**CSCI63 Final Project Report: NLTK and Spark Near-Real Time Text Analysis**

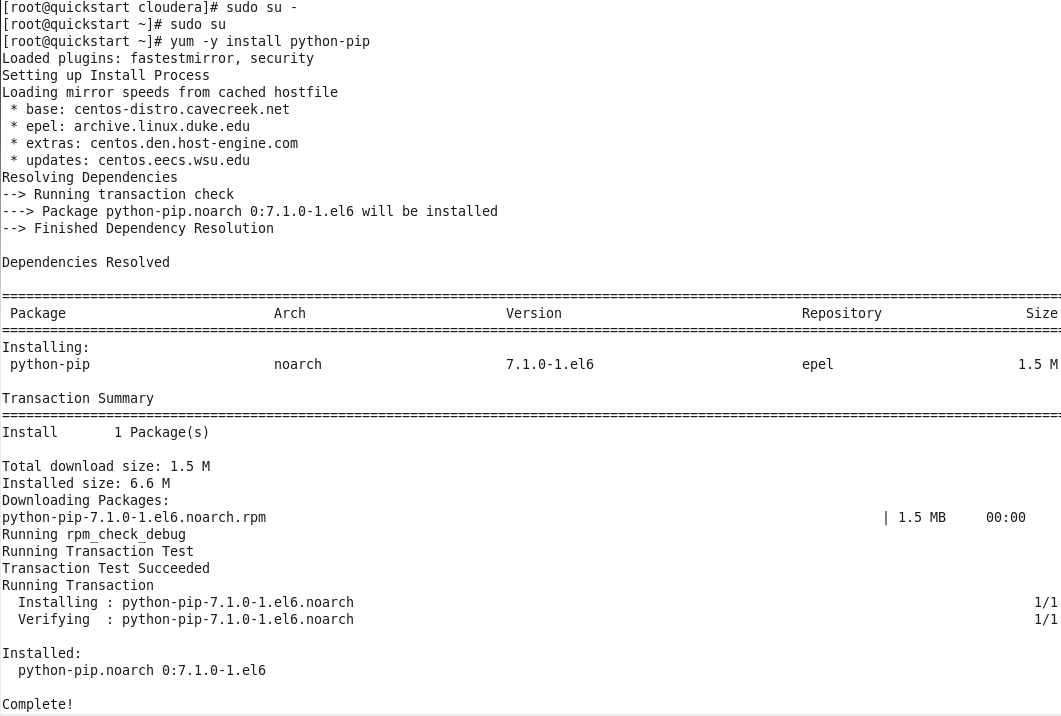
**Spark Installation**

Environment: Cloudera Quickstart CDH 5.8 on Virtualbox (<https://www.cloudera.com/downloads.html>)

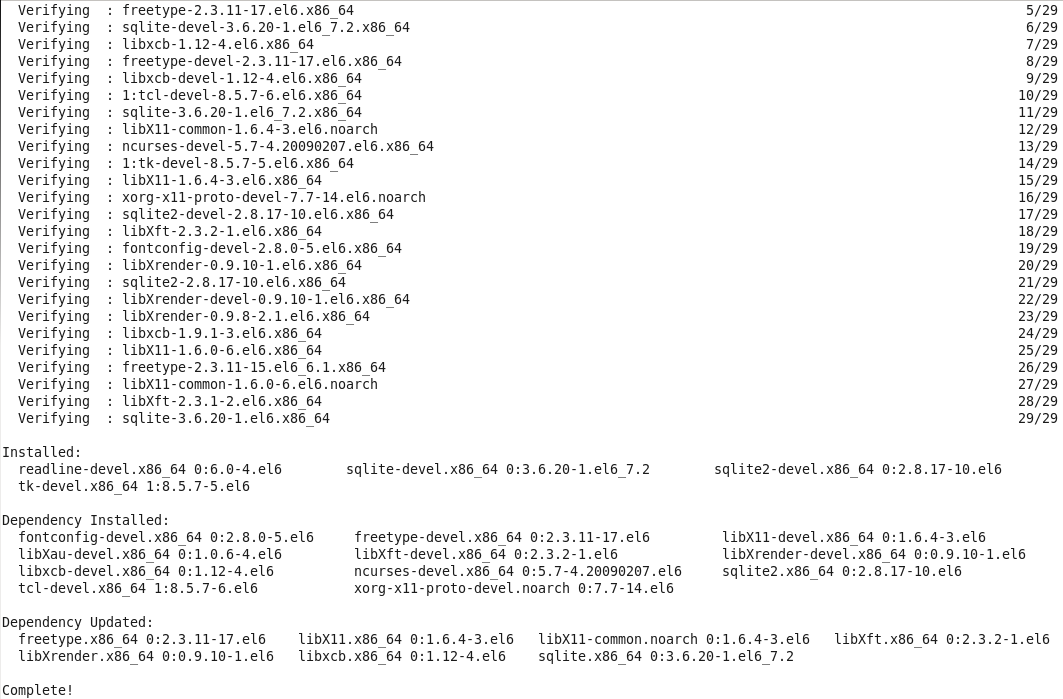
The NLTK 3 library must be installed in order to handle some of the natural language processing we do in cleaning tweet messages. NLTK 3 only supports Python 2.7, not 2.6 which is the default version in Cloudera and PySpark. With the following steps below, we create a virtual environment for Python 2.7 and install NLTK 3, sklearn, pandas, matplotlib, and scipy:

Using terminal in Cloudera, become root (password ‘cloudera’): *sudo su*

Useyum toinstall pip: *yum –y install python-pip*

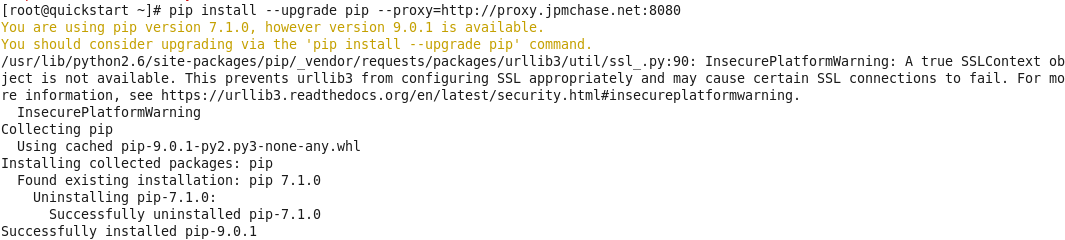


Use yum to install important libraries: *yum –y install readline-devel tk-devel sqlite-devel sqlite2-devel*

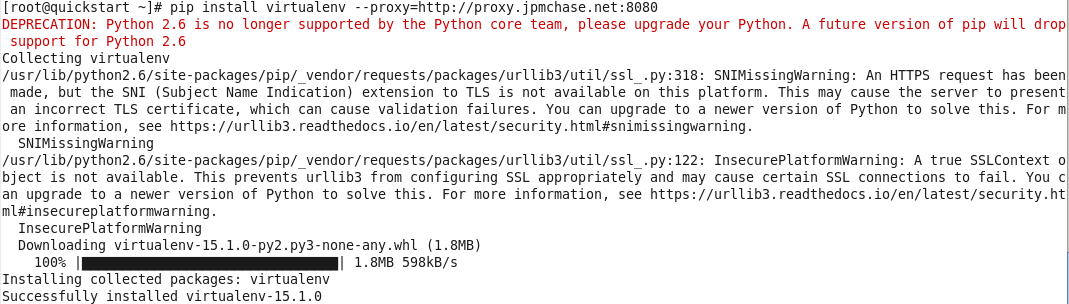


Upgrade pip: *pip install --upgrade pip*

Note: for pip commands, you can add a “--proxy=[PROXY\_SERVER]” parameter if behind proxy server (but otherwise don’t need to)

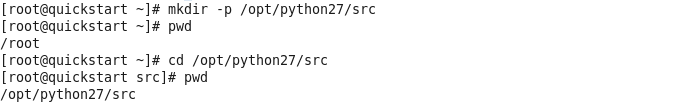


Install virtualenv: *pip install virtualenv*



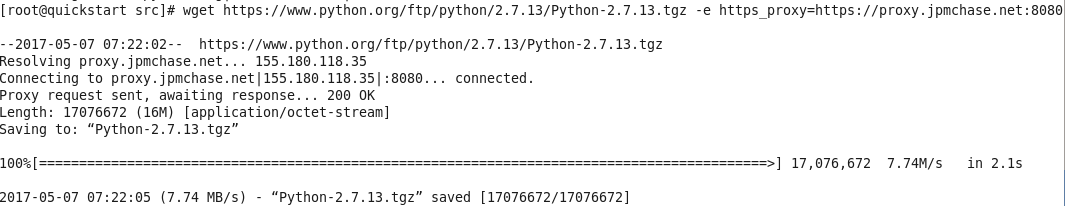
Make directory for Python 2.7: *mkdir –p /opt/python27/src*

Navigate to the new folder: *cd /opt/python27/src*

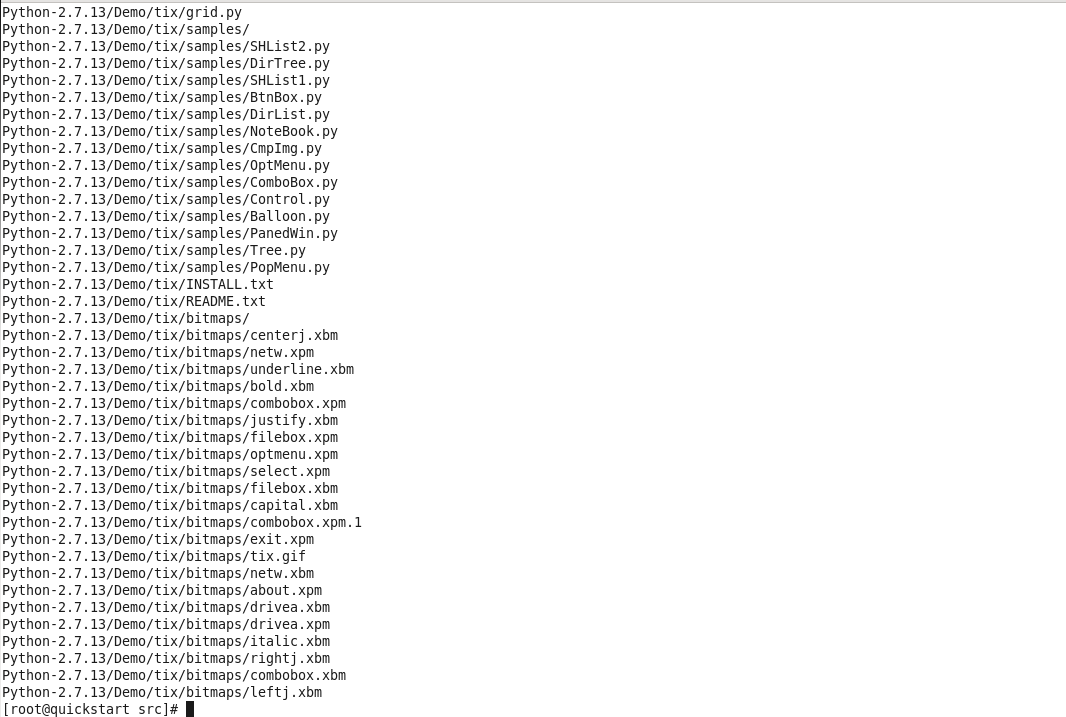


Download python 2.7: *wget https://www.python.org/ftp/python/2.7.13/Python-2.7.13.tgz*

Note: for wget commands, you can add the following parameter to get around proxy servers: “-e https\_proxy=[PROXY\_SERVER]” but again only if this is needed.

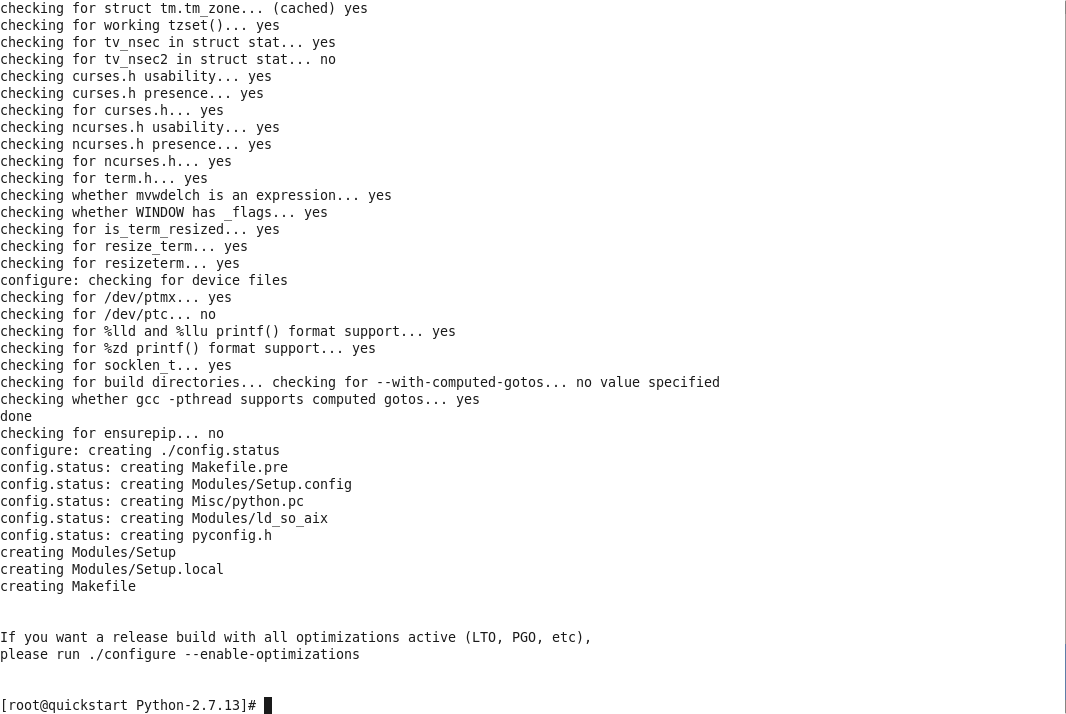


Unzip the file: *tar –xvf Python-2.7.13.tgz*

**

Change directory: *cd Python-2.7.13*

Run configure script: *./configure --prefix=/opt/python27/*

**

Run the make procedure: *make*



Run the make install procedure: *make install*



Exit root session: *exit*

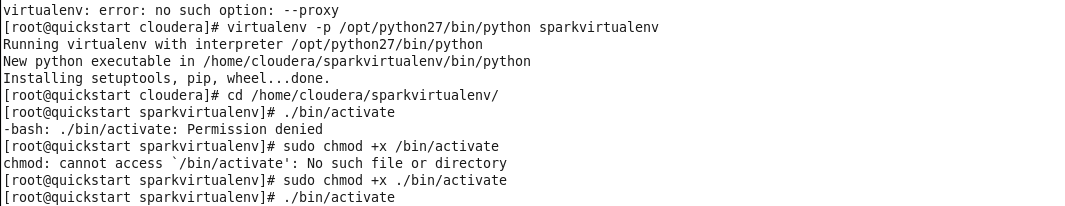
Go to this directory*: cd /home/cloudera/*

Create virtual environment: *virtualenv –p /opt/python27/bin/python sparkvirtualenv*

Change into virtualenv*: cd /home/cloudera/sparkvirtualenv/*

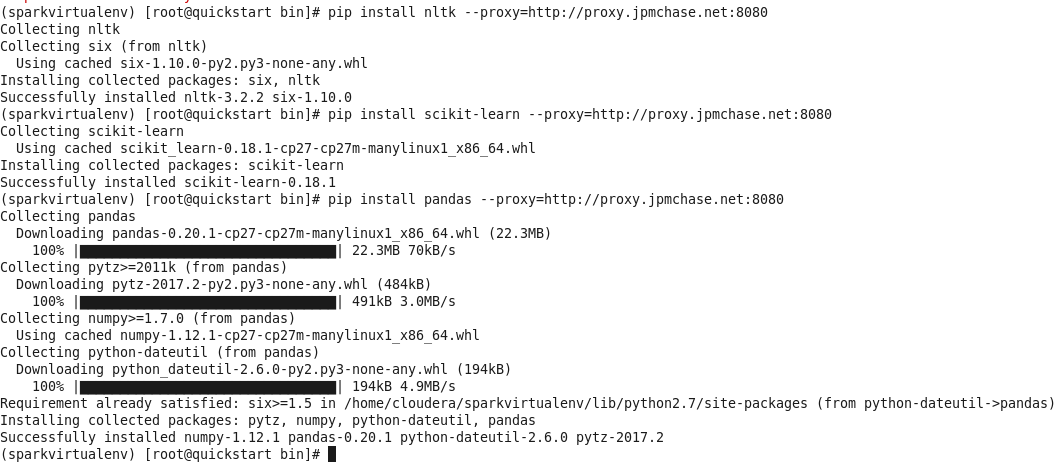
Change permissions to run activate: *sudo chmod +x ./bin/activate*

Activate the virtualenv: *source ./bin/activate*

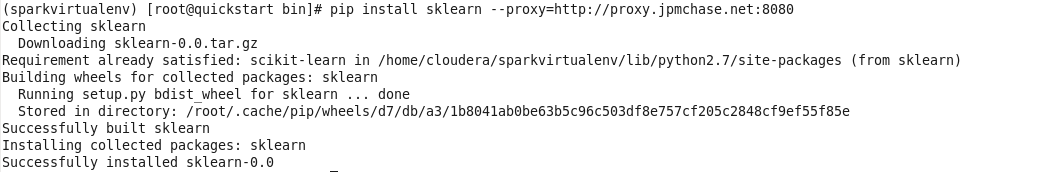
**

*pip install nltk*

*pip install pandas*

**

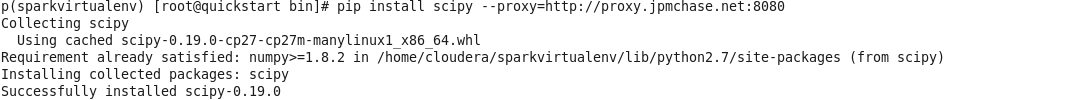
*pip install sklearn*

**

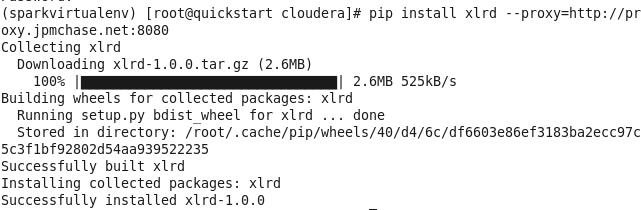
*pip install matplotlib*

**

*pip install scipy*

**

*pip install xlrd*

**

To test if this works, open pyspark and import nltk, sklearn, matplotlib, and pandas libraries. We also download NLTK, which is necessary to use the PerceptronTagger later. Note that the version of Python is the newly installed 2.7.13.

*pyspark*

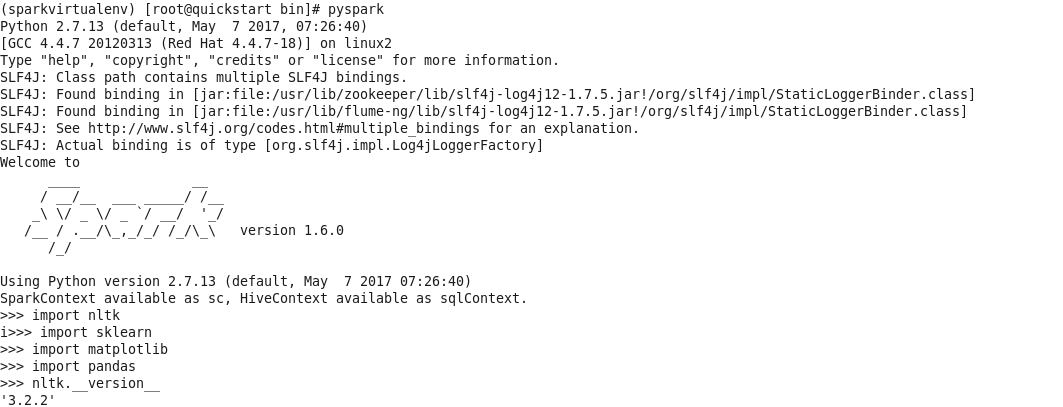
*import nltk*

*import sklearn*

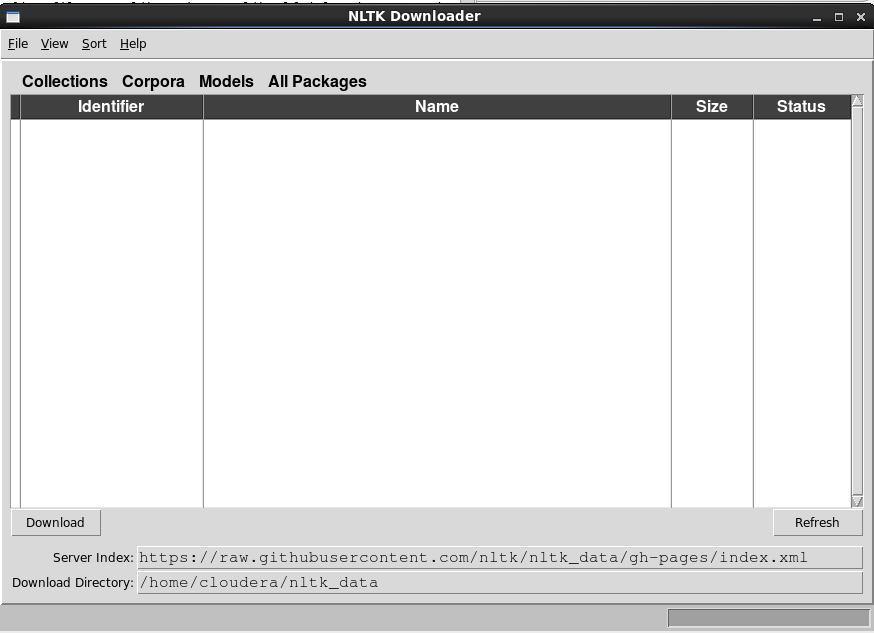
*import matplotlib*

*import pandas*

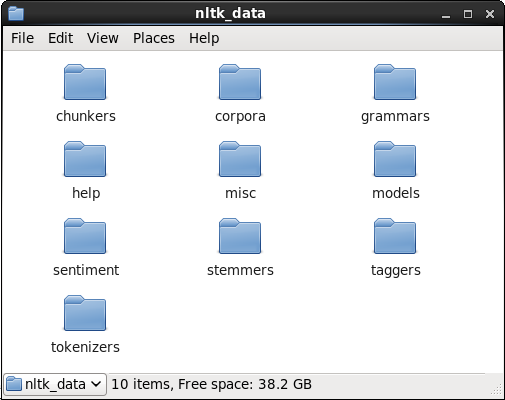
*nltk.\_\_version\_\_*



*nltk.download()*



Most configurations would show a list of items and installation would be complete after installing everything. In our test installation, the proxy server remained an issue, so we took the following steps. The optional command *nltk.set\_proxy(‘[PROXY\_SERVER]’)* allows for a proxy server but it did not work either. We downloaded the zip files from the NLTK github site (<https://github.com/nltk/nltk_data/tree/gh-pages/>) and saved and unzipped the contents of the packages to the folder /home/cloudera/nltk\_data/



This concludes the installation instructions, and in the next section we examine how a stream of tweets can be processed in near real-time.

**Running Spark Streaming**

To illustrate near-real time Spark streaming, we use the provided files stream-dict.py and splitAndSendFinal.sh. For data, we use a csv file which contained all StockTwits downloaded from March 28th, 2017. The file (StockTwits.20170328.csv) is sorted chronologically and does not include any row headers. It is in the same column format as the StockTwits data files we had downloaded. This file was copied over to the Cloudera home directory /home/cloudera/. In addition, I copied over the two dictionary files LoughranMcDonald\_MasterDictionary\_2014.xlsx and inquirerbasic.xls into the Cloudera home directory.

We have taken the file of StockTwits downloaded from March 28th, 2017. We run the following commands to set up the file directory and copy necessary files over:

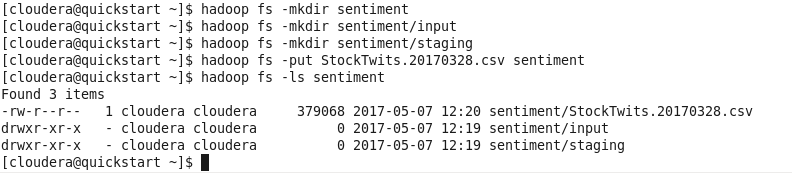
Open a Terminal window and enter the command:

*hadoop fs –mkdir sentiment*

*hadoop fs –mkdir sentiment/input*

*hadoop fs –mkdir sentiment/staging*

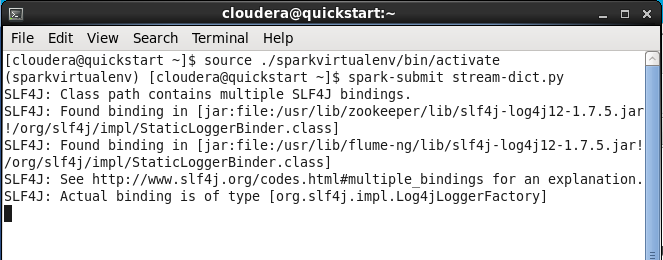
*hadoop fs –put StockTwits.20170328.csv sentiment*

**

In one terminal window, we run the Spark stream listener which runs the sentiment analysis using the Harvard dictionary on a Bag-Of-Words. We can run the process in a new terminal window:

*source ./sparkvirtualenv/bin/activate*

*spark-submit stream-dict.py*

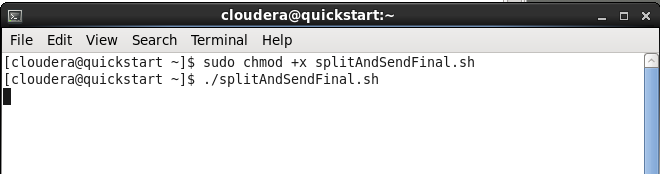
**

In a different terminal, we need to change permissioning to run the splitAndSendFinal script and run it:

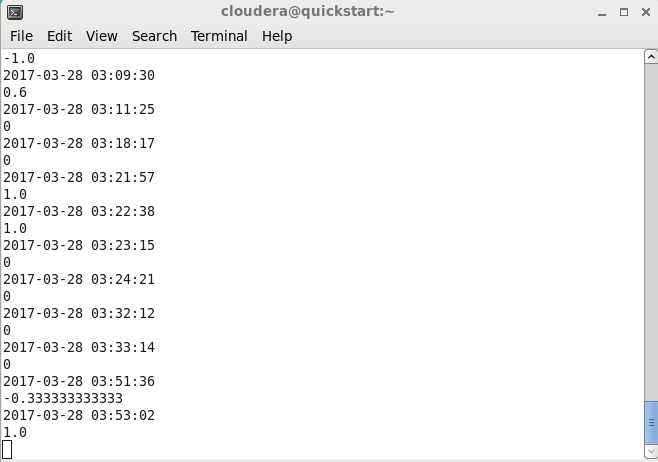
*source ./sparkvirtualenv/bin/activate*

*sudo chmod +x splitAndSendFinal.sh*

*./splitAndSendFinal.sh*

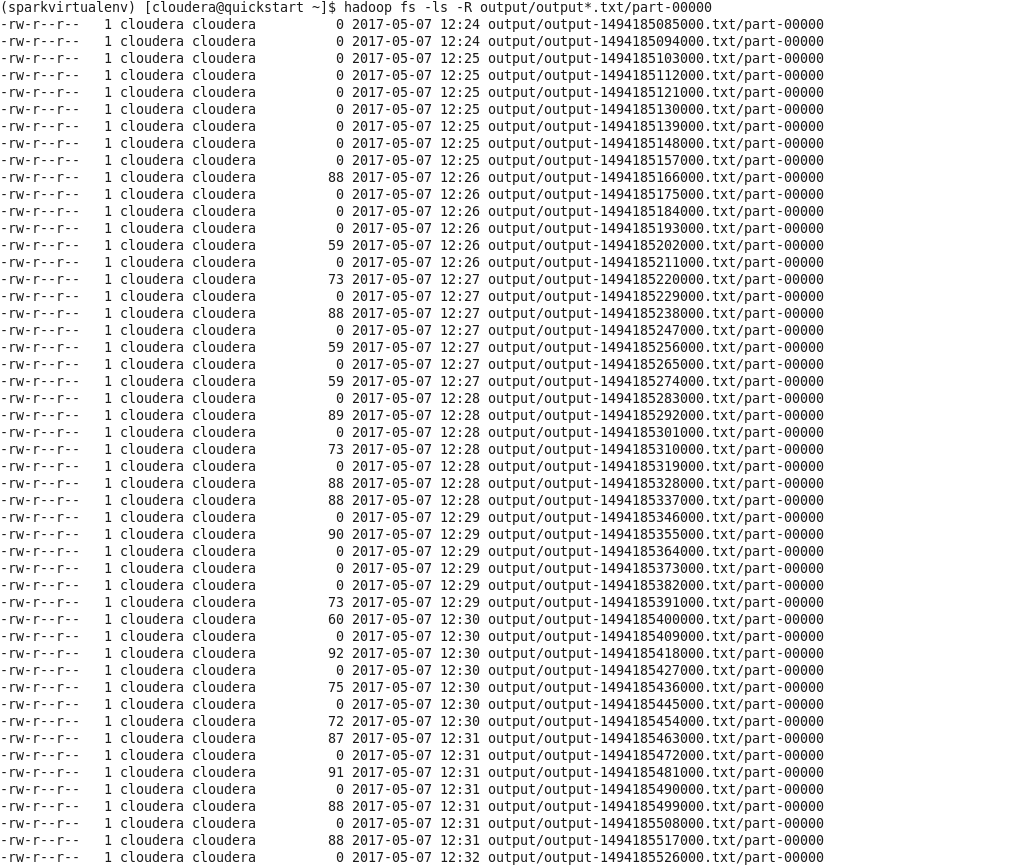
**

The splitAndSendFinal file divides tweets into clusters of 100 tweets, and pauses 10 seconds before sending another batch to the listener.



Our listener (stream-dict.py) provides some intermediate information in the terminal by displaying the immediate score for each single tweet. There were 2,791 tweets for the three stocks on March 28th, the sender streamed data through in 28 chunks, and everything finished in roughly 5 minutes.

To view the result files: *hadoop fs –ls –R output/output\*.txt/part-00000*

**

To view individual results:

*hadoop fs -cat output/output-1494185517000.txt/part-00000*

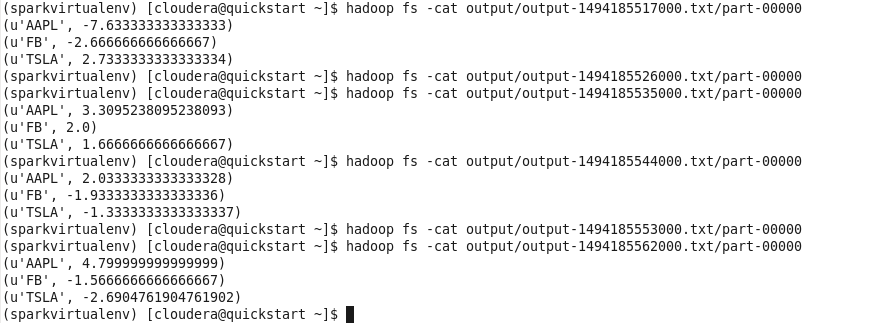
*hadoop fs -cat output/output-1494185526000.txt/part-00000*

*hadoop fs -cat output/output-1494185535000.txt/part-00000*

*hadoop fs -cat output/output-1494185544000.txt/part-00000*

*hadoop fs -cat output/output-1494185553000.txt/part-00000*

*hadoop fs -cat output/output-1494185562000.txt/part-00000*

**

The final output provides a total sentiment score for each stock during that time interval it was received. In real-time streaming, we calculate the raw sum of sentiment scores rather than total sentiment score divided by the number of tweets, because the distribution of tweets is fairly random. Also, notice that some of the output files were empty. This is due to the 10-second pause in between simulated tweets being broadcast.

To clear files in output directory: *hadoop fs –rm output/output\*.txt/\**

To clear directories: *hadoop fs –rmdir output/\**