NUS
National University of Singapore
School of Computing

CS4225/CS5425
Big Data Systems for Data Science

Assignment 1: Introduction and Hadoop

Outline

- Description of Tasks 0&1
- Submission requirements

Task Overview

- Motivation
  - Text and documents are big data.
  - Text and document processing is fundamental for many Web applications.

- The assignment consists of an example program (Task 0: to help you test your setup and give you an example of a working program, no need to submit it) and the actual task to submit (Task 1):
  - Task 0: A simple word count program for testing your setup.
  - Task 1: Given two textual files, output the top k words that are common and have length greater than 4.

Task 0

- This is a simple test program to test your setup and get accustomed to logging in and working with the cluster
- The code is already given to you and will not be graded. You can use it as an example of what a working MapReduce program looks like.
- To submit this, follow the instructions in the Student Guide sections 1-2. If the test passes and you can successfully submit, it means everything has been set up correctly.
  - Run: sbatch slurm_run.sh
  - Submit: ./submit

Task 1

- Motivation
  - We choose **CommonWords** as our task because it is representative, and it is based on **WordCount**.

Task 1

- Problem definition
  - Given TWO textual files, find common words between the two files that have length greater than 4 and take the *smaller* number of times it appears between the two files. Output the top k words in sorted order of occurrence frequency. For words of the same frequency, sort them by ascending lexicographic order.
  - Example: if the word "Fresh" appears 5 times in the 1st file and 3 times in the 2nd file, the smaller number of times is 3.
  - Example: if the answer is "29 Holiday, 18 Happy, 18 Snacks, 10 Pineapple-tart" the output must be in that sorted order. i.e "29 Holiday, 18 Snacks, 18 Happy, 10 Pineapple-tart" will **not** be accepted
- Requirements
  - Split the input text with "(space)\t\n\r\f". Any other tokens like ",.:`" will be regarded as a part of the words
  - Remove stop-words as given in Stopwords.txt, such as "a", "the", "that", "of", … (case sensitive)
  - Sort the common words in descending order of the smaller number of occurrences in the two files. Then sort the words with common frequency in lexicographic order and output the top k words.
  - For the public test case, use k = 10 when testing your code.
  - In general, words with different case or different non-whitespace punctuation are considered different words

A Running Example

- Stopwords
  - as, had
- File 1
  - he put some sugar into his coffee, ~~as~~ he likes sugar filled coffee. Please!
- File 2
  - he ~~had~~ sugar in his coffee, though he is diabetic and he suffer. Please!
- k = 2
- Output:

```
1          Please!
1          coffee,
```

Frequencies of common words

Common words sorted in lexicographical order

A Running Example

- Stopwords
  - as, had

File 1 :

```
{'he': 2,'put': 1,
 'some': 1,'sugar': 2,
 'into': 1,'his': 1,
 'coffee,': 1,'likes': 1,
 'filled': 1,'coffee.': 1,
 'Please!': 1}
```

File 2 :

```
{'he': 3,'sugar': 1,
 'in': 1,'his': 1,
 'coffee,': 1,'though': 1,
 'is': 1,'diabetic': 1,
 'and': 1, 'suffer.': 1,
 'Please!': 1}
```

- Output:

```
1          Please!
1          coffee,
```

Frequencies of common words

Common words sorted in lexicographical order

A Running Example

- Stopwords
  - as, had

File 1 :

```
{'~~he~~': 2,'put': 1,
 'some': 1,'sugar': 2,
 'into': 1,'~~his~~': 1,
 'coffee,': 1,'likes': 1,
 'filled': 1,'coffee.': 1
 'Please!': 1}
```

File 2 :

```
{'~~he~~': 3,'sugar': 1,
 'in': 1,'~~his~~': 1,
 'coffee,': 1,'though': 1,
 'is': 1,'diabetic': 1,
 'and': 1, 'suffer.': 1
 'Please!': 1}
```

- Output:

```
1          Please!
1          coffee,
```

Frequencies of common words

Common words sorted in lexicographical order

Task 1

- Your java class should take in 5 inputs:
  - *TopkCommonWords <File_1> <File_2> <Stopwords> <Output_Dir> <value_of_k>*
- Inputs:
  - Two source files:
    - File_1: Task1-input1.txt
    - File_2: Task1-input2.txt
  - The stop-word file:
    - Stopwords: Stopwords.txt
  - Output_dir:
    - output/
  - Value of k:
    - Value_of_k: For the public testcase, use k = 10
- Output
  - Top-k output of the result using the data files listed above (you only need to extract the k outputs with the highest frequency), with each line:
    - **Freq**\t**word**\n, where \t is tab and \n creates a new line.

Task 1

- As SoC cluster has migrated to use Slurm, users are no longer able to directly log into each machine to run their jobs.
- To submit a job request on the SoC cluster, you can run “sbatch slurm_run.sh”. The script will submit a job request and automatically upload the files to HDFS and set HDFS path as parameters.
- If you are running the code on a local setup, an example command can be:
  - hadoop jar cm.jar TopkCommonWords data/input/task1-input1.txt data/input/task1-input2.txt data/input/stopwords.txt ./output/ 10

Submission Requirements

- Deadline: 27 Feb 2359hrs
- Task 0: You can optionally run ./submit on the SoC cluster to test your setup, but it will not be graded
- Task 1: Run ./submit on the SoC cluster to submit your codes
  - When submitting, make sure you input your matriculation number (i.e. starting with capital A), not your email address
  - Submit your code onto Canvas
  - Note: We will only mark based on the code submitted onto the cluster. The code on Canvas only acts as a backup for any unforeseen circumstances.
  - Submit a video **no longer than 5 minutes** to explain your code, onto Canvas
  - Make sure the video captures both your face and your code throughout the full duration

Marking

- The assignment contains a public dataset 'data/' and expected output 'answer.txt'. If your codes are correct, your output should be the same as 'answer.txt'. Different orders will be considered as wrong in marking.
- All the codes will be automatically compiled and marked by similar scripts as 'slurm_run.sh' on a private test dataset
  - The private test dataset is very similar to the test data given to you. If your program gives the correct output on the given test data, it is very likely to give the correct output on the private test data as well.
- So, ensure your codes can be compiled by the script in your package.
- All submitted codes will be auto-checked for plagiarism. Do NOT copy others' codes or share your codes with others.

Marking Schemes

- Total: 25% of total grade.
  - Code: 17/25
  - Video presentation: 8/25

Notice

- We have zero-tolerance on plagiarism.
- Do not “Copy and paste” from others.
- Do not share your code with others.

Questions regarding the Assignment

- You can post your questions in the Canvas discussion (preferred).
- Or, Email the tutor in charge of your group.

The End