Session 20 : SPARK Streaming

Assignment 2

**Problem Statement**

Read two streams

1. List of strings input by user

2. Real-time set of offensive words

Find the word count of the offensive words inputted by the user as per the

real-time set of offensive words.

**Solution:-**

**Scala Project -** 

**Offensive Words files -**  

**Jar file -** 

**Command to run NetCat ->** nc -lk 9999

**Command to run Jar file ->** spark-submit --master local[2] --class org.scala.OffensiveWordCount Assignment20\_2.jar localhost 9999

**Command to put files in HDFS –**

hadoop fs -put offensiveWordsFile.txt /offensiveWords

hadoop fs -put offWords2.txt /offensiveWords

**Code:---**

**1 . StreamingExample.scala**

package org.scala

import org.apache.log4j.{Level, Logger}

import org.apache.spark.internal.Logging

object StreamingExamples extends Logging {

def setStreamingLogLevels() {

val log4jInitialized = Logger.getRootLogger.getAllAppenders.hasMoreElements

if (!log4jInitialized) {

// We first log something to initialize Spark's default logging, then we override the

// logging level.

logInfo("Setting log level to [WARN] for streaming example." +

" To override add a custom log4j.properties to the classpath.")

Logger.getRootLogger.setLevel(Level.WARN)

}

}

}

**2. OffensiveWordCount.scala**

package org.scala

import org.apache.spark.SparkConf

import org.apache.spark.storage.StorageLevel

import org.apache.spark.streaming.{Seconds, StreamingContext}

import scala.collection.mutable.ArrayBuffer

import org.apache.spark.streaming.dstream.DStream

import org.apache.spark.SparkContext

/\*\*

\* Counts words in UTF8 encoded, '\n' delimited text received from the network every second.

\*

\* Usage: OffensiveWordCount <hostname> <port>

\* <hostname> and <port> describe the TCP server that Spark Streaming would connect to receive data.

\*

\* To run this on your local machine, you need to first run a Netcat server

\* `$ nc -lk 9999`

\*/

object OffensiveWordCount {

//ArrayBuffer to store list of offensive words in memory

val wordList: ArrayBuffer[String] = ArrayBuffer.empty[String];

def main(args: Array[String]) {

if (args.length < 2) {

System.err.println("Usage: OffensiveWordCount <hostname> <port>")

System.exit(1)

}

StreamingExamples.setStreamingLogLevels()

// Create the context with a 60 second batch size

val sparkConf = new SparkConf().setAppName("OffensiveWordCount")

val ssc = new StreamingContext(sparkConf, Seconds(60))

//Creating text file stream to store offensive words.

//It will monitor HDFS directory /offensiveWords

val offensiveLines = ssc.textFileStream("hdfs://localhost:9000/offensiveWords/");

val lines = ssc.socketTextStream(args(0), args(1).toInt, StorageLevel.MEMORY\_AND\_DISK\_SER);

// Create a socket stream on target ip:port and count the

// words in input stream of \n delimited text (eg. generated by 'nc')

// Note that no duplication in storage level only for running locally.

// Replication necessary in distributed scenario for fault tolerance.

//getting offensive words from file

val offensiveWordCount = offensiveLines.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(\_ + \_);

//storing offensive words in ArrayBuffer

offensiveWordCount.foreachRDD(a => { a.foreach(f => {wordList += f.\_1})});

//Getting all word count of all words entered by user

val wordCount = lines.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(\_ + \_);

//Getting word count of offensive words only

val offensiveWordsRDD = wordCount.filter {x => matchWord(x.\_1)%2==1 };

offensiveWordsRDD.print();

ssc.start()

ssc.awaitTermination()

}

/\*\*

\* Filter Method for offensive words

\*/

def matchWord(ln : String): Double={

val lineWords = ln.trim.toLowerCase();

var num: Double = 0;

for(y<-wordList)

{

if(y.toLowerCase() == lineWords)

{

num = 1;

return num;

}

}

return num;

}

}

**-----------------------------------Now Running the Spark Streaming program---------------------------------------------**

**Step 1- Starting netcat**



**Step 2 – Starting Spark streaming application**



Now the application has started we can provide text input in netcat as well as put offensive word file in HDFS directory.

Sample input text –

he is a dumb idiot hello

he is a dumb idiot and got troll

test john hello dash

he is a dumb idiot



Nothing is printed on spark app as the list of offensive words in empty

Now loading first offensive word file



Now upon entering same input text, the app is returning word count of offensive words in file **offensiveWordsFile.txt**



Adding another file with offensive words



From below screenshot we could see that the app is now returning the word count of offensive words that were loaded previously as well as loaded later from file **offWords2.txt.**

