CS6350
Big data Management Analytics and Management
Spring 2023
Homework 3
Submission Deadline: April 15th, 2023, 11:59pm

Part #1: TF-IDF

i) You are given a dummy database, containing 15 files, where each file contains the lyrics to a song. Your task is to first compute a Spark dataframe containing the TF-IDF of each token in the document following the format below. (Note: The values in the table are all dummy values and not necessarily accurate)

Song ID	Token	Term Frequency	Inverse Document Frequency	TF-IDF
Song 1	Token 1	1	0.477	0.477
Song 1	Token 2	1	0.123	0.123
Song 1	Token 3	2	0.477	0.954
Song 2	Token 1	1	0.123	0.123

The formula used to calculate TF-IDF is given below, where N represents the frequency of a word in a document. Note that you do not need to perform preprocessing steps such as lemmatization, stop word removal etc. Just **remove the punctuations** before you compute the tf-idf values.

$$\text{TF-IDF(w)} = N * \log\left(\frac{Total\ Number\ of\ Documents}{Number\ of\ Documents\ Containing\ Word\ w}\right)$$

You must implement the formula from scratch using pyspark or scala and cannot use any existing library functions.

ii) Once the table is computed, find the word with the highest tf-idf score for each song. You can use Spark SQL commands for this.

iii) Find the names of the top 3 songs that can be categorized as having a 'sad' mood. A song can be said to be 'sad' if it contains all or some of the following keywords: **"tear", "feel' and 'hate'**. This means, you will need to rank the songs based on how often these keywords are repeated in them. The formula to rank is as follows:

Rank_score(song, keywords) = TFIDF(song, "tear") + TFIDF(song, "feel") + TFIDF(song, "hate")

Show the top 3 songs using the ranking algorithm mentioned above. Also show the *Rank_score* for each those songs.

Part #2: Inverted Index:

Write a program that will construct inverted index in the following way: The **map** function parses each line in an input file, userdata.txt, and emits a sequence of <token, line number> pairs. The **reduce** function accepts all pairs for a given token, sorts the corresponding line numbers, and emits a <token, list(line numbers)> pair.

Once this is computed, filter the inverted index to keep only those tokens that are words (include names, leave out any numbers, dates or userids). Then, find all those words(s) whose number of occurrences is the maximum among all the words in the inverted index. Use Spark to compute your result.

Output Format:
<Word><TAB><Number of occurrences>

Part #3: Recommendation Systems

Use Collaborative filtering to find the accuracy of ALS model. Use ratings.dat file. It contains:

User id :: movie id :: ratings :: timestamp.

Your program should report the accuracy of the model. For details follow the link: https://spark.apache.org/docs/latest/mllib-collaborative-filtering.html. Please use 60% of the data for training and 40% for testing and report the MSE of the model.

What to submit:
Submit the code along with the output of your code for all the questions.