csv file, the genre for each movie is listed with a seperator "|". Each movie row containing multiple genres is split into multiple rows with one genre each. For example, the movie "Scary Movie 2" has Comedy|Horror as its genres. It is split into two rows with Comedy and Horror as genres.
In the movie-actor.csv, the ranks of the actors range from 1 to 383. This is normalized to fit in the range (0,1).
A metadata table is create to store the distinct counts of all the rows in each table to provide constant lookup time.

5.2 Task 1

The main objective of this task is to represent actors as tag vectors. First, all the rows corresponding to that actor is filtered. Then one of the two following ways is used:

5.2.1 Term Frequency

The term frequency of an actor is defined as:

$$tf_{actor,t} = \frac{n_{actor,t}}{K_{actor}} \tag{1}$$

where,
$n_{actor,t} = sum(timestamps_{normalized} * ranks_{normalized})$ for tag t
$K_{actor} = sum(timestamps_{normalized} * ranks_{normalized})$ for all tags

The tf_{actor} is calculated for each tag. A tag vector consisting of all TF scores for each tag sorted in descending order of scores is printed.

5.2.2 Term Frequency-Inverse Document Frequency

The inverse document frequency of the actor is defined as:

$$idf_{actor,t} = log(\frac{N}{m_t}) \tag{2}$$

where,
N = Total number of actors from the entire database.
$m_t = sum(timestamps_{normalized})$ from actors containing tag t.

The score for each tag is then calculated by multiplying TF and IDF.

$$tfidf_{actor} = tf_{actor} * idf_{actor} \tag{3}$$

The $tfidf_{actor,t}$ is calculated for each tag. A tag vector consisting of all TFIDF scores $tfidf_{actor}$, sorted in descending order of scores is printed.

5.3 Task 2

The main objective of this task is to represent genres as tag vectors. First, all the rows corresponding to that genre is filtered. Then one of the two following ways is used:

5.3.1 Term Frequency

The term frequency of the tags of a given genre is defined as:

$$tf_{genre,t} = \frac{n_{genre,t}}{K_{genre}} \tag{4}$$

where,
$n_{genre,t} = sum(timestamps_{normalized})$ for tag t
$K_{genre} = sum(timestamps_{normalized})$ for all tags

The $tf_{genre,t}$ is calculated for each tag t. A tag vector consisting of all TF scores for each tag sorted in descending order of scores is printed.

5.3.2 Term Frequency-Inverse Document Frequency

The inverse document frequency of the genre is defined as:

$$idf_{genre,t} = log(\frac{N}{m_t}) \tag{5}$$

where,
N = Total number of genres from the entire database.
$m_t = sum(timestamps_{normalized})$ from genres containing tag t.

The score for each tag is then calculated by multiplying TF and IDF.

$$tfidf_{genre,t} = tf_{genre,t} * idf_{genre,t} \tag{6}$$

The $tfidf_{genre,t}$ is calculated for each tag. A tag vector consisting of all TFIDF scores $tfidf_{genre}$, sorted in descending order of scores is printed.

5.4 Task 3

The main objective of this task is to represent users as tag vectors. First, all the rows corresponding to that user is filtered.
Note: There are two files mlratings and mltags with user entries.
My assumption: A user has watched a movie if he has rated it or tagged it. So both mlratings table and mltags table userId,movieId are joined and TF and IDF scores are computed for all users as follows:

5.4.1 Term Frequency

The term frequency of the tags of a given genre is defined as:

$$tf_{genre,t} = \frac{n_{genre,t}}{K_{genre}} \tag{7}$$

where,
$n_{genre,t} = sum(timestamps_{normalized})$ for tag t
$K_{genre} = sum(timestamps_{normalized})$ for all tags

The $tf_{genre,t}$ is calculated for each tag t. A tag vector consisting of all TF scores for each tag sorted in descending order of scores is printed.

5.4.2 Term Frequency-Inverse Document Frequency

The inverse document frequency of the genre is defined as:

$$idf_{genre,t} = log(\frac{N}{m_t}) \tag{8}$$

where,
N = Total number of genres from the entire database.
$m_t = sum(timestamps_{normalized})$ from genres containing tag t.

The score for each tag is then calculated by multiplying TF and IDF.

$$tfidf_{genre,t} = tf_{genre,t} * idf_{genre,t} \tag{9}$$

The $tfidf_{genre,t}$ is calculated for each tag. A tag vector consisting of all TFIDF scores $tfidf_{genre}$, sorted in descending order of scores is printed.

5.5 Task 4

The main objective of this task is to produce a differentiating tag vector given two genres g_1 and g_2. First, all the rows corresponding to those two genres are filtered. The following three approaches are used to compute the differentiating tag vector:

5.5.1 TF-IDF-DIFF

The term frequency of the tags of a given genre is the same from Task 2. Equation (4) is used to compute the tag vectors for genre g_1 and genre g_2. The IDF score will be the same for both genres. It is not the same as Task 2 since we will only be considering g_1 and g_2. The IDF is calculated as follows:

$$idf_t = log(\frac{N}{m_t}) \tag{10}$$

where,
N = Total number of movies from $g_1 \cup g_2$.
$m_t = sum(timestamps_{normalized})$ from $g_1 \cup g_2$ containing tag t.
Then the $tfidf_g1$ and $tfidf_g2$ are calculated as follows:

$$tfidf_{g1,t} = tf_{g1,t} * idf_t \tag{11}$$

$$tfidf_{g2,t} = tf_{g2,t} * idf_t \tag{12}$$

The differentiating tag vector is obtained by subtracting(difference) of $tfidf_g1$ and $tfidf_g2$.

$$TFIDFDIFF_t = tfidf_{g1,t} - tfidf_{g2,t} \tag{13}$$

This prints a tag vector consisting of the TF-IDF-DIFF scores, sorted in descending order.

5.5.2 P-DIFF1

The objective of this task is to compute the $w_{1,j}$ tag vector which is used to discriminate between the given genres g_1 and g_2. The given formula is used with a minor change of adding 0.5 to the terms to handle edge cases is done. The final formula used to compute $w_{1,j}$ is as follows:

$$w_{1,j} = log(\frac{r_{1,j} + 0.5/R - r_{1,j} + 0.5}{m_{1,j} - r_{1,j} + 0.5/M - m_{1,j} - R + r_{1,j} + 0.5}) * |\frac{r_{1,j} + 0.5}{R+1} * \frac{m_{1,j} - r_{1,j} + 0.5}{M - R + 1}| \quad (14)$$

where
$r_{1,j}$ is the number of movies in genre,g_1 ,containing the tag t_j
$m_{1,j}$ is the number of movies in genre,g_1 or g_2 ,containing the tag t_j
$R = ||movies(g_2)||$
$M = ||movies(g_1) \cup movies(g_2)||$
The $w_{1,j}$ tag vector is printed after sorting in descending order.

5.5.3 P-DIFF2

The objective of this task is to compute the $w_{1,j}$ tag vector which is used to discriminate between the given genres g_1 and g_2. The given formula is used with a minor change of adding 0.5 to the terms to handle edge cases is done. The final formula used to compute $w_{1,j}$ is as follows:

$$w_{1,j} = log(\frac{r_{1,j} + 0.5/R - r_{1,j} + 0.5}{m_{1,j} - r_{1,j} + 0.5/M - m_{1,j} - R + r_{1,j} + 0.5}) * |\frac{r_{1,j} + 0.5}{R+1} * \frac{m_{1,j} - r_{1,j} + 0.5}{M - R + 1}| \quad (15)$$

where
$r_{1,j}$ is the number of movies in genre,g_2 ,not containing the tag t_j
$m_{1,j}$ is the number of movies in genre,g_1 or g_2 ,not containing the tag t_j
$R = ||movies(g_2)||$
$M = ||movies(g_1) \cup movies(g_2)||$
The $w_{1,j}$ tag vector is printed after sorting in descending order.

6 User Interface Specifications

6.1 Instructions to execute:

6.2 System Requirements

There are hardware prerequisites a system or PC needs to have, these are;
• Hard-disk space: 20 GB
• Minimum RAM: 4 GB
• Minimum Processor speed: 2.2 GHz

For software, any of the following can be used:
• Command prompt/Terminal

7 Related Work

8 References