CS542000 Cloud Programming
HW2: Inverted Index

Josh Kao

NTHU LSA Lab

2015/4/13

Outline I

1. Problem Description

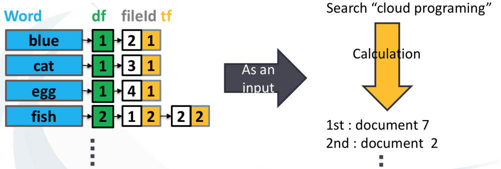
2. Input/Output Formats

3. Grading

4. Reminder

Overview

- You have to write a ranked-based search engine, including
  - Part 1 : inverted index
  - Part 2 : retirval
- Your inverted index table should include **term frequency**(tf) and **document** frequency(df) of each word. Thus, you can search by this table in part 2.



Word df fileId tf

blue → 1 → 2 1

cat → 1 → 3 1

egg → 1 → 4 1

fish → 2 → 1 2 → 2 2

⋮

As an input

Search "cloud programing"

Calculation

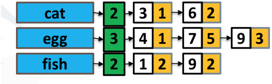
1st : document 7

2nd : document 2

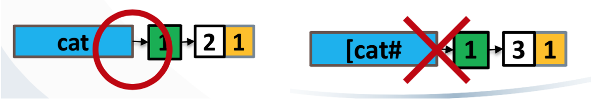
⋮

Requirement

- part1 - Inverted Index
  Write mapreduce code to output inverted index table
  1. Your table should include **document frequency** and **term frequency** for each word



  2. File name should be sorted.
  3. Words in your table should not contain useless notation



(a) Correct (b) Wrong

Requirement

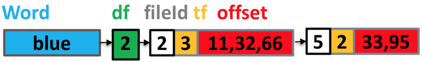
- part2 - Retrival HDFS API
  Use **MapReduce API** to seach words based on your inverted index table, and output their rank
  1. Use **TF.IDF Term Weighting** to rank words
     $$w_{i,j} = tf_{i,j} * log\frac{N}{df_i}$$
  2. Be able to retrival **multiple** with key words for each query
  3. Output the 10 highest files OR operation
  4. You should not fix #files. (Demo with another testcase)

Requirement

- Extend to full inverted index

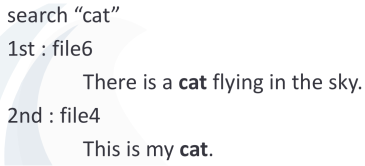
need to sort offset

1. Add field **offset** for each file



2. Output some fragments of file which contain at least one of key words

前後各一個字



search “cat”

1st : file6

There is a **cat** flying in the sky.

2nd : file4

This is my **cat**.

Requirement

- Implement **one** advanced function
  - Retrival can support "AND/NOT"
  - Retrival can support "Ignore uppercase or lowercase"
  - Any other interesting extentsion you can think of!

Requirement

- Report
  - **Instruction** : how to compile and execute your program
  - **Design** : explain your algorithm
  - **Questions** : choose two of them to answer
    1. How many #phases you used to run mapreduce in part1?
       Is there any other way to do it?
       What's the pros and cons?
    2. What's your extension?
       What's the most difficult part in your implementation?
    3. How do you filter those useless notation?
       If we need to search these special notations, how to modify your filter?

Outline I

1. Problem Description
2. Input/Output Formats
3. Grading
4. Reminder

Input

- Input files are Shakespeare's book splitting into 44 files
- Input files are at shared/HW2/input

Output

- **Inverted Index Table** (We would checkout content in the table)

  Word df file1 tf1 [offset1,offset2,$\cdots$];file2 tf2$\cdots$

- **Retrival**

  ```
  {RANK}      {FILE1} score = {SCORE}
  ************************

  {FILE_FRAGMENT1}
  {FILE_FRAGMENT2}
  ************************
  ```

Outline I

1 Problem Description

2 Input/Output Formats

3 Grading

4 Reminder

Grading

1. **[45%]** Inverted index
2. **[20%]** Retrival
3. **[10%]** Extend to full inverted index
4. **[5%]** Implement one extension
5. **[20%]** Report + Demo

Outline I

1. Problem Description

2. Input/Output Formats

3. Grading

4. Reminder

Reminder

- Upload **HW2_{Student-ID}.zip** to iLMS before 5/18(Mon) 23:59:59
  1. HW2_{Student-ID}_code.tar.gz
  2. HW2_{Student-ID}_report.pdf
- Please **start your work ASAP** and do not leave it until the last day!
- Late submission penalty policy please refer to syllabus.
- Asking questions on iLMS or through e-mail is also welcome!

Hint

- Get input file name
  - In mapper, use **Reporter** and **FileSplit** class