Extreme Computing
Assignment 2

1 Introduction

This is the second practical assignment for the Extreme Computing course 2023/24 (ungraded). You need to use Apache Spark to solve problems you might encounter when working with collections. This section will give you administrative information and help with solving the assignment. This is followed by the actual tasks.

1.1 Administrative Information

Deadline There is not deadline for this assignment :)

Questions All questions should go on Piazza

https://piazza.com/class/lmgiwvg3vlj5ec

Feel free to discuss general techniques amongst each other unless you would reveal an answer. If your question / discussion reveals an answer, ask privately.

Marking The assignment is not graded :)

2 Tasks

For the dataset you will use in this assignment, there is a `imdb-small-data.zip` file that can be accessed at:

https://amirsh.github.io/files/exc23/imdb-small-data.zip

You can copy the extracted `tsv` files to `src/main/resources/imdb/` for local testing and debugging.

2.1 Internet Movie Database (IMDB)

This assignment will also be on processing the IMDB dataset. However, this time you need to use Apache Spark to efficiently process structured text. Note that you are only allowed to use Spark RDDs (e.g., `map`, `groupBy`, `join`). This means that you are NOT allowed to use other facilities provided by the Spark ecosystem including Spark SQL.

You can refer to the first assignment for the description of the IMDB dataset.

3 Tasks

Download `imdb-spark-src.zip` and it extract it somewhere on your machine. You have to complete the missing implementations (specified by `???`) in `src/main/scala/imdb/ImdbAnalysis.scala`.

You are encouraged to look at the Apache Spark API documentation while solving this exercise:

`https://spark.apache.org/docs/3.0.3/api/scala/org/apache/spark/rdd/index.html`

Task 1

◁ **Task**

Calculate the average, minimum, and maximum runtime duration for all titles per movie genre.

Note that a title can have more than one genre, thus it should be considered for all of them. The results should be kept in minutes and titles with 0 runtime duration are valid and should be accounted for in your solution.

return type: RDD[(Float, Int, Int, String)]
avg_runtime:Float
min_runtime:Int
max_runtime:Int
genre:String

Task 2

◁ **Task**

Return the titles of the movies which were released between 1990 and 2018 (inclusive), have an average rating of 7.5 or more, and have received 500000 votes or more.

For the titles use the `primaryTitle` field and account only for entries whose `titleType` is `'movie'`.

return type: RDD[String]
title:String

Task 3

◁ **Task**

Return the top rated movie of each genre for each decade between 1900 and 1999.

For the titles use the `primaryTitle` field and account only for entries whose `titleType` is `'movie'`. For calculating the top rated movies use the `averageRating` field and for the release year use the `startYear` field.

The output should be sorted by decade and then by genre. For the movies with the same rating and of the same decade, print only the one with the title that comes first alphabetically. Each decade should be represented with a single digit, starting with 0 corresponding to 1900-1909.

return type: RDD[(Int, String, String)]
decade:Int
genre:String
title:String

Task 4

◁ **Task**

In this task we are interested in all the crew names (`primaryName`) for whom there are at least two known-for films released since the year 2010 up to and including the year 2021. You need to return the crew name and the number of such films.

return type: RDD[(String, Int)]
crew_name:String
film_count:Int