Lab 5: Simple Hadoop Programs

Due date: February 13, 11:59pm.

Lab Assignment

Assignment Preparation

This is an individual lab. I expect every person to complete it without consulting others.

0.1 Overview

In this lab you will create a number of Hadoop MapReduce programs that perform a variety of simple tasks on simple data inputs. All programs you will write share the same features:

- The inputs are files with simple, easy-to-understand structure, with each record stored in a single line.
- The `map()` and `reduce()` methods do not need to use any functionality beyond core Java libraries.
- The MapReduce job you are asked to implement is an example of a specific type of a data-processing operation discussed in class.

Program 1: Filtering

Create a MapReduce job named `repeatLetters.java` that works as follows.

Input. The input to the MapReduce process is a file containing a list of words, one word per line.

Processing. Your MapReduce job shall output a list of words that have double letters in them. For example, words like `"hello"`, `"book"`, `"barrow"` shall be emitted, while words like `"table"`, `"break"`, `analysis` shall be filtered out. The MapReduce job shall be case-insensitive, so, `"Llan"`, for example, shall be printed out.

Output. The reduce() method shall emit the word as the key, and the letter that is doubled as the value. For example, for `"Hello"`, the output shall be

```
Hello l
```

(the space between the key and the value is flexible - it can be a tab)

For words with multiple double letters, e.g., `"toothiness"`, report one of the letters (either `"o"` or `"s"`).

Program 2: Transformation

Create a MapReduce job named `invert.java` that works as follows.

Input. The input to the MapReduce process is a file containing a list of words, one word per line.

Processing. Your MapReduce job shall output the list of word pairs: the first word of the pair (the key) is the original word with its characters in reverse order; the second word - the original word spelled in CamelCase repeated twice. For example if the input word is `"stream"`, the program shall produce as output the following pair:

```
maerts StreamStream
```

`Output.` The `reduce()` method shall output the the inverted word as the key, and the original word in CamelCase. All occurrences of the same word shall be reported together (note: if you write your `reduce()` method correctly, you will get this one for free, just make sure that there is no duplicate removal for this program, i.e., if your input file contains two entries of the word `"stream"` your output will contain the line above twice in a row).

Program 3: Grouping and Aggregation

Create a MapReduce program named `countySales.java` that works as follows.

Input. The input to the program is a file named `iowa.csv` located in `/data` directory on HDFS. You all should have read access to this file.

The file represents a CSV version of some of the data from our Iowa liquor sales database. Each line in the file is a single record in the format

```
<Invoice>, <Store Number>, <Store County>, <Vendor Name>, <Item Number>, <Item Description>, <Bottles Sold>, <Sale
```

Here are a few sample lines from the input file:

```
'S24966600138',2614,'Scott','Brown-Forman Corporation',86817,'Southern Comfort Cherry',2,29.56
'S24043700013',2641,'Pottawattamie','Mccormick Distilling Company',36903,'Mccormick Vodka',96,163.2
'S22293800009',4925,'Polk','Sazerac Co., Inc.',64866,'Fireball Cinnamon Whiskey',12,161.64
'S15881700017',3976,'Iowa','Luxco-St Louis',81208,'Paramount Peppermint Schnapps',2,21.24
```

Processing. The MapReduce job shall compute, for each county, (a) how many total bottles were purchased, (b) the total amount of money the purchased alcohol was worth, and (c) the average price per bottle.

Output. The `reduce()` method shall output the county name as the key, and the triple $\langle$ number of purchased bottles, total amount of money spent on purchasing alcohol, average price per bottle $\rangle$ as the value.

Program 4: Inverting Construct a MapReduce program `invertedIndex.java` that works as follows.

Input. The input to the program is the `iowa.csv` file introduced in the description of **Program 3**.

Processing. For this task, you are interested in the `Item Description` column of the input CSV file (sixth column in the file). Your program shall split each `Item Description` into individual words (e.g., 'Southern Comfort Cherry' is split into `'Southern'`, `'Comfort'` and `'Cherry'` words). For each word found in the entire input (all `Item Description` values), your program shall compute and output the following information:

- Number of unique occurrences of the word
- Number of unique counties that occur in the same record as the word
- Number of unique `Item Description` values in which the word occurs

Output. In the output, use the word as the key, and output a printable object that consists of the three values specified above.

Submission

Use `handin` for submission. Submit the four Java programs (you should not need any additional java files for these assignments), and a `README` file with your name and any comments you need to make about your code.

Use the following command for the submission:

```
$ handin dekhtyar lab05 <files>
```