2 Tasks

For each of the two datasets you will use in this assignment, there is a `large` repository to be used when deploying your solutions using Hadoop, and a `small` repository which you can copy to your local machine for local testing and debugging. All datasets can be found on the Teaching Hadoop Cluster in the `hdfs` file system under `/data`.

2.1 Text Analysis

We have taken a subset of Project Gutenberg's free text book repository for you to perform operations over. There is no structure between this data to be exploited, and these tasks require simple analysis of the properties of the text files as they are processed.

- `/data/small/gutenberg` – for development and local testing
- `/data/large/gutenberg` – for the final output
- `/data/samples/gutenberg` – for sample solutions for the `small` data

Here is an input example for the Gutenberg dataset:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Mauris fermentum, sapien quis consequat laoreet, risus
lorem laoreet nisi, non viverra elit metus id felis.
Pellentesque varius lorem eu nisl consectetur, id
rutrum dui suscipit.
```

Task 1

◁ **Task**

Implement a simpler version of `wc` in Hadoop streaming that calculates only the number of words and lines. Run your implementation on the Gutenberg corpus. Note that the native `wc` may have a different definition of a word to ours. **(1+3 marks)**

> num_words:int num_lines:int

Output example:

```
34 5
```

Task 2

◁ **Task**

Compute bi-grams on the Gutenberg dataset. Return an unordered list of all bi-grams whose frequency is greater than 5. Do not worry about non-alphabetical characters, and case-sensitivity (i.e. "The" and "the" are separate words). **(2+4 marks)**

> [(bigram:str, frequency:int)]

Output example:

```
Lorem_ipsum 1
ipsum_dolor 1
...
```

2.2 Internet Movie Database (IMDB)

The bulk of 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 streams, and efficiently process structured text using Hadoop Streaming.

Please note that we have removed the first line of each `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 2.3.

- `/data/small/imdb/*.tsv` – for development and local testing
- `/data/large/imdb/*.tsv` – for the final results
- `/data/samples/imdb` – sample results on the `small` data

Note that not all `.tsv` files are required for all questions. Consult the schema in Section 2.3 to ascertain which one(s) you require for the task at hand. Be aware that `skipVal` (`'\N'`) may be present where fields are denoted `Optional`, meaning no data is present. You are expected to account for this possibility.

2.2.1 Counting

Sometimes we have to be able to compute the cardinality of some group of data which may or may not fit in memory. These tasks look to solve this problem for finding out statistics about groups of interest in the IMDB data.

Task 3

◁ **Task**

People in the dataset can have many different professions, but we are only interested in finding out how many `actors` there are. Identify only those people whose primary job is of an `actor` or `actress` and count how many you encounter. **(2+2 marks)**

num_actors:int

Task 4

◁ **Task**

Titles are grouped by the `genres` into which they fit, and `genres` may be shared between many different `titleTypes`. List all genres in the IMDB dataset, regardless of `titleTypes`, sorted in alphabetical-order. `genres` should only appear once in your output. **(2+2 marks)**

[genre:str]

Task 5

◁ **Task**

Find the earliest and latest title release year for all titles of any `titleType`, and return them in that order. **(2+2 marks)**

earliest_year:int latest_year:int

2.2.2 Averaging

Task 6

◁ Task

For all the titles that have ratings, find the average number of votes a title has received, rounded to 2 decimal places. A rating of 0 is a valid rating. **(2+2 marks)**

```
avg_vote:float
```

Task 7

◁ Task

Find the average number of writers for all titles that have at least one writer, rounded to the nearest integer. **(2+2 marks)**

```
avg_num_writers:int
```

Task 8

◁ Task

Find the average rating achieved for movies released in each decade for every decade between `1/1/1900` and `31/12/1999`. As this is a simple average, you may ignore the number of votes for each title. This task may require multiple Map-Reduce-Jobs. For full marks you should not need more than two. **(4+4 marks)**

```
[(decade:int, avg_rating:float)]
```

2.2.3 Exploring Relationships

These questions require you to think about joining data from more than one file based on shared attributes (keys) between them. You should consult the schema in Section 2.3 when designing your solutions in order to extract the correct data.

Task 9

◁ Task

For every genre in the dataset, count the number of all titles released between `01/01/2000` and `31/12/2014` and return the values grouped by genre. Each `<genre, num_releases>` pair should appear only once in your output. The ordering of the output is not important. **(2+3 marks)**

```
[(genre:str, num_releases:int)]
```

Task 10

◁ Task

Give all the crew IDs (`nconst`) that are `knownFor` films released since the year 2010 up to and including the current year (2018). Some crew will be known for more than one title, therefore duplicate `nconst` values are both expected and accepted. **(3+4 marks)**

```
[crew_id:str]
```

2.3 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.

- `Optional[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 := str`

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