**HU Extension Assignment 04 E63 Big Data Analytics**

Issued on: February 20, 2016 Due by 11:30PM EST, February 26, 2016

**Problem 1.** Write a working WordCount program using Spark Java API that reads a file, e.g. Ulysis/4300.txt from an HDFS directory and writes the results of your calculations to an HDFS file. To improve your word count, remove any punctuation that might have attached itself to your words. Also transform all words into lower case so that the capitalization does not affect the word count. The original code used in lecture notes is provided in the attached mini-example-java.tar file. That archive also contains Maven’s pom.xml file. Run your program and demonstrate that it works. Submit working code inside the customary MS Word Document. Describe steps in your program.

This is the my version of WordCount program that uses Spark JAVA API:

package e63.course.assignment4;

import java.util.Arrays;

import org.apache.spark.SparkConf;

import org.apache.spark.api.java.JavaPairRDD;

import org.apache.spark.api.java.JavaRDD;

import org.apache.spark.api.java.JavaSparkContext;

import org.apache.spark.api.java.function.FlatMapFunction;

import org.apache.spark.api.java.function.Function2;

import org.apache.spark.api.java.function.PairFunction;

import scala.Tuple2;

public class WordCount{

public static void main(String[] args) throws Exception {

// set input and output files/dirs

String inputFileName = args[0];

String outputDirName = args[1];

// Create a Java Spark Configuration

SparkConf sparkConf = new SparkConf().setAppName("Assignment4\_Problem1");

// Create a Java Spark Context

JavaSparkContext sparkContext = new JavaSparkContext(sparkConf);

// Load the input data.

JavaRDD<String> inputFileContents = sparkContext.textFile(inputFileName);

// Split the input data into words

JavaRDD<String> wordsListRDD = inputFileContents.flatMap(new FlatMapFunction<String, String>() {

private static final long serialVersionUID = 1L;

public Iterable<String> call(String line) {

return Arrays.asList(line.split(" "));

}

});

// Transform into a map of word and default count of 1

JavaPairRDD<String, Integer> wordsListWithDefaultCount = wordsListRDD

.mapToPair(new PairFunction<String, String, Integer>() {

private static final long serialVersionUID = 1L;

public Tuple2<String, Integer> call(String word) {

**// the below is done to eliminate punctuations and to**

**// convert the words into lower case**

word = word.replaceAll("[^A-Za-z0-9]", "").toLowerCase();

return new Tuple2<String, Integer>(word, 1);

}

});

// Filter the list to eliminate blanks

JavaPairRDD<String, Integer> filteredWordsList = wordsListWithDefaultCount

.filter((Tuple2<String, Integer> tuple) -> !tuple.\_1.isEmpty());

// Call reduceByKey to count occurrence of each word

JavaPairRDD<String, Integer> wordsAndTheirCount = filteredWordsList

.reduceByKey(new Function2<Integer, Integer, Integer>() {

private static final long serialVersionUID = 1L;

public Integer call(Integer x, Integer y) {

return x + y;

}

});

// Sort the list by word (which is the key)

JavaPairRDD<String, Integer> sortedWordsAndTheirCount = wordsAndTheirCount.sortByKey();

// Save the word count list back to the text file, causing evaluation.

sortedWordsAndTheirCount.saveAsTextFile(outputDirName);

// close the spark context

sparkContext.close();

}

}

I then compiled and built the package using maven:

rpulekar-m1:Assignment4 rpulekar$ pwd

/Users/rpulekar/work/big-data-analytics-harvard/eclipse\_workspace\_for\_course/Assignment4

rpulekar-m1:Assignment4 rpulekar$ ls

bin files pom.xml src target

rpulekar-m1:Assignment4 rpulekar$ mvn clean && mvn compile && mvn package

[INFO] Scanning for projects...

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] Building Assignment4 0.0.1

[INFO] ------------------------------------------------------------------------

[INFO]

[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ Assignment4 ---

[INFO] Deleting /Users/rpulekar/work/big-data-analytics-harvard/eclipse\_workspace\_for\_course/Assignment4/target

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 0.183 s

[INFO] Finished at: 2016-02-25T22:22:31-05:00

[INFO] Final Memory: 8M/309M

[INFO] ------------------------------------------------------------------------

[INFO] Scanning for projects...

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] Building Assignment4 0.0.1

[INFO] ------------------------------------------------------------------------

[INFO]

[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ Assignment4 ---

[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!

[INFO] skip non existing resourceDirectory /Users/rpulekar/work/big-data-analytics-harvard/eclipse\_workspace\_for\_course/Assignment4/src/main/resources

[INFO]

[INFO] --- maven-compiler-plugin:3.3:compile (default-compile) @ Assignment4 ---

[INFO] Changes detected - recompiling the module!

[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!

[INFO] Compiling 4 source files to /Users/rpulekar/work/big-data-analytics-harvard/eclipse\_workspace\_for\_course/Assignment4/target/classes

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 1.578 s

[INFO] Finished at: 2016-02-25T22:22:33-05:00

[INFO] Final Memory: 31M/399M

[INFO] ------------------------------------------------------------------------

[INFO] Scanning for projects...

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] Building Assignment4 0.0.1

[INFO] ------------------------------------------------------------------------

[INFO]

[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ Assignment4 ---

[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!

[INFO] skip non existing resourceDirectory /Users/rpulekar/work/big-data-analytics-harvard/eclipse\_workspace\_for\_course/Assignment4/src/main/resources

[INFO]

[INFO] --- maven-compiler-plugin:3.3:compile (default-compile) @ Assignment4 ---

[INFO] Nothing to compile - all classes are up to date

[INFO]

[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ Assignment4 ---

[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!

[INFO] skip non existing resourceDirectory /Users/rpulekar/work/big-data-analytics-harvard/eclipse\_workspace\_for\_course/Assignment4/src/test/resources

[INFO]

[INFO] --- maven-compiler-plugin:3.3:testCompile (default-testCompile) @ Assignment4 ---

[INFO] No sources to compile

[INFO]

[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ Assignment4 ---

[INFO] No tests to run.

[INFO]

[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ Assignment4 ---

[INFO] Building jar: /Users/rpulekar/work/big-data-analytics-harvard/eclipse\_workspace\_for\_course/Assignment4/target/Assignment4-0.0.1.jar

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 1.079 s

[INFO] Finished at: 2016-02-25T22:22:35-05:00

[INFO] Final Memory: 17M/309M

[INFO] ------------------------------------------------------------------------

rpulekar-m1:Assignment4 rpulekar$ ls -lha target/Assignment4-0.0.1.jar

-rw-r--r-- 1 rpulekar 1327142227 21K Feb 25 22:22 target/Assignment4-0.0.1.jar

rpulekar-m1:Assignment4 rpulekar$

I then copied over the generated jar file to the VM:

rpulekar-m1:shared\_dir\_via\_scp rpulekar$ scp -i ../private\_keys/joe\_id\_dsa Assignment4-0.0.1.jar joe@192.168.71.152:~/shared/

Assignment4-0.0.1.jar 100% 21KB 20.9KB/s 00:00

rpulekar-m1:shared\_dir\_via\_scp rpulekar$

I then executed the program on VM using spark-submit. The input to the program was Ulysses file on hdfs and output was sent to hdfs directory:

[joe@localhost ~]$ spark-submit --class e63.course.assignment4.WordCount shared/Assignment4-0.0.1.jar ulysses/4300.txt assignment4\_problem1\_output

SLF4J: Class path contains multiple SLF4J bindings.

SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/usr/lib/flume-ng/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.

SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]

16/02/25 19:55:01 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

16/02/25 19:55:02 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 192.168.71.152 instead (on interface eth0)

16/02/25 19:55:02 WARN Utils: Set SPARK\_LOCAL\_IP if you need to bind to another address

16/02/25 19:55:04 WARN MetricsSystem: Using default name DAGScheduler for source because spark.app.id is not set.

16/02/25 19:55:06 WARN DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.

Then I copied output from hdfs directory to local filesystem and copied head and tail of output file to different files

[joe@localhost ~]$ hadoop fs -get assignment4\_problem1\_output shared/

[joe@localhost ~]$ head -20 shared/assignment4\_problem1\_output/part-00000 >> shared/assignment4\_problem1\_output/assignment4\_problem1\_output\_head.txt

[joe@localhost ~]$ tail -20 shared/assignment4\_problem1\_output/part-00000 >> shared/assignment4\_problem1\_output/assignment4\_problem1\_output\_tail.txt

First 20 words in output (sorted by the word in ascending order):

(0,2)

(001,5)

(002,1)

(003,2)

(004,3)

(007,1)

(010,1)

(016,1)

(0175,1)

(020,1)

(028,1)

(049,1)

(050,1)

(1,49)

(10,14)

(100,4)

(1000,4)

(10000,1)

(1000000,1)

(1020,1)

Last 20 words in output (sorted by the word in ascending order):

(zinfandel,8)

(zinfandels,1)

(zingari,1)

(zion,5)

(zivio,1)

(zmellz,1)

(zodiac,2)

(zodiacal,2)

(zoe,103)

(zoefanny,1)

(zoes,2)

(zones,1)

(zoo,2)

(zoological,1)

(zouave,1)

(zouaves,1)

(zrads,4)

(zulu,1)

(zulus,1)

(zut,1)

Deliverables provided in assignment submission:

* The JAVA program (WordCount.java)
* Head of program output (assignment4\_problem1\_output\_head.txt)
* Tail of program output (assignment4\_problem1\_output\_tail.txt)

**Problem 2.** Write a working WordCount program using Spark Scala API that reads a file, e.g. Ulysis/4300.txt from a local file system directory and writes the results of your calculations to a local file. To improve your word count, remove any punctuation that might have attached itself to your words. Also transform all words into lower case so that the capitalization does not affect the word count. The original code is provided in the attached mini-example-scala.tar file. That archive also contains Scala Build Tool build.sbt file. Run your program and demonstrate that it works. Submit working code inside the customary MS Word Document. Describe steps in your program.

This is my version of WordCount program that uses Spark Scala API

package e63.course.assignment4

import org.apache.spark.\_

import org.apache.spark.SparkContext.\_

// This program is for the Assignment4 Problem2 of e63 course (Big Data Analytics)

object WordCount {

def main(args: Array[*String*]) {

// set input and out files/dirs

val inputFileName = args(0)

val outputDirName = args(1)

// Create a Spark Configuration

val sparkConf = new SparkConf().setAppName("Assignment4\_Problem2")

// Create a Scala Spark Context.

val sparkContext = new SparkContext(sparkConf)

// Load the input data.

val inputFileContents = sparkContext.textFile(inputFileName)

// Split the input data into words.

val wordsListRDD = inputFileContents.flatMap(line => line.split(" "))

// Transform into a map of word and default count of 1.

// The below replaceAll() is called to eliminate punctuations and to

// convert the words into lower case

val wordsListWithDefaultCount = wordsListRDD.map(word => (word.replaceAll("[^A-Za-z0-9]", "").toLowerCase(), 1))

// Filter the list to eliminate blanks

val filteredWordsList = wordsListWithDefaultCount.filter{case (word, count) => !word.isEmpty()}

// Call reduceByKey to count occurrence of each word

val wordsAndTheirCount = filteredWordsList.reduceByKey{case (x, y) => x + y}

// Sort the list by word (which is the key)

val sortedWordsAndTheirCount = wordsAndTheirCount.sortByKey()

// Save the word count back out to a text file, causing evaluation.

sortedWordsAndTheirCount.saveAsTextFile(outputDirName)

}

}

Then I built the program using **sbt**:

rpulekar-m1:Assignment4\_Problem2 rpulekar$ pwd

/Users/rpulekar/work/big-data-analytics-harvard/scala\_workspace\_for\_course/Assignment4\_Problem2

rpulekar-m1:Assignment4\_Problem2 rpulekar$ ls

bin build.sbt project src target

rpulekar-m1:Assignment4\_Problem2 rpulekar$ sbt clean package

[info] Loading global plugins from /Users/rpulekar/.sbt/0.13/plugins

[info] Set current project to Assignment4\_Problem2 (in build file:/Users/rpulekar/work/big-data-analytics-harvard/scala\_workspace\_for\_course/Assignment4\_Problem2/)

[success] Total time: 0 s, completed Feb 25, 2016 11:29:35 PM

[info] Updating {file:/Users/rpulekar/work/big-data-analytics-harvard/scala\_workspace\_for\_course/Assignment4\_Problem2/}assignment4\_problem2...

[info] Resolving org.fusesource.jansi#jansi;1.4 ...

[info] Done updating.

[info] Compiling 1 Scala source to /Users/rpulekar/work/big-data-analytics-harvard/scala\_workspace\_for\_course/Assignment4\_Problem2/target/scala-2.10/classes...

[info] Packaging /Users/rpulekar/work/big-data-analytics-harvard/scala\_workspace\_for\_course/Assignment4\_Problem2/target/scala-2.10/assignment4\_problem2\_2.10-0.0.1.jar ...

[info] Done packaging.

[success] Total time: 6 s, completed Feb 25, 2016 11:29:41 PM

rpulekar-m1:Assignment4\_Problem2 rpulekar$

rpulekar-m1:Assignment4\_Problem2 rpulekar$ ls -lha target/scala-2.10/

total 16

drwxr-xr-x 4 rpulekar 1327142227 136B Feb 25 23:29 .

drwxr-xr-x 5 rpulekar 1327142227 170B Feb 25 23:29 ..

-rw-r--r-- 1 rpulekar 1327142227 6.6K Feb 25 23:29 assignment4\_problem2\_2.10-0.0.1.jar

drwxr-xr-x 3 rpulekar 1327142227 102B Feb 25 23:29 classes

rpulekar-m1:Assignment4\_Problem2 rpulekar$

Then I

Then I copied over the jar file to VM:

rpulekar-m1:shared\_dir\_via\_scp rpulekar$ scp -i ../private\_keys/joe\_id\_dsa assignment4\_problem2\_2.10-0.0.1.jar joe@192.168.71.152:~/shared/

assignment4\_problem2\_2.10-0.0.1.jar 100% 6777 6.6KB/s 00:00

rpulekar-m1:shared\_dir\_via\_scp rpulekar$

Then I submitted the program to spark (with input as Ulysses file from local file system and output on local file system):

[joe@localhost ~]$ spark-submit --class e63.course.assignment4.WordCount shared/assignment4\_problem2\_2.10-0.0.1.jar file:///home/joe/ulysses\_on\_local\_file\_system/4300.txt file:///home/joe/program\_outputs/assignment4\_problem2\_output

SLF4J: Class path contains multiple SLF4J bindings.

SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/usr/lib/flume-ng/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.

SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]

16/02/25 20:56:57 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

16/02/25 20:56:57 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 192.168.71.152 instead (on interface eth0)

16/02/25 20:56:57 WARN Utils: Set SPARK\_LOCAL\_IP if you need to bind to another address

16/02/25 20:56:59 WARN MetricsSystem: Using default name DAGScheduler for source because spark.app.id is not set.

16/02/25 20:57:02 WARN DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.

[joe@localhost ~]$

[joe@localhost ~]$ cat program\_outputs/assignment4\_problem2\_output/part-00000 | head -20

(0,2)

(001,5)

(002,1)

(003,2)

(004,3)

(007,1)

(010,1)

(016,1)

(0175,1)

(020,1)

(028,1)

(049,1)

(050,1)

(1,49)

(10,14)

(100,4)

(1000,4)

(10000,1)

(1000000,1)

(1020,1)

[joe@localhost ~]$ cat program\_outputs/assignment4\_problem2\_output/part-00000 | tail -20

(zinfandel,8)

(zinfandels,1)

(zingari,1)

(zion,5)

(zivio,1)

(zmellz,1)

(zodiac,2)

(zodiacal,2)

(zoe,103)

(zoefanny,1)

(zoes,2)

(zones,1)

(zoo,2)

(zoological,1)

(zouave,1)

(zouaves,1)

(zrads,4)

(zulu,1)

(zulus,1)

(zut,1)

[joe@localhost ~]$

Above is listed the first 20 and last 20 lines of output (sorted by the word in ascending order).

Deliverables provided in assignment submission:

* The Scala Program (Assignment4\_Problem2.scala)
* Head of the output (assignment4\_problem2\_output\_head.txt)
* Tail of the output (assignment4\_problem2\_output\_tail.txt)

**Problem 3.** Write a working WordCount script using Spark Python API. Read Ulysis (4300.txt) file from an HDFS directory and write the results of your calculations to an HDFS file. To improve your word count, remove any punctuation that might have attached itself to your words. Also transform all words into lower case so that the capitalization does not affect the word count. Run your script using submit-spark tool and demonstrate that it works. Submit working code. Describe steps in your program in the MS Word document.

My version of WordCount program in Spark Python API is:

'''

Created on Feb 23, 2016

@author: rpulekar

This program is for Assignment4 Problem3 of e63 course (Big Data Analytics)

'''

from pyspark import SparkConf, SparkContext

import string

import sys

# Set the Spark configuration

conf = SparkConf().setAppName('Assignment4\_Problem3')

# Set the spark context

sparkContext = SparkContext(conf = conf)

# set input/output filenames/dirs

input\_file\_name = sys.argv[1]

output\_dir\_name = sys.argv[2]

# Load the input data.

inputFileContents = sparkContext.textFile(input\_file\_name)

# Tokenize each line

words\_list\_RDD = inputFileContents.flatMap(lambda line: line.split())

# Convert tokens to lower-case & removes punctuation before creating a tuple (token, 1) for each word

words\_list\_with\_default\_count = words\_list\_RDD.map(lambda word: (str(word.lower()).translate(None,string.punctuation), 1))

# Filter the list to eliminate blanks

filtered\_words\_list = words\_list\_with\_default\_count.filter(lambda keyValue: len(keyValue[0])>0)

# Call reduceByKey to count occurrence of each word

words\_and\_their\_count = filtered\_words\_list.reduceByKey(lambda a, b: a+b)

# Sort the list by word (which is the key)

sorted\_words\_and\_their\_count=words\_and\_their\_count.sortByKey()

# Save the word count list back to the text file, causing evaluation.

sorted\_words\_and\_their\_count.saveAsTextFile(output\_dir\_name)

Then I copied over the py file to virtual machine:

rpulekar-m1:shared\_dir\_via\_scp rpulekar$ scp -i ../private\_keys/joe\_id\_dsa WordCount.py joe@192.168.71.152:~/shared

WordCount.py 100% 1345 1.3KB/s 00:00

rpulekar-m1:shared\_dir\_via\_scp rpulekar$

Then I submitted the program to spark-submit. Input to the program is Ulysses text present in hdfs. Output of the program is directory in hdfs

[joe@localhost ~]$ spark-submit shared/WordCount.py ulysses/4300.txt assignment4\_problem3\_output

SLF4J: Class path contains multiple SLF4J bindings.

SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/usr/lib/flume-ng/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.

SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]

16/02/25 21:32:49 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

16/02/25 21:32:49 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 192.168.71.152 instead (on interface eth0)

16/02/25 21:32:49 WARN Utils: Set SPARK\_LOCAL\_IP if you need to bind to another address

16/02/25 21:32:52 WARN MetricsSystem: Using default name DAGScheduler for source because spark.app.id is not set.

16/02/25 21:32:54 WARN DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.

[joe@localhost ~]$

Then I loaded the output file from hdfs into local filesystem:

[joe@localhost ~]$ hadoop fs -get assignment4\_problem3\_output shared/

[joe@localhost ~]$ head -20 shared/assignment4\_problem3\_output/part-00000 >> shared/assignment4\_problem3\_output/assignment4\_problem3\_output\_head.txt

[joe@localhost ~]$ tail -20 shared/assignment4\_problem3\_output/part-00000 >> shared/assignment4\_problem3\_output/assignment4\_problem3\_output\_tail.txt

[joe@localhost ~]$

First 20 lines of output (sorted by the word in ascending order):

('0', 2)

('001', 5)

('002', 1)

('003', 2)

('004', 3)

('007', 1)

('010', 1)

('016', 1)

('0175', 1)

('020', 1)

('028', 1)

('049', 1)

('050', 1)

('1', 49)

('10', 14)

('100', 4)

('1000', 4)

('10000', 1)

('1000000', 1)

('1020', 1)

Last 20 lines of output (sorted by the word in ascending order):

('zinfandel', 8)

('zinfandels', 1)

('zingari', 1)

('zion', 5)

('zivio', 1)

('zmellz', 1)

('zodiac', 2)

('zodiacal', 2)

('zoe', 103)

('zoefanny', 1)

('zoes', 2)

('zones', 1)

('zoo', 2)

('zoological', 1)

('zouave', 1)

('zouaves', 1)

('zrads', 4)

('zulu', 1)

('zulus', 1)

('zut', 1)

Deliverables provided in assignment submission:

* the spark program (WordCount.py)
* head of the output (assignment4\_problem3\_output\_head.txt)
* tail of the output (assignment4\_problem3\_output\_tail.txt)

**Problem 4**. In a Spark API of your choice, write a working BigramCount program which would count occurrences of every pair of consecutive words. You should clean your words just as you did in the previous problem by removing punctuations and cases. However, do not count two words separated by a point at the end of a sentence as a bigram. If you are an experienced programmer add to the bigram count word pairs in which the first word is the last word on the line and the second word is the first word on the subsequent line. If you are not an experienced programmer, than do not do it. Test your program on a small text file, where for comparison, you could identify bigrams manually. Run your program on Ulysis(4300.txt) file and demonstrate that it works. Provide us with the total count of your bigrams, first 20 bigrams and all bigrams containing word “heaven”. Read your file from the local operating system and write results to the local operating system. Include working code in the MS Word Document. Submit the file with the complete working code separately. Describe steps in your program in the MS Word document.

I have used the experienced programmer approach.

The program is in JAVA

package e63.course.assignment4;

import java.util.ArrayList;

import java.util.List;

import java.util.StringTokenizer;

import org.apache.spark.SparkConf;

import org.apache.spark.api.java.JavaPairRDD;

import org.apache.spark.api.java.JavaRDD;

import org.apache.spark.api.java.JavaSparkContext;

import org.apache.spark.api.java.function.FlatMapFunction;

import org.apache.spark.api.java.function.Function2;

import org.apache.spark.api.java.function.PairFunction;

import scala.Tuple2;

/\*\*

\* This program is for the Assignment4 Problem4 of e63 course (Big Data

\* Analytics)

\*

\* This uses the EXPERIENCED PROGRAMMER approach

\*

\* @author rpulekar

\*

\*/

public class BigramCount {

public static void main(String[] args) throws Exception {

// set input and out files/dirs

String inputFileName = args[0];

String outputDirName = args[1];

// Create a Java Spark configuration.

SparkConf sparkConf = new SparkConf().setAppName("Assignment4\_Problem4\_Advanced");

// Create a Java Spark Context

JavaSparkContext sparkContext = new JavaSparkContext(sparkConf);

// Load the input data as whole test file, so that entire file is read

// as one string

JavaPairRDD<String, String> inputFileContents = sparkContext.wholeTextFiles(inputFileName);

// Calculate full bigrams list

JavaRDD<String> bigramsListRDD = inputFileContents

.flatMap(new FlatMapFunction<Tuple2<String, String>, String>() {

private static final long serialVersionUID = 1L;

public Iterable<String> call(Tuple2<String, String> tuple) {

StringTokenizer bigramsListTemp = new StringTokenizer(tuple.\_2, ".");

List<String> listOfBigrams = new ArrayList<String>();

while (bigramsListTemp.hasMoreTokens()) {

listOfBigrams.addAll(getBigrams(bigramsListTemp.nextToken()));

}

return listOfBigrams;

}

});

// Transform into word and count.

JavaPairRDD<String, Integer> bigramsAndDefaultCount = bigramsListRDD

.mapToPair(new PairFunction<String, String, Integer>() {

private static final long serialVersionUID = 1L;

public Tuple2<String, Integer> call(String x) {

return new Tuple2<String, Integer>(x, 1);

}

});

// use reduceByKey to count occurrence of each bigram

JavaPairRDD<String, Integer> bigramsAndFinalCount = bigramsAndDefaultCount

.reduceByKey(new Function2<Integer, Integer, Integer>() {

private static final long serialVersionUID = 1L;

public Integer call(Integer x, Integer y) {

return x + y;

}

});

// sort the list by bigrams (which is the key)

JavaPairRDD<String, Integer> sortedBigramsAndFinalCount = bigramsAndFinalCount.sortByKey();

// save the sorted bigrams into a text file

sortedBigramsAndFinalCount.saveAsTextFile(outputDirName);

// close the spark context

sparkContext.close();

}

// Below is function to get bigrams from a string

public static List<String> getBigrams(String line) {

// tokenize the line

StringTokenizer tokenizer = new StringTokenizer(line);

int numberOfWordsInSentence = tokenizer.countTokens();

// if there are less than 2 tokens on the line then there is no bigram

// on that line

if (numberOfWordsInSentence < 2) {

return new ArrayList<String>(0);

}

// construct list of words in the sentence

List<String> wordsInSentence = new ArrayList<String>(numberOfWordsInSentence);

while (tokenizer.hasMoreTokens()) {

String token = tokenizer.nextToken();

if (token != null && !token.isEmpty()) {

wordsInSentence.add(token);

}

}

List<String> bigrams = new ArrayList<String>(numberOfWordsInSentence - 1);

// loop for creating bigrams

for (int wordCounter = 0; wordCounter < numberOfWordsInSentence - 1; wordCounter++) {

// get the first word of potential bigram

String firstWord = wordsInSentence.get(wordCounter);

// if first word ends with . or ? or ! then it is not part of a

// bigram

if (firstWord.endsWith(".") || firstWord.endsWith("?") || firstWord.endsWith("!")) {

continue;

}

// clean first word by elimiating punctuations and convert it to

// lower case

firstWord = firstWord.replaceAll("[^A-Za-z0-9]", "").toLowerCase();

// clean second word by elimiating punctuations and convert it to

// lower case

String secondWord = wordsInSentence.get(wordCounter + 1).replaceAll("[^A-Za-z0-9]", "").toLowerCase();

// add current bigram to the list of bigrams if neither of them are

// not empty

if (!firstWord.isEmpty() && !secondWord.isEmpty()) {

bigrams.add(firstWord + " " + secondWord);

}

}

return bigrams;

}

}

And this is the program to find bigrams **containing word heaven**:

package e63.course.assignment4;

import java.util.ArrayList;

import java.util.List;

import java.util.StringTokenizer;

import org.apache.spark.SparkConf;

import org.apache.spark.api.java.JavaPairRDD;

import org.apache.spark.api.java.JavaRDD;

import org.apache.spark.api.java.JavaSparkContext;

import org.apache.spark.api.java.function.FlatMapFunction;

import org.apache.spark.api.java.function.Function2;

import org.apache.spark.api.java.function.PairFunction;

import scala.Tuple2;

/\*\*

\* This program is for the Assignment4 Problem4 of e63 course (Big Data

\* Analytics)

\*

\* This program is for finding all words containing the word heaven

\*

\* This uses the EXPERIENCED PROGRAMMER approach

\*

\* @author rpulekar

\*

\*/

public class BigramCount\_WordHeaven {

public static void main(String[] args) throws Exception {

// set input and out files/dirs

String inputFileName = args[0];

String outputFileDir = args[1];

// Create a Java Spark Config.

SparkConf sparkConf = new SparkConf().setAppName("Assignment4\_Problem4\_Advanced\_WordHeaven");

// Create a Java Spark Context

JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConf);

// Load the input data as whole test file, so that entire file is read

// as one string

JavaPairRDD<String, String> inputFileContents = javaSparkContext.wholeTextFiles(inputFileName);

JavaRDD<String> bigramsListRDD = inputFileContents

.flatMap(new FlatMapFunction<Tuple2<String, String>, String>() {

private static final long serialVersionUID = 1L;

public Iterable<String> call(Tuple2<String, String> tuple) {

StringTokenizer tokenizer = new StringTokenizer(tuple.\_2, ".");

List<String> bigramsListTemp = new ArrayList<String>();

while (tokenizer.hasMoreTokens()) {

// will be executed for each sentence

bigramsListTemp.addAll(getBigrams(tokenizer.nextToken()));

}

return bigramsListTemp;

}

});

// Transform into word and count.

JavaPairRDD<String, Integer> bigramsAndDefaultCount = bigramsListRDD

.mapToPair(new PairFunction<String, String, Integer>() {

private static final long serialVersionUID = 1L;

public Tuple2<String, Integer> call(String bigram) {

return new Tuple2<String, Integer>(bigram, 1);

}

});

// filter the bigrams to include only those that contain word **heaven**

JavaPairRDD<String, Integer> filteredBigramsAndDefaultCount = bigramsAndDefaultCount

.filter((Tuple2<String, Integer> tuple) -> (tuple.\_1 != null && tuple.\_1.contains("heaven")));

// use reduceByKey to count occurrence of each bigram

JavaPairRDD<String, Integer> bigramsAndFinalCount = filteredBigramsAndDefaultCount

.reduceByKey(new Function2<Integer, Integer, Integer>() {

private static final long serialVersionUID = 1L;

public Integer call(Integer x, Integer y) {

return x + y;

}

});

// sort the list by bigrams (which is the key)

JavaPairRDD<String, Integer> sortedBigramsAndCount = bigramsAndFinalCount.sortByKey();

// save the sorted bigrams into a text file

sortedBigramsAndCount.saveAsTextFile(outputFileDir);

// close the spark context

javaSparkContext.close();

}

public static List<String> getBigrams(String line) {

// tokenize the line

StringTokenizer tokenizer = new StringTokenizer(line);

int numberOfWordsInSentence = tokenizer.countTokens();

// if there are less than 2 tokens on the line then there is no bigram

// on that line

if (numberOfWordsInSentence < 2) {

return new ArrayList<String>(0);

}

// construct list of words in the sentence

List<String> wordsInSentence = new ArrayList<String>(numberOfWordsInSentence);

while (tokenizer.hasMoreTokens()) {

String token = tokenizer.nextToken();

if (token != null && !token.isEmpty()) {

wordsInSentence.add(token);

}

}

List<String> bigrams = new ArrayList<String>(numberOfWordsInSentence - 1);

// loop for creating bigrams

for (int wordCounter = 0; wordCounter < numberOfWordsInSentence - 1; wordCounter++) {

// get the first word of potential bigram

String firstWord = wordsInSentence.get(wordCounter);

// if first word ends with . or ? or ! then it is not part of a

// bigram

if (firstWord.endsWith(".") || firstWord.endsWith("?") || firstWord.endsWith("!")) {

continue;

}

// clean first word by elimiating punctuations and convert it to

// lower case

firstWord = firstWord.replaceAll("[^A-Za-z0-9]", "").toLowerCase();

// clean second word by elimiating punctuations and convert it to

// lower case

String secondWord = wordsInSentence.get(wordCounter + 1).replaceAll("[^A-Za-z0-9]", "").toLowerCase();

// add current bigram to the list of bigrams if neither of them are

// not empty

if (!firstWord.isEmpty() && !secondWord.isEmpty()) {

bigrams.add(firstWord + " " + secondWord);

}

}

return bigrams;

}

}

I created the jar with mvn command:

rpulekar-m1:Assignment4 rpulekar$ mvn clean && mvn compile && mvn package

[INFO] Scanning for projects...

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] Building Assignment4 0.0.1

[INFO] ------------------------------------------------------------------------

[INFO]

[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ Assignment4 ---

[INFO] Deleting /Users/rpulekar/work/big-data-analytics-harvard/eclipse\_workspace\_for\_course/Assignment4/target

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 0.341 s

[INFO] Finished at: 2016-02-26T02:02:11-05:00

[INFO] Final Memory: 7M/245M

[INFO] ------------------------------------------------------------------------

[INFO] Scanning for projects...

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] Building Assignment4 0.0.1

[INFO] ------------------------------------------------------------------------

[INFO]

[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ Assignment4 ---

[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!

[INFO] skip non existing resourceDirectory /Users/rpulekar/work/big-data-analytics-harvard/eclipse\_workspace\_for\_course/Assignment4/src/main/resources

[INFO]

[INFO] --- maven-compiler-plugin:3.3:compile (default-compile) @ Assignment4 ---

[INFO] Changes detected - recompiling the module!

[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!

[INFO] Compiling 4 source files to /Users/rpulekar/work/big-data-analytics-harvard/eclipse\_workspace\_for\_course/Assignment4/target/classes

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 4.650 s

[INFO] Finished at: 2016-02-26T02:02:17-05:00

[INFO] Final Memory: 29M/281M

[INFO] ------------------------------------------------------------------------

[INFO] Scanning for projects...

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] Building Assignment4 0.0.1

[INFO] ------------------------------------------------------------------------

[INFO]

[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ Assignment4 ---

[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!

[INFO] skip non existing resourceDirectory /Users/rpulekar/work/big-data-analytics-harvard/eclipse\_workspace\_for\_course/Assignment4/src/main/resources

[INFO]

[INFO] --- maven-compiler-plugin:3.3:compile (default-compile) @ Assignment4 ---

[INFO] Nothing to compile - all classes are up to date

[INFO]

[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ Assignment4 ---

[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!

[INFO] skip non existing resourceDirectory /Users/rpulekar/work/big-data-analytics-harvard/eclipse\_workspace\_for\_course/Assignment4/src/test/resources

[INFO]

[INFO] --- maven-compiler-plugin:3.3:testCompile (default-testCompile) @ Assignment4 ---

[INFO] No sources to compile

[INFO]

[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ Assignment4 ---

[INFO] No tests to run.

[INFO]

[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ Assignment4 ---

[INFO] Building jar: /Users/rpulekar/work/big-data-analytics-harvard/eclipse\_workspace\_for\_course/Assignment4/target/Assignment4-0.0.1.jar

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 1.540 s

[INFO] Finished at: 2016-02-26T02:02:20-05:00

[INFO] Final Memory: 16M/309M

[INFO] ------------------------------------------------------------------------

rpulekar-m1:Assignment4 rpulekar$

rpulekar-m1:Assignment4 rpulekar$ ls -lha target/

total 48

drwxr-xr-x 7 rpulekar 1327142227 238B Feb 26 02:02 .

drwxr-xr-x 10 rpulekar 1327142227 340B Feb 26 02:02 ..

-rw-r--r-- 1 rpulekar 1327142227 20K Feb 26 02:02 Assignment4-0.0.1.jar

drwxr-xr-x 3 rpulekar 1327142227 102B Feb 26 02:02 classes

drwxr-xr-x 3 rpulekar 1327142227 102B Feb 26 02:02 generated-sources

drwxr-xr-x 3 rpulekar 1327142227 102B Feb 26 02:02 maven-archiver

drwxr-xr-x 3 rpulekar 1327142227 102B Feb 26 02:02 maven-status

rpulekar-m1:Assignment4 rpulekar$

I copied the jar file to the VM:

rpulekar-m1:shared\_dir\_via\_scp rpulekar$ scp -i ../private\_keys/joe\_id\_dsa Assignment4-0.0.1.jar joe@192.168.71.152:~/shared/

Assignment4-0.0.1.jar 100% 20KB 20.4KB/s 00:00

rpulekar-m1:shared\_dir\_via\_scp rpulekar$ Assignment4-0.0.1.jar

Then I submitted the BigramCount program to Spark (program input is from local filesystem and output is to local filesystem):

[joe@localhost ~]$ spark-submit --class e63.course.assignment4.BigramCount shared/Assignment4-0.0.1.jar file:///home/joe/ulysses\_on\_local\_file\_system/4300.txt file:///home/joe/shared/assignment4\_problem4\_bigrams/

SLF4J: Class path contains multiple SLF4J bindings.

SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/usr/lib/flume-ng/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.

SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]

16/02/25 23:20:02 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

16/02/25 23:20:02 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 192.168.71.152 instead (on interface eth0)

16/02/25 23:20:02 WARN Utils: Set SPARK\_LOCAL\_IP if you need to bind to another address

16/02/25 23:20:04 WARN MetricsSystem: Using default name DAGScheduler for source because spark.app.id is not set.

16/02/25 23:20:07 WARN DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.

[joe@localhost ~]$ wc shared/assignment4\_problem4\_bigrams/part-00000 -l

138518

So basically there are 138518 bigrams (counted with the experienced programmer approach).

First 20 bigrams are (ordered by bigram in ascending order)

format: *(bigram, number\_of\_occurrences)*:

(0 would,1)

(001 004,2)

(001 007,1)

(001 016,1)

(001 050,1)

(002 020,1)

(003 001,2)

(004 003,1)

(004 004,1)

(004 170,1)

(007 010,1)

(010 002,1)

(016 001,1)

(0175 2193,1)

(020 028,1)

(028 001,1)

(049 176,1)

(050 001,1)

(1 12,1)

(1 2,1)

Last 20 bigrams are (ordered by bigram in ascending order):

format: *(bigram, number\_of\_occurrences)*:

(zoe twirls,1)

(zoe what,1)

(zoe whispers,2)

(zoe who,2)

(zoe widening,1)

(zoe with,1)

(zoe yes,1)

(zoe yorkshire,1)

(zoe you,3)

(zoe youll,1)

(zoefanny i,1)

(zoes neck,2)

(zones its,1)

(zoological biscuits,1)

(zouave to,1)

(zouaves uniform,1)

(zrads and,1)

(zrads zrads,2)

(zulu chief,1)

(zulus or,1)

To find all bigrams containing the word heaven,

I ran the following:

[joe@localhost ~]$ spark-submit --class e63.course.assignment4.BigramCount\_WordHeaven shared/Assignment4-0.0.1.jar file:///home/joe/ulysses\_on\_local\_file\_system/4300.txt file:///home/joe/shared/assignment4\_problem4\_bigrams\_heaven/

SLF4J: Class path contains multiple SLF4J bindings.

SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/usr/lib/flume-ng/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.

SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]

16/02/25 23:33:09 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

16/02/25 23:33:10 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 192.168.71.152 instead (on interface eth0)

16/02/25 23:33:10 WARN Utils: Set SPARK\_LOCAL\_IP if you need to bind to another address

16/02/25 23:33:11 WARN MetricsSystem: Using default name DAGScheduler for source because spark.app.id is not set.

16/02/25 23:33:14 WARN DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.

[joe@localhost ~]$

And listed all bigrams containing the word heaven:

format: *(bigram, number\_of\_occurrences)*:

(a heaven,1)

(a heavenbeast,1)

(a heavengrot,1)

(a heavenman,1)

(a heaventree,1)

(and heaven,1)

(angry heavens,1)

(by heaven,2)

(deserted heavens,1)

(gracious heaven,1)

(hearts heaven,1)

(heaven 4,1)

(heaven a,1)

(heaven and,4)

(heaven becomes,1)

(heaven by,2)

(heaven calling,1)

(heaven ever,1)

(heaven foretold,1)

(heaven had,1)

(heaven hight,1)

(heaven i,1)

(heaven if,1)

(heaven murmuring,1)

(heaven ned,1)

(heaven spilt,1)

(heaven theodore,1)

(heaven theres,1)

(heaven was,2)

(heaven were,2)

(heaven when,1)

(heavenbeast not,1)

(heavenborn earth,1)

(heavengrot not,1)

(heavenly host,1)

(heavenly man,1)

(heavenly weather,1)

(heavens alone,1)

(heavens by,1)

(heavens nay,1)

(heavens own,1)

(heavens so,1)

(heavens to,1)

(heavens upon,1)

(heavens were,1)

(heaventree not,1)

(heaventree of,1)

(heavenworld he,1)

(in heaven,9)

(into heaven,1)

(like heaven,1)

(nearer heaven,1)

(of heaven,7)

(seventh heaven,1)

(thank heaven,1)

(the heavenborn,1)

(the heavenly,2)

(the heavens,5)

(the heaventree,1)

(the heavenworld,1)

(thumb heavenward,1)

(to heaven,8)

(to heavens,1)

(visit heaven,1)

(was heavenly,1)

There are 65 bigrams containing the word heaven (counted with experienced programmer approach)

Deliverables provided in assignment submission:

* JAVA Spark program for finding bigrams (BigramCount.java)
* JAVA Spark program for finding programs with word heaven (BigramCount\_WordHeaven.java)
* First 20 bigrams (bigrams\_first\_20.txt)
* Last 20 bigrams (bigrams\_last\_20.txt)
* Bigrams with word heaven (bigrams\_with\_word\_heaven.txt)

Please, describe every step of your work and present all intermediate and final results in a Word document. Please, copy past text version of your command. We cannot retype text that is in JPG images. Please, always submit a copy of original, working scripts and class files you used as separate files. Sometimes we need to run your code and retyping is too costly. Please, submit to the class drop box. For issues and comments visit the class Discussion Board .