Extreme Computing
Assignment

1 Introduction

This is the coursework assignment for the Extreme Computing course 2018/19. You need use the Hadoop Streaming API to solve problems you might encounter when working in a big data context. This section will give you administrative information and help with solving the assignment. This is followed by the actual tasks and finally a description of how to submit your solutions.

1.1 Administrative Information

Deadline The assignment is due at **16:00 on 29 October**.

Deadline Extension The School of Informatics has a policy on coursework deadlines, which applies across all taught courses. Further information can be found here:

http://web.inf.ed.ac.uk/infweb/student-services/ito/admin/coursework-projects/late-coursework-extension-requests

Questions All questions should go on Piazza

https://piazza.com/ed.ac.uk/fall2018/infr11088

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

Assignment office hours We will provide office hours in week 2 and 4 after assignment release to answer questions in person regarding the assignment. Please look at the course's LEARN page for times and locations.

Marking The assignment is worth 50 marks in total and counts for 40% of the final course mark. Marks are given for correctness, efficiency and proper use of tools. This includes running your solution in a scalable way.

For purposes of correctness marks, we will normalise your output by concatenating files, stripping leading and trailing whitespace, and sorting. That means you can have a different partitioner, different sorting comparison function, and different number of reducers while getting correctness marks. However, if a question requires a certain order of elements as output or has only singular data output, you should use a single reducer in your last MapReduce job, i.a. a single output file. The same is true if the task specifically asks for such behaviour. We will also expect your code to behave correctly on other inputs.

Marks are indicated on the right margin of the page by two numbers, e.g. **1+3 marks**. The first number indicates achievable marks for correctness while the second number indicates achievable marks for efficiency.

Submission The submission process for your solution is described in Section 3. We start marking at the deadline and only mark once. If you are submitting on time, you can submit as many times as you want and the last one will be marked. If you are submitting late, you may only submit once in total (which implies that you should not submit before the deadline) or run the risk that we will mark an old version then penalise for the last submission time.

Marking Feedback You will receive your marks and feedback for your solution to your student email address within three weeks following your submission. Once you have received your marks, you have three days to ask questions about them, e.g. feedback clarification. Please post any such questions in a private Piazza message.

Good Scholarly Practice Please remember the University requirement as regards all assessed work for credit. Details about this can be found at:

http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct

Furthermore, you are required to take reasonable measures to protect your assessed work from unauthorised access. For example, if you put any such work on a public repository then you must set access permissions appropriately (generally permitting access only to yourself, or your group in the case of group practicals).

1.2 Teaching Hadoop Cluster

You should use the teaching Hadoop Cluster[1] and the Hadoop Streaming API with **Python 2.7** code for Mappers, Reducers and Combiners for all of your solutions. You will have local filespace (under your own credentials) in HDFS at `/user/UUN`

If you are logging in from outside Informatics, first

```
ssh s12345678@student.ssh.inf.ed.ac.uk
```

or use the Informatics VPN:

http://computing.help.inf.ed.ac.uk/openvpn

Once inside Informatics, connect to cluster's resource manager node `scutter02` to submit your jobs:

```
ssh hadoop.exc
```

To manage cluster jobs use the `mapred` command or view the web interface of the job tracker at:

http://hadoop.exc.inf.ed.ac.uk:8088/cluster

1.3 Official Resources and Supplementary Material

You are expected to find your own way to solve the tasks. You can use the internet to find resources which might help you, e.g. Hadoop tutorials or documentation. We will, however, provide supplementary material tailored to using the teaching Hadoop Cluster to get you started. You can find it on the course's `Learn` page under `Coursework and feedback` → `Supplementary Material`

[1] You can run your own, but we won't support it and you need to copy output back to the teaching one.

1.4 General Hints

Task Runtime All of the solutions take less than an hour and less than 30 total workers (mappers + reducers). You should kill any jobs that take longer:

```
mapred job -kill $jobid
```

We may use a bot to kill non-conforming jobs.

Complexity If not otherwise stated, we generally expect O(1) space complexity for full marks in mappers, combiners and reducers.

Map-Reduce-Jobs If not otherwise stated, we expect a single Map-Reduce-Job for full marks. If multiple jobs are required, results from the first job can be used as input for a following job.

Expected Outputs Each question defines the expected output format by specifying expected fields and data types and whether singular or multiple outputs are expected. For multiple outputs, the types are enclosed in []. The following example expects multiple lines of two fields with integer and string value.

> [value1:int value2:str]

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	–

3 Submissions

To submit your work, please do the following:

1. Make sure that you store the output of your MAPREDUCE jobs in HDFS. Please store the output for the X^{th} task in the folder: `/user/sXXXXXXX/assignment/taskX` (replace `sXXXXXXX` with your student number). This way we can easily check whether you got the right output. When marking, we will only be interested in the output for the **large** version inputs. Do not put the output for the small version inputs in the same folder as it can be mistaken for your output for the large version. **Do not delete these files from HDFS until you are told it is OK to do so.**

2. To submit your code, please prepare a directory for submission that will contain one directory for each task and name the directory for the X^{th} task `taskX`. Inside each task folder, please include:

   - One file with `.sh` extension. This should be the shell script that executes your MAPREDUCE job(s) for X^{th} task.
   - One file with `.out` extension. This file should contain the first 20 lines of your output. If Hadoop splits your output into multiple files, please use the first twenty lines from the first output file (usually called `part-00000`). You can use the command:

     ```
     hdfs dfs -cat filename | head -20
     ```

     for showing just the first 20 lines of the file. If the whole output has less then 20 lines together, then your `*.out` file should contain the whole output.
   - Files with your code. This includes all code that you wrote to solve the task. **Do not submit a Word document or a PDF file—only submit a plain text file**. It is not required, but it will make marking easier if the file name with mapper, combiner, reducer code starts with `mapper`, `combiner` and `reducer` respectively. You can see an example below:

     ```
     submissionFolder/
     -- task1/
        -- run.sh
        -- output.out
        -- mapper.py
        -- mapper2.py
        -- reducer.py
        -- reducer2.py
        -- combiner.py
        -- combiner2.py
     -- task2/
        -- run.sh
        -- output.out
     ...
     ```

     This is an artificial example, it does not mean that you need two MAPREDUCE jobs with combiners to solve the first task.

   Once you have prepared your submission directory with the specified structure, please run

   ```
   /afs/inf.ed.ac.uk/group/teaching/exc/submit-assignment.sh path/to/submission_folder
   ```

   This will give you friendly warnings to help you spot if a folder (e.g. `task8`) or a file is missing. You will be asked whether you want to submit your submission directory (regardless of whether any warnings were given).

   NOTE: If you submit using `ssh` to access to the university network, make sure you are on `student.compute`. The gateway server does not have all the required tools for the script to work.