CSEE5590 Big Data Programming

In Class Programming –11 Report
(Jongkook Son)

Project Overview:

Lesson Title: Apache Spark Streaming
Lesson Description: Apache Spark Streaming

Requirements/Task(s):

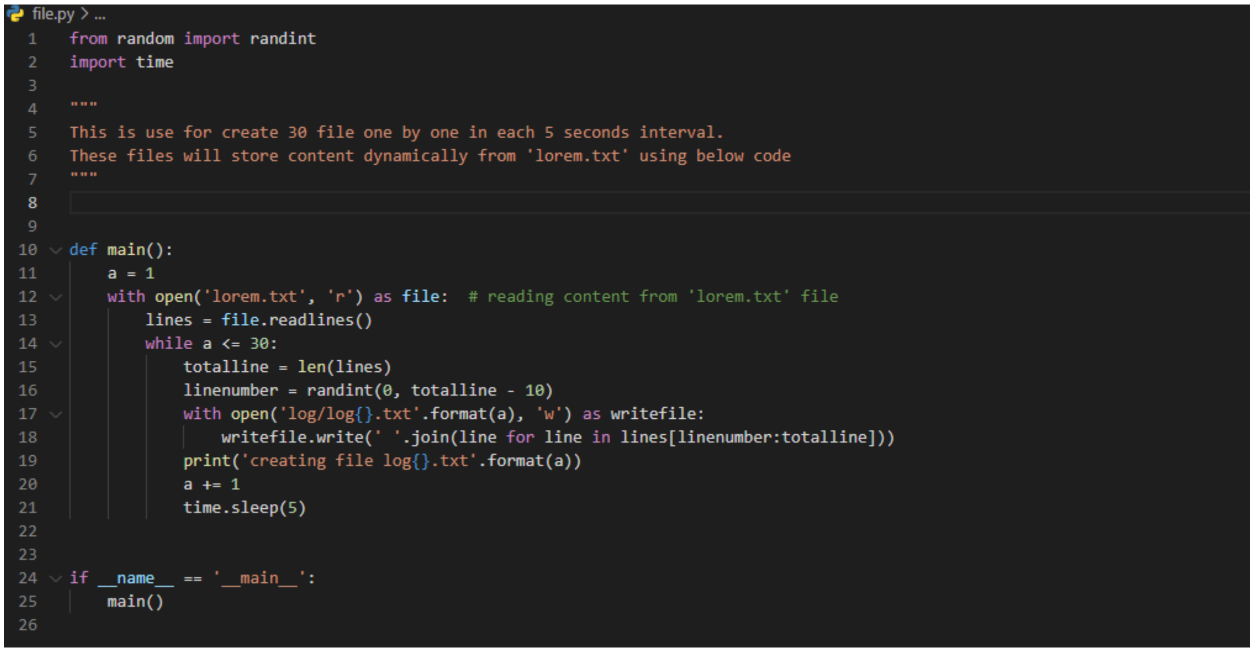
1. **Spark Streaming using Log File Generator:**
Spark Streaming using log file generator. Use the instructions in the slides
2. **Spark Streaming for TCP Socket:**
Write a spark word count program of Spark Streaming received from a data server listening on a TCP socket.
3. **Spark Streaming for Character Frequency using TCP Socket**:
Output Example
Character Count => List of Words (Distinct)
2 => [to, in, go]
3 => [the, law, and, can, how]
4 => [java, data, file]
Note: Ignore the case-sensitivity. Assume all the words are in lower case.

What I learned in ICP:

I learned How to implement streaming the data on spark. Spark Streaming is an extension of the core Spark API that provides scalable, high-performance, and fault-tolerant real-time streaming of data streams. Data Streaming is a technique for transferring data so that it can be processed as a steady and continuous stream. Spark Streaming is used to track the behavior of customers which can be used in business analysis. In this ICP, I used TextfileStream and SocketTextStream to handle realtime data. Wordcount and WordFrequency program was successfully implemented by sparkstreaming

Task1: Spark Streaming using Log File Generator

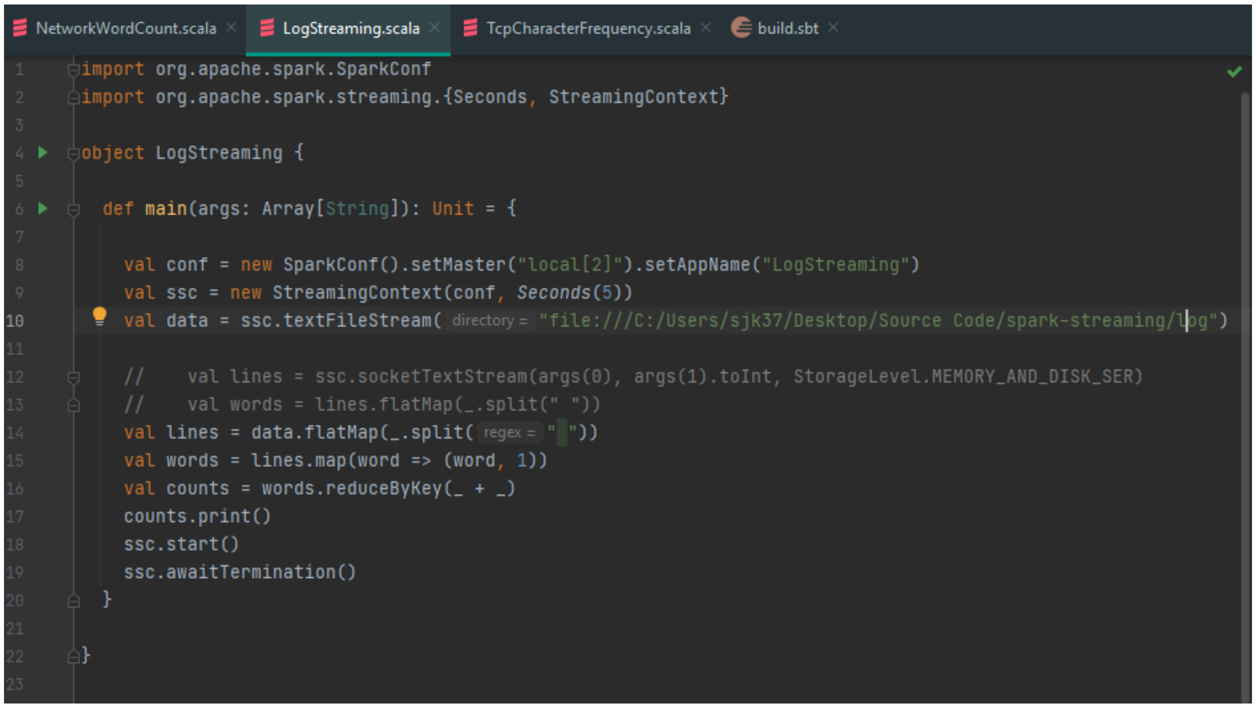
1. First using file.py create log file that is used for logStraming



file.py > ...

```python
from random import randint
import time

"""
This is use for create 30 file one by one in each 5 seconds interval.
These files will store content dynamically from 'lorem.txt' using below code
"""


def main():
    a = 1
    with open('lorem.txt', 'r') as file:  # reading content from 'lorem.txt' file
        lines = file.readlines()
        while a <= 30:
            totalline = len(lines)
            linenumber = randint(0, totalline - 10)
            with open('log/log{}.txt'.format(a), 'w') as writefile:
                writefile.write(' '.join(line for line in lines[linenumber:totalline]))
            print('creating file log{}.txt'.format(a))
            a += 1
            time.sleep(5)


if __name__ == '__main__':
    main()

```

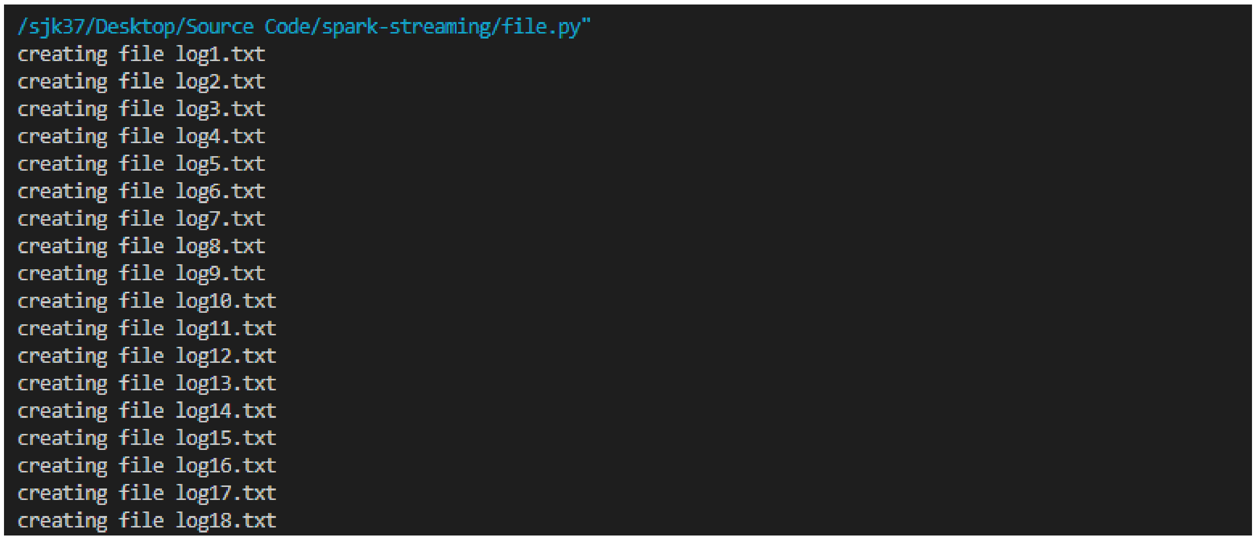
2. I created a Logstraming.scala which in charge of streaming log files and display word count for each file. To do this, First I need to run LogStreaming file then run the files.py. Because Spark's textFileStream creates a stream that watches a directory for new files only.



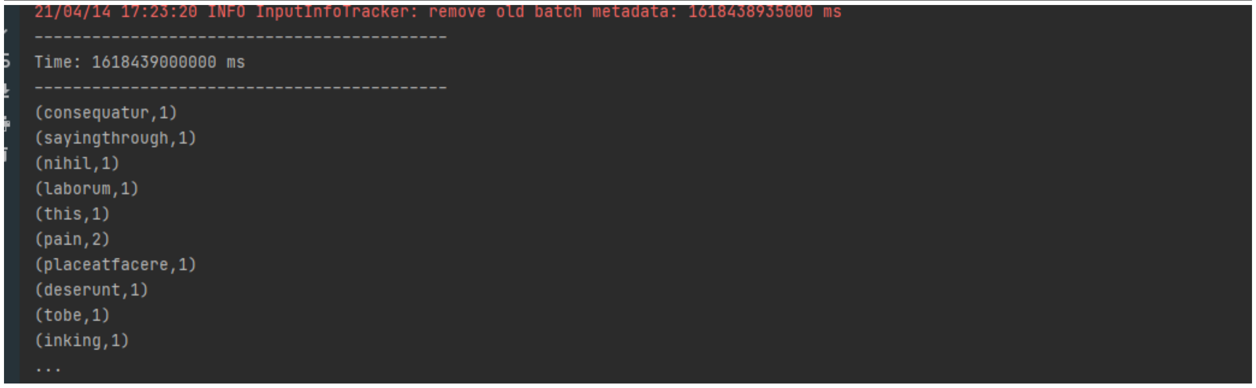
NetworkWordCount.scala × LogStreaming.scala × TcpCharacterFrequency.scala × build.sbt ×

```scala
import org.apache.spark.SparkConf
import org.apache.spark.streaming.{Seconds, StreamingContext}

object LogStreaming {

  def main(args: Array[String]): Unit = {

    val conf = new SparkConf().setMaster("local[2]").setAppName("LogStreaming")
    val ssc = new StreamingContext(conf, Seconds(5))
    val data = ssc.textFileStream( directory = "file:///C:/Users/sjk37/Desktop/Source Code/spark-streaming/log")

    //    val lines = ssc.socketTextStream(args(0), args(1).toInt, StorageLevel.MEMORY_AND_DISK_SER)
    //    val words = lines.flatMap(_.split(" "))
    val lines = data.flatMap(_.split( regex = " "))
    val words = lines.map(word => (word, 1))
    val counts = words.reduceByKey(_ + _)
    counts.print()
    ssc.start()
    ssc.awaitTermination()
  }

}

```

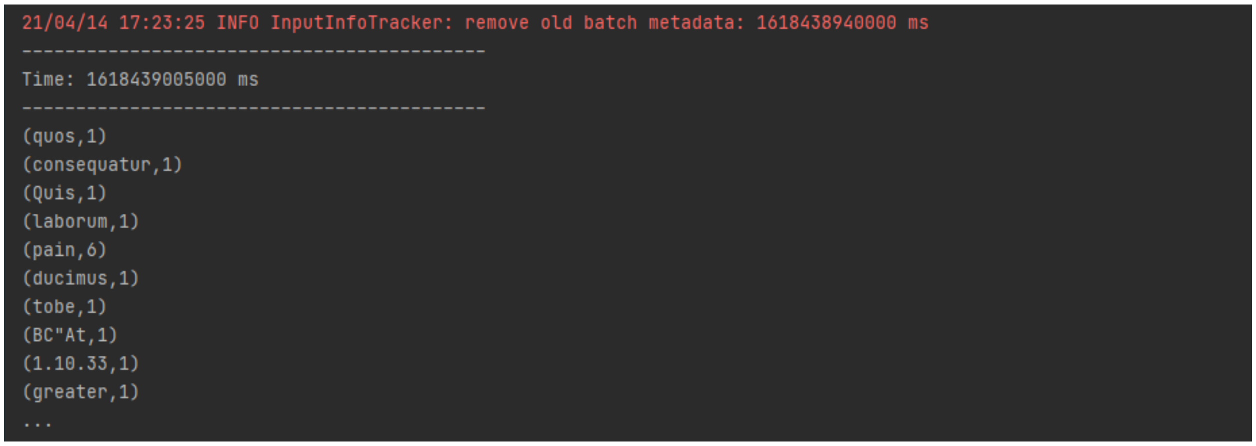
3. You can find out that while logfile is created, TextfileStreamer catches it and stream it successfully



```
/sjk37/Desktop/Source Code/spark-streaming/file.py"
creating file log1.txt
creating file log2.txt
creating file log3.txt
creating file log4.txt
creating file log5.txt
creating file log6.txt
creating file log7.txt
creating file log8.txt
creating file log9.txt
creating file log10.txt
creating file log11.txt
creating file log12.txt
creating file log13.txt
creating file log14.txt
creating file log15.txt
creating file log16.txt
creating file log17.txt
creating file log18.txt
```



```
21/04/14 17:23:20 INFO InputInfoTracker: remove old batch metadata: 1618438935000 ms
-------------------------------------------
Time: 1618439000000 ms
-------------------------------------------
(consequatur,1)
(sayingthrough,1)
(nihil,1)
(laborum,1)
(this,1)
(pain,2)
(placeatfacere,1)
(deserunt,1)
(tobe,1)
(inking,1)
...
```



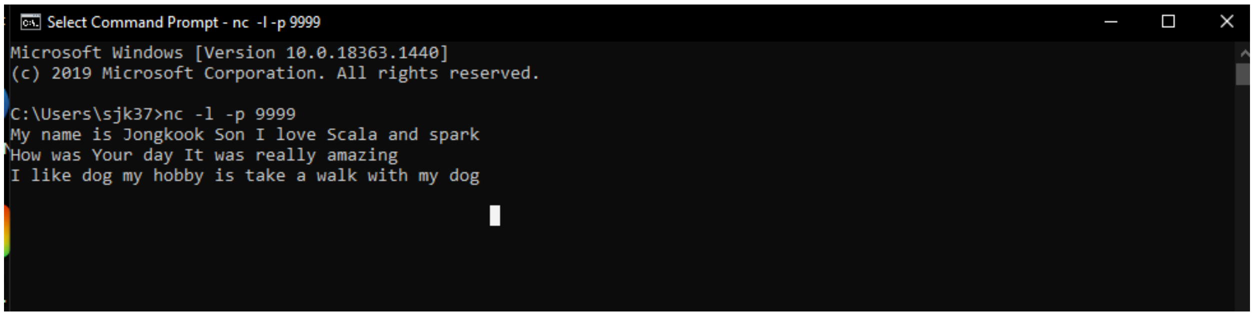
```
21/04/14 17:23:25 INFO InputInfoTracker: remove old batch metadata: 1618438940000 ms
-------------------------------------------
Time: 1618439005000 ms
-------------------------------------------
(quos,1)
(consequatur,1)
(Quis,1)
(laborum,1)
(pain,6)
(ducimus,1)
(tobe,1)
(BC"At,1)
(1.10.33,1)
(greater,1)
...
```

Task2: Spark Streaming for TCP Socket

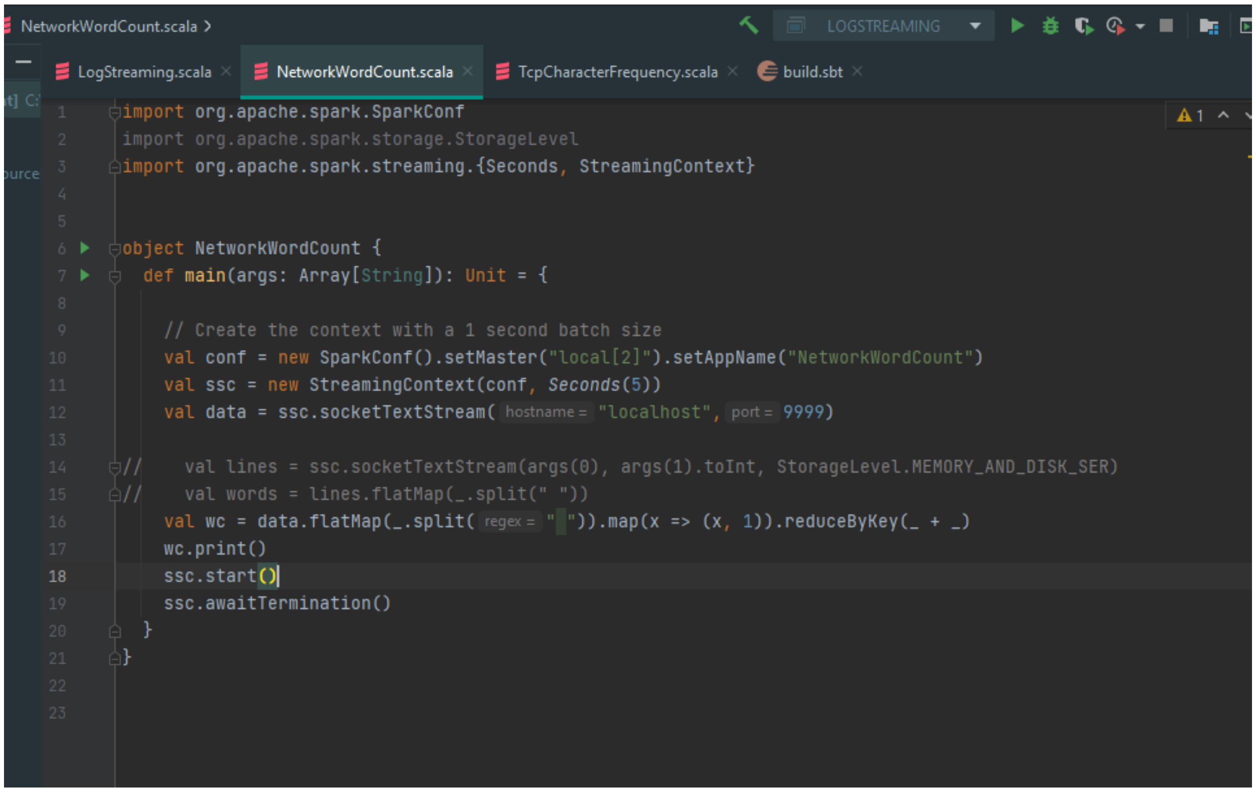
1.First, I need to run netcat using localhost number 9999.

2. Then I need to run Networkwordcount.scala, then I can connect to the local host successfully

3. Then I give an input to the stramer by typing some sentence in netcat cmd

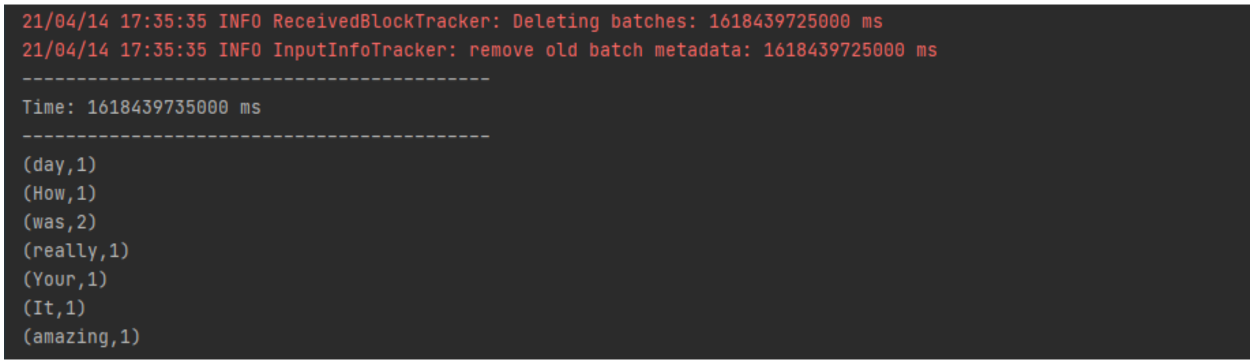


```
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\sjk37>nc -l -p 9999
My name is Jongkook Son I love Scala and spark
How was Your day It was really amazing
I like dog my hobby is take a walk with my dog
```

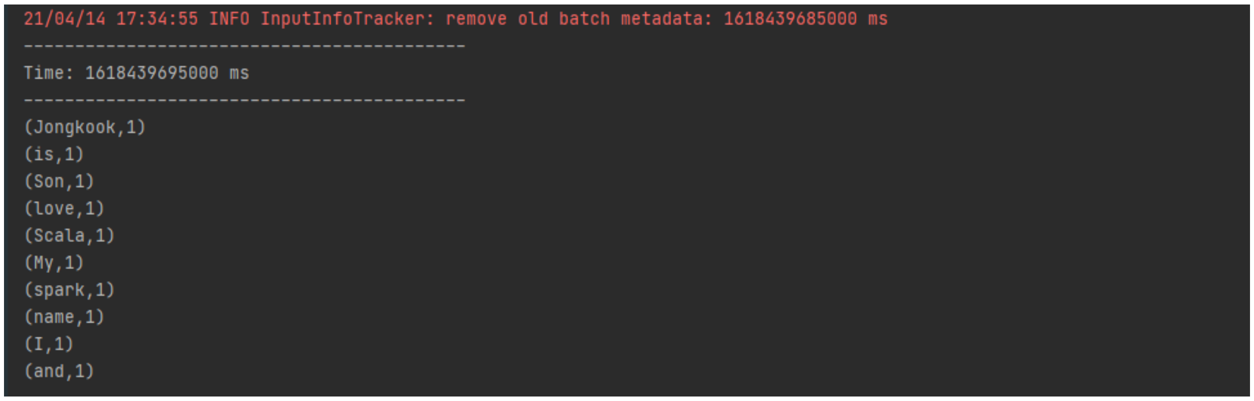


```scala
import org.apache.spark.SparkConf
import org.apache.spark.storage.StorageLevel
import org.apache.spark.streaming.{Seconds, StreamingContext}


object NetworkWordCount {
  def main(args: Array[String]): Unit = {

    // Create the context with a 1 second batch size
    val conf = new SparkConf().setMaster("local[2]").setAppName("NetworkWordCount")
    val ssc = new StreamingContext(conf, Seconds(5))
    val data = ssc.socketTextStream( hostname = "localhost", port = 9999)

//    val lines = ssc.socketTextStream(args(0), args(1).toInt, StorageLevel.MEMORY_AND_DISK_SER)
//    val words = lines.flatMap(_.split(" "))
    val wc = data.flatMap(_.split( regex = " ")).map(x => (x, 1)).reduceByKey(_ + _)
    wc.print()
    ssc.start()
    ssc.awaitTermination()
  }
}
```

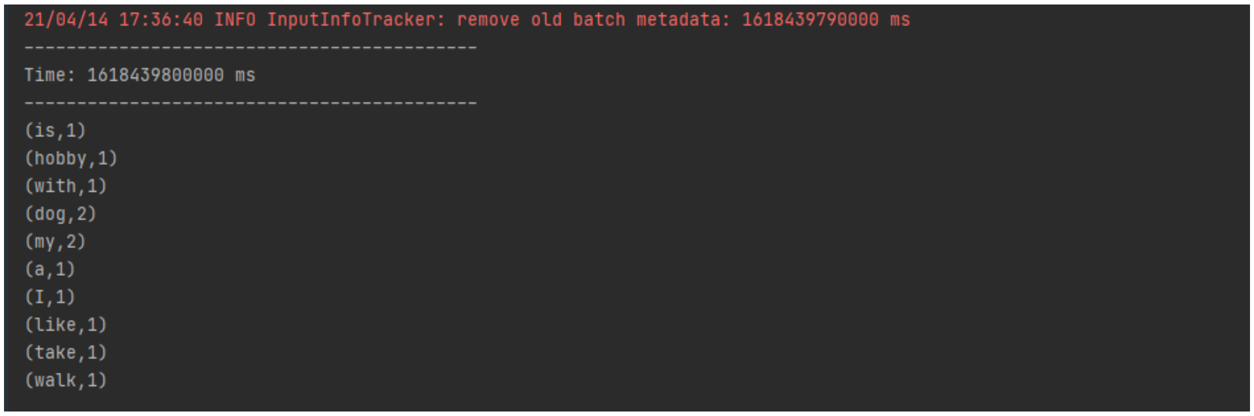
4.Below are the outputs of the input sentences



```
21/04/14 17:35:35 INFO ReceivedBlockTracker: Deleting batches: 1618439725000 ms
21/04/14 17:35:35 INFO InputInfoTracker: remove old batch metadata: 1618439725000 ms
-------------------------------------------
Time: 1618439735000 ms
-------------------------------------------
(day,1)
(How,1)
(was,2)
(really,1)
(Your,1)
(It,1)
(amazing,1)
```



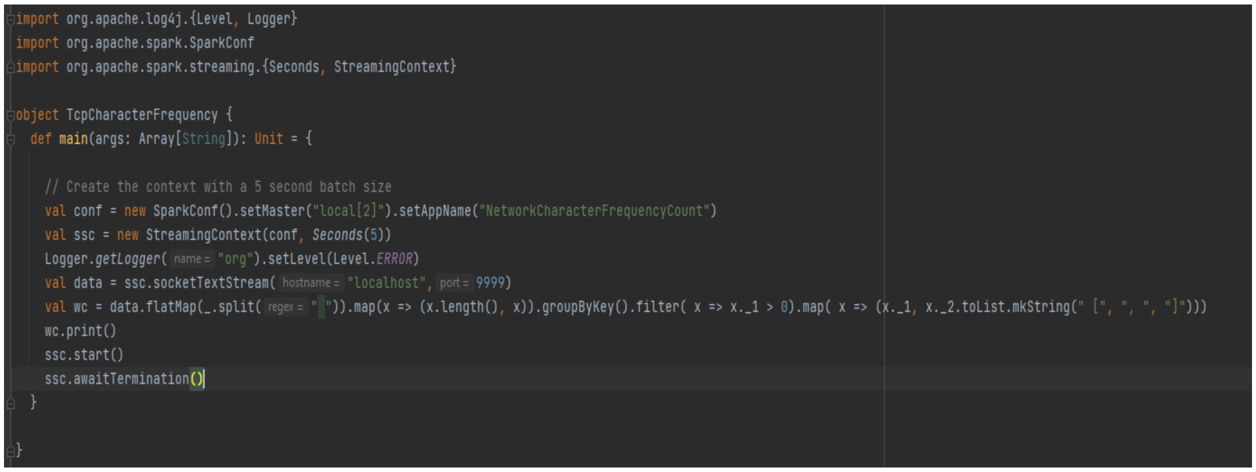
```
21/04/14 17:34:55 INFO InputInfoTracker: remove old batch metadata: 1618439685000 ms
-------------------------------------------
Time: 1618439695000 ms
-------------------------------------------
(Jongkook,1)
(is,1)
(Son,1)
(love,1)
(Scala,1)
(My,1)
(spark,1)
(name,1)
(I,1)
(and,1)
```



```
21/04/14 17:36:40 INFO InputInfoTracker: remove old batch metadata: 1618439790000 ms
-------------------------------------------
Time: 1618439800000 ms
-------------------------------------------
(is,1)
(hobby,1)
(with,1)
(dog,2)
(my,2)
(a,1)
(I,1)
(like,1)
(take,1)
(walk,1)
```

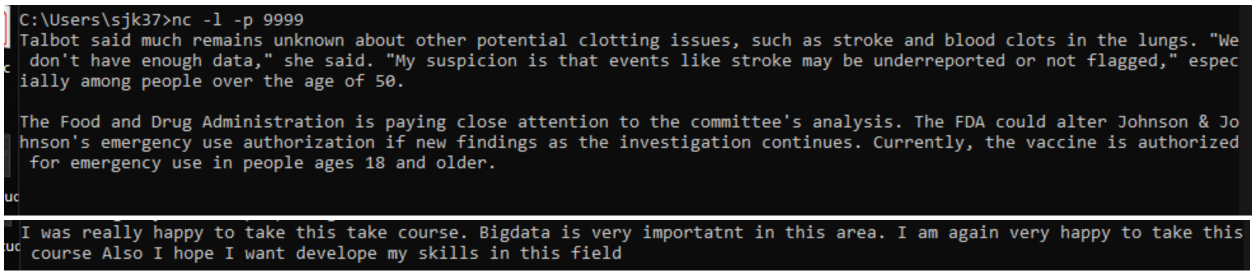
Task3: Spark Streaming for TCP Socket

1.First, I created a TcpCharacter Frequency scala which deals with character frequency of words



```scala
import org.apache.log4j.{Level, Logger}
import org.apache.spark.SparkConf
import org.apache.spark.streaming.{Seconds, StreamingContext}

object TcpCharacterFrequency {
  def main(args: Array[String]): Unit = {

    // Create the context with a 5 second batch size
    val conf = new SparkConf().setMaster("local[2]").setAppName("NetworkCharacterFrequencyCount")
    val ssc = new StreamingContext(conf, Seconds(5))
    Logger.getLogger("org").setLevel(Level.ERROR)
    val data = ssc.socketTextStream("localhost", 9999)
    val wc = data.flatMap(_.split(" ")).map(x => (x.length(), x)).groupByKey().filter( x => x._1 > 0).map( x => (x._1, x._2.toList.mkString(" [", ", ", "]")))
    wc.print()
    ssc.start()
    ssc.awaitTermination()
  }

}
```

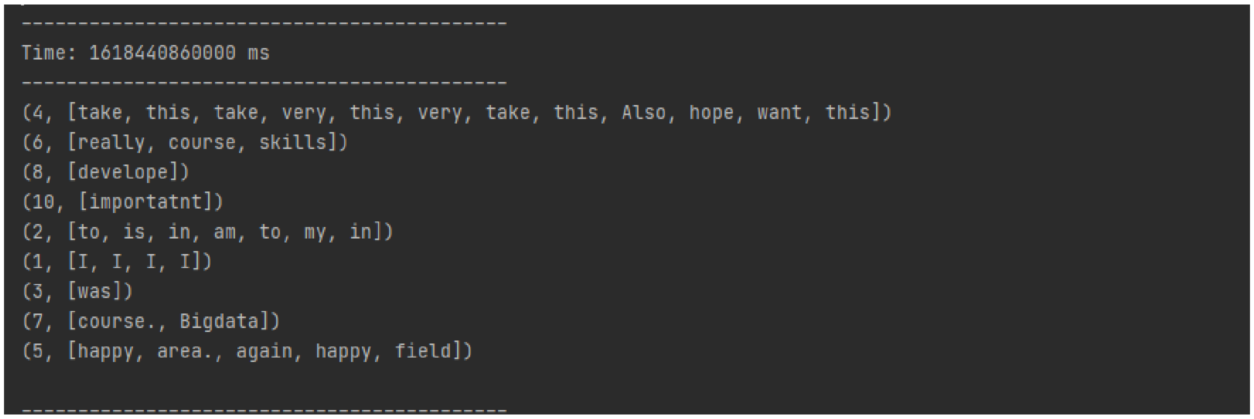
2.Then I typed input through netcat cmd



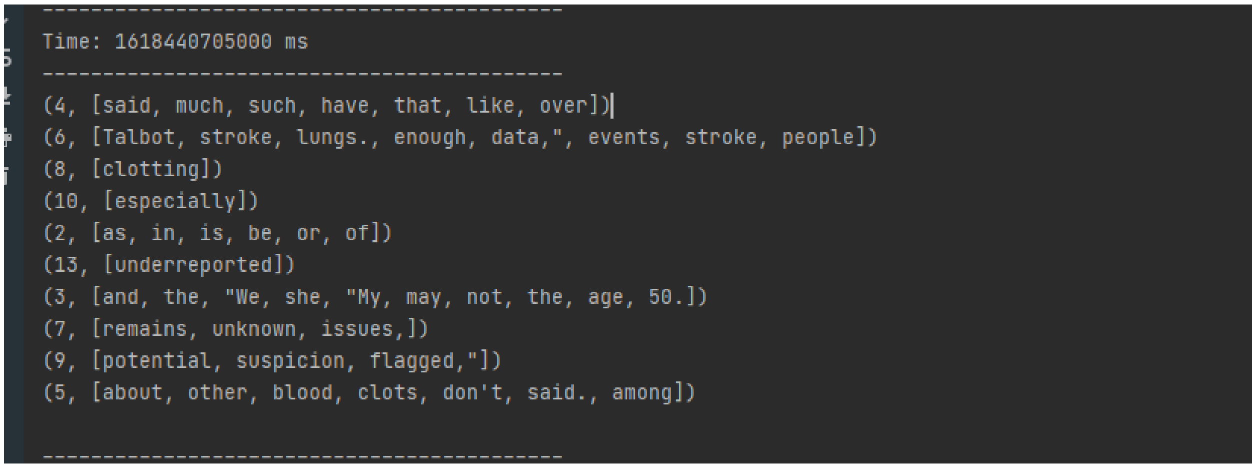
```
C:\Users\sjk37>nc -l -p 9999
Talbot said much remains unknown about other potential clotting issues, such as stroke and blood clots in the lungs. "We
 don't have enough data," she said. "My suspicion is that events like stroke may be underreported or not flagged," espec
ially among people over the age of 50.

The Food and Drug Administration is paying close attention to the committee's analysis. The FDA could alter Johnson & Jo
hnson's emergency use authorization if new findings as the investigation continues. Currently, the vaccine is authorized
 for emergency use in people ages 18 and older.
```

```
I was really happy to take this take course. Bigdata is very importatnt in this area. I am again very happy to take this
 course Also I hope I want develope my skills in this field
```

3.You can find out that the output format is matched with requirements.



```
-------------------------------------------
Time: 1618440860000 ms
-------------------------------------------
(4, [take, this, take, very, this, very, take, this, Also, hope, want, this])
(6, [really, course, skills])
(8, [develope])
(10, [importatnt])
(2, [to, is, in, am, to, my, in])
(1, [I, I, I, I])
(3, [was])
(7, [course., Bigdata])
(5, [happy, area., again, happy, field])

-------------------------------------------
```



```
-------------------------------------------
Time: 1618440705000 ms
-------------------------------------------
(4, [said, much, such, have, that, like, over])
(6, [Talbot, stroke, lungs., enough, data,", events, stroke, people])
(8, [clotting])
(10, [especially])
(2, [as, in, is, be, or, of])
(13, [underreported])
(3, [and, the, "We, she, "My, may, not, the, age, 50.])
(7, [remains, unknown, issues,])
(9, [potential, suspicion, flagged,"])
(5, [about, other, blood, clots, don't, said., among])

-------------------------------------------
```