CSE427S

HOMEWORK 5

M. Neumann

Due: THU 3 OCT 2019 4PM

Getting Started

Update your SVN repository.

When needed, you will find additional materials for *homework x* in the folder `hwx`. So, for the current assignment the folder is `hw5`.

SUBMISSION INSTRUCTIONS

WRITTEN:

- all written work needs to be submitted electronically in *pdf format*[1] via GRADESCOPE
- provide the following information on *every* page of your `pdf` file:
  - name
  - student ID
- start every problem on a *new page*
- **FOR GROUPS**: make a **group submission** on GRADESCOPE and provide names and student IDs for **all group members** on every page of your `pdf` file.

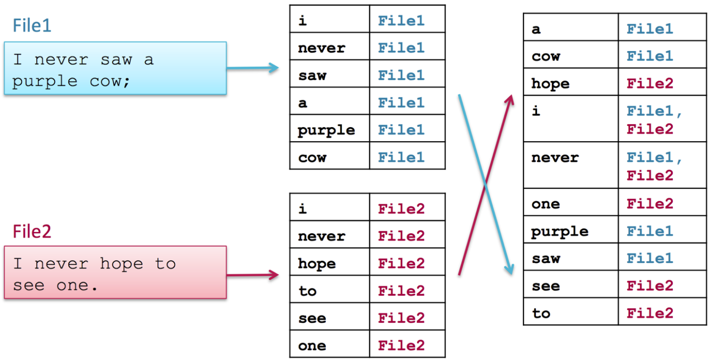
CODE:

- code needs to be submitted electronically in a *single .zip file* via GRADESCOPE (detailed submission instructions are provided under HOMEWORK 5 CODE below)
- make sure to always use the required *file name(s)* and *submission format(s)*
- **comment your code** to receive maximum credit

[1] Please, **type your solutions** or use **clear hand-writing**. If we cannot read your answer, we cannot give you credit nor will we be able to meet any regrade requests concerning your writing.

Problem 1: Creating an Inverted Index (20%)

An *index* is a list of contents (e.g., words) per file or document, i.e. it maps files or documents to its content. An *inverted index* maps contents (e.g., words) to files or documents. See also [MMDS]: Ch1.3.3 Indexes. Illustration:



i	File1
never	File1
saw	File1
a	File1
purple	File1
cow	File1

i	File2
never	File2
hope	File2
to	File2
see	File2
one	File2

a	File1
cow	File1
hope	File2
i	File1, File2
never	File1, File2
one	File2
purple	File1
saw	File1
see	File2
to	File2

Write a MAPREDUCE job that produces an inverted index. Use the stubs provided in:

```
~/workspace/inverted_index/src/stubs/
```

Use the input provided in the file `invertedIndexInput.tgz` located in

```
~/training_materials/developer/data/invertedIndexInput.tgz
```

Preparation: Extract the `invertedIndexInput` directory and upload it to HDFS .
When decompressed, this archive contains a directory of files; each is a Shakespeare play formatted as follows:

```
0 HAMLET
1
2
3 DRAMATIS PERSONAE
4
5
6 CLAUDIUS king of Denmark. (KING CLAUDIUS:)
7
8 HAMLET son to the late, and nephew to the present king.
...
```

Each line contains:

- *Line number*
- *separator*: a tab character
- *value*: the line of text

IMPLEMENTATION SPECIFICATIONS:

- Your program should transform all words to lowercase.
- Programs to implement and submit (cf. HOMEWORK `hw5`CODE below): `IndexMapper.java`, `IndexReducer.java`, `InvertedIndex.java`

(a) Before implementing the job, think about the following **design questions**. Provide the answers in your written submission.

- Which `InputFormat` can be used to read this data directly as key value pairs?
- How can you retrieve the file name in the Mapper? Provide the respective code in your write-up.
  HINT: Consult Table 8-7. *File split properties* in HTDG and note that the `getPath()` method has a `getName()` function.
- In which method of your Mapper will you retrieve the filename? Why?

(b) Our goal is to implement an indexer that produces an index of **all** the words in the input data. For each word, the index should have a list of all the locations where the word appears in the following format: *filename @ line*.

For instance for the word *honeysuckle* the final output should look like this:

```
honeysuckle 2kinghenryiv@1038,midsummernightsdream@2175
```

What would be the **Mapper output** for the following test input lines from `Hamlet`:

```
282 have heaven and earth
133 there are more things in heaven and earth
```

Problem 2: Utility Matrix - Similarity Measures (30%)

Consider the following utility matrix, representing the ratings, on a 1-5 star scale, of eight items, a through h, by three users A, B, and C.

	a	b	c	d	e	f	g	h
A	4	5		5	1		3	2
B		3	4	3	1	2	1	
C	2		1	3		4	5	3

Compute the following from the data of this matrix. This is essentially Exercise 9.3.1 in **MMDS**.

(a) Treating the utility matrix as boolean, compute the Jaccard similarity between each pair of users.

(b) Repeat Part (a), but use the cosine similarity.

(c) Treat ratings of 3, 4, and 5 as 1 and 1, 2, and blank as 0. Compute the Jaccard similarity between each pair of users.

(d) Normalize the matrix by subtracting from each non-blank entry the average value for its user. Using this normalized matrix, compute the cosine distance between each pair of users.

(e) Which of the four measures best reflects your intuition about the similarity of the three users? Provide a justification.

Problem 3: Collaborative Filtering - Similarity Measures (30%)

(a) Show formally that the **normalized cosine similarity** measure corresponds to the **Pearson correlation**.

(b) **Quality vs. implementation effort and efficiency**

- From a quality perspective, what is the benefit of using the normalization (i.e., Pearson correlation instead of cosine similarity)?
- From an implementation and data storage perspective, what is the disadvantage of using the normalization (i.e., Pearson correlation instead of cosine similarity)?

(c) **Jaccard similarity**

- What is the main advantage of using the Jaccard similarity for collaborative filtering?
- What is the main disadvantage of the Jaccard similarity measure?
- Can you think of a way to pre-process the rating data to overcome this problem?

continue to next page...

CSE427S

HOMEWORK 5 CODE

M. Neumann

Due: THU 3 OCT 2019 4PM

SUBMISSION INSTRUCTIONS

CODE:

- create a **single zip file** named `hw5_YourName.zip` (or `hw5_YourName_YouPartnersName.zip` for groups) including all files listed below

  Example file names:

  `hw5_MarionNeumann.zip`
  or for teams:
  `hw5_MarionNeumann_AnnaLee.zip`
  or
  `hw5_AnnaLee_MarionNeumann.zip`

- submit the *zip file* to the `hw5` CODE assignment in GRADESCOPE
- your code will be **autograded** for *correctness* and **manually inspected** for *comments*

Problem 4: Creating an Inverted Index (20%)

(a) Submit your `IndexMapper.java` program. Remove the provided comments and add your *own* comments to your code to receive maximum credit.

(b) Submit your `IndexReducer.java` program. Comment your code (i.e., add you *own* descriptive inline comments) to receive maximum credit.

(c) Submit your `InvertedIndex.java` program. Comment your code (i.e., add you *own* descriptive inline comments documenting all changes) to receive maximum credit.

Files to be added to your `zip` file:

```
IndexMapper.java
IndexReducer.java
InvertedIndex.java
```

Reflection (Bonus Problem for 5% up to a max. of 100%)

Reflect on your homework experience! Write a paragraph of at least 50 words to express your experiences and feelings when working on this assignment. Answer at least 2 of the following questions:

- What did you like/dislike about the assignment and why?
- What is the most important thing you learned and why do you think so?
- What surprised you, and why?
- Assuming you could start over again (with working on the assignment), what would you do differently and why?

Do not include/copy and past the questions into your reflection!

Submission Instructions

Store your reflection in the `hw5_reflection.txt` file provided in the `hw5`folder in your SVN repository and commit it.

This file should only include the reflection, **no other personal information** such as name, wustlkey, etc. reflections are not graded based on the content, but solely for completion.

To submit your reflection `cd` into the `hw5` folder and run:

```
$ svn commit -m 'hw5 reflection submission' .
```

Take 1 minute to provide an overall **star rating** for this homework.
Submit it via this link: `https://wustl.az1.qualtrics.com/jfe/form/SV_bIRKP1xobpO8yIB`.

Grading - no group work!

You can only earn bonus points if you write a *meaningful* **reflection** of **at least 50 words** answering at least 2 of the prompted questions and provide the corresponding **star rating**. You will **not** be graded on what your reflection says and the number of stars you assign, but rather solely the completion of it.
Bonus points are given to the **owner of the repository only**. No group work!.