Extreme Computing
Assignment 1

1 Introduction

This is the first practical assignment for the Extreme Computing course 2023/24 (ungraded). You need to use the Scala Collection API 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 be on processing the IMDB dataset – we have chosen a subset for these tasks to encourage you to think about how to structure your solutions to use multiple input data collections, and efficiently process structured text using Scala collections.

Please note that we have removed the first line of the `tsv` file, which contained the column names in the original dataset. We have done this for your convenience, as in your code, you can assume all lines are data. The four files used, including their structures, are detailed in Section 2.2.

Note that not all `.tsv` files are required for all questions. Consult the schema in Section 2.2 to ascertain which one(s) you require for the task at hand. Be aware that `skipVal` (`'\N'`) may be present where fields are denoted `Option`, meaning no data is present. You are expected to account for this possibility and ignore those entries from your solutions.

2.2 IMDB Schema Reference

The following table defines the columns in each of the provided files from the IMDB dataset to aid you in your solution design.

- `Option[T]` means either type `T` is present, or `skipVal` (‘`\N`’) otherwise
- `List[T]` means a comma-delimited list of type `T` is present, e.g. ‘`dog,cat,bear`’, where `T := String`

INDEX	FIELD	TYPE	EXAMPLES/NOTES
			name.basics.tsv
0	`nconst`	`String`	`nmXXXXXXX` – Unique person/crew ID
1	`primaryName`	`Option[String]`	–
2	`birthYear`	`Option[Int]`	–
3	`deathYear`	`Option[Int]`	–
4	`primaryProfession`	`Option[List[String]]`	‘`editor,manager`’, ‘`actor`’, ‘`actress`’
5	`knownForTitles`	`Option[List[String]]`	‘`tconst1,tconst2,tconst3`’
			title.basics.tsv
0	`tconst`	`String`	`ttXXXXXXX` – Unique title ID
1	`titleType`	`Option[String]`	‘`tvMovie`’, ‘`short`’, ‘`movie`’, ‘`videoGame`’
2	`primaryTitle`	`Option[String]`	–
3	`originalTitle`	`Option[String]`	–
4	`isAdult`	`Int`	–
5	`startYear`	`Option[Int]`	`YYYY` – Release year
6	`endYear`	`Option[Int]`	`YYYY` – End year, e.g. when a play ends.
7	`runtimeMinutes`	`Option[Int]`	–
8	`genres`	`Option[List[String]]`	‘`Documentary,Short,Sport`’
			title.crew.tsv
0	`tconst`	`String`	Joins `title.basics.tconst`
1	`directors`	`Option[List[String]]`	‘`nmXXXXXX1,nmXXXXXX2`’ – Joins `nconst`
2	`writers`	`Option[List[String]]`	‘`nmXXXXXX1,nmXXXXXX2`’ – Joins `nconst`
			title.ratings.tsv
0	`tconst`	`String`	Joins `title.basics.tconst`
1	`averageRating`	`Float`	–
2	`numVotes`	`Int`	–

3 Tasks

Download `imdb-scala-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 Scala API documentation while solving this exercise, which can be found here:

`https://www.scala-lang.org/api/2.12.15/index.html`

Consult the schema in Section 2.2 when designing your solutions in order to extract the correct data.

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: List[(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: List[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: List[(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: List[(String, Int)]
crew_name:String
film_count:Int
```