Twitter Sentiment Analysis using Syncfusion Data Platform

In this blog post, we can see how to get the insights from Twitter feeds using Apache Spark and populate the data into a Dashboard. We are going to perform sentiment analysis over a subset of a Twitter stream using Stanford CoreNLP.

Sentiment analysis is commonly used to *discover the sentiment* of a brand, product, or any topic. Like,

* How are customers responding to our brand / new product release?
* Is a product review positive or negative?

‘Stanford NLP’ used in this blog post provides a set of human language processing tools available for languages like English, Arabic, Chinese, French, German, and Spanish. To know more about it, refer [here](https://stanfordnlp.github.io/CoreNLP/).

We use Apache spark shipped in ‘Syncfusion Big Data Platform’ for processing data and ‘Syncfusion Dashboard Platform’ for final visualization.

Note: We used streaming support available in Spark to fetch tweets at real time, to know more about it refer [here](http://spark.apache.org/docs/latest/streaming-programming-guide.html).

# Stanford NLP

Stanford CoreNLP deep learning model builds up a new type of Recursive Neural Network on top of grammatical structures of given whole sentences and computes the sentiment on the meaning of longer phrases. Its API provides a document processing pipeline in terms of annotators, which extract the words, phrases and perform different linguistic analysis based on the input parameters. For our analysis, we used below annotators.

|  |  |
| --- | --- |
| *tokenize* | Divides text into a sequence of words |
| *ssplit* | Split the text into sentence. Identify full stop, exclamation etc. and split sentences |
| *pos* | Reads the text and assigns parts of speech to each word such as noun, verb, adjective, etc. Ex. "This is a simple sentence" will be tagged as "This/DT is/VBZ a/DT sample/NN sentence/NN" |
| *lemma* | Forms of a word are grouped together so that they can be analyzed as a single item. |
| *parse* | Provides syntactic analysis. |
| *sentiment* | Provides model for sentiment analysis |

val nlpPropts = {

val propts = new Properties()

propts.setProperty("annotators", "tokenize, ssplit, pos, lemma, parse, sentiment")

propts

}

// Create a pipeline with NLP properties

val pipeline = new StanfordCoreNLP(nlpPropts)

// Run sample text through the Pipeline to process against the annotators

val annotation = pipeline.process("This is a sample text, you can provide actual text input here")

Here from “annotation” pipeline object, we need to extract the sentence annotation first followed by its sentiment annotated tree to get the predicted score. We use below code snippet to get sentiment score.

// An Annotation is a Map and you can get and use the various analyses individually.

// For instance, this gets the parse tree of the first sentence in the text.

val sentenceAnnotation = annotation.get(classOf[CoreAnnotations.SentencesAnnotation])

// Create a RNN parse tree

val sentimentAnnotatedTree = sentenceAnnotation.get(classOf[SentimentCoreAnnotations.AnnotatedTree])

// Detect Sentiment

val sentimentScore = RNNCoreAnnotations.getPredictedClass(sentimentAnnotatedTree)

Based on the sentiment score, we can classify the sentiment of the given text to “POSITIVE, NEGATIVE, NEUTRAL, Not Understood” based on the ranges.

if (sentimentScore <= 0.0)

println ("NOT\_UNDERSTOOD")

else if (sentimentScore < 1.6)

println (""NEGATIVE")

else if (sentimentScore <= 2.0)

println ("NEUTRAL")

else if (sentimentScore < 5.0)

println (""POSITIVE")

else println (""NOT\_UNDERSTOOD")

# Play time!

## Setting up Environment

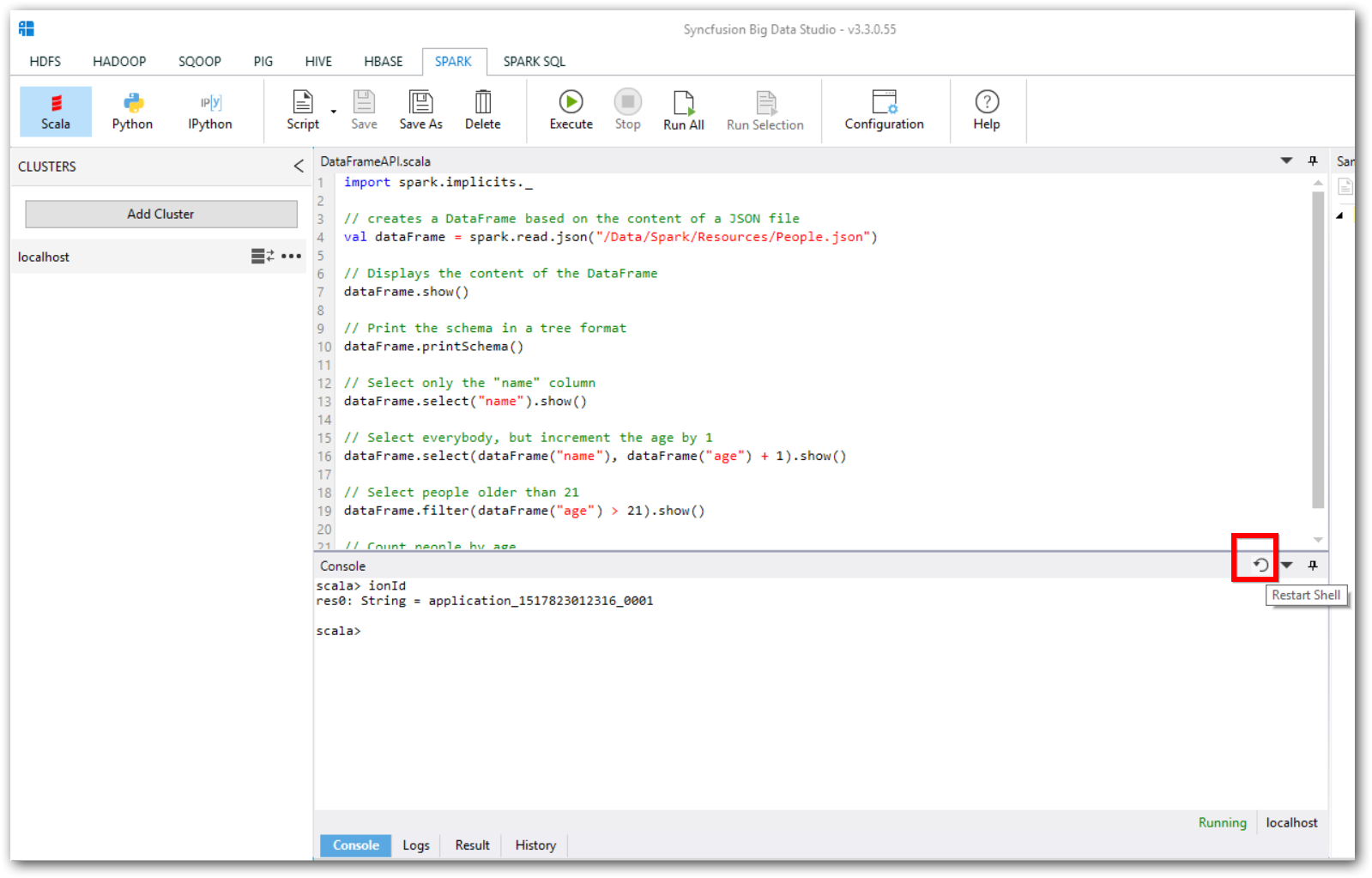
Install Syncfusion Big Data Studio and Syncfusion Dashboard products in your system. To know more about our products, check below links:

|  |  |
| --- | --- |
| Syncfusion Big Data Studio | <https://www.syncfusion.com/products/big-data>  <https://help.syncfusion.com/bigdata/bigdata-studio/overview#getting-started> |
| Syncfusion Dashboard | <https://www.syncfusion.com/products/dashboard>  <https://help.syncfusion.com/dashboard-platform/overview> |

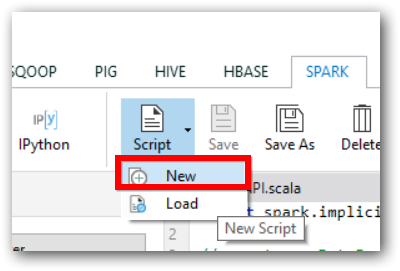
## Getting tweets

Let us see how simple it is to get current trending tweets based on the keyword / brandname / product using Spark

1. Download the following jar files,
   1. Spark-Twitter – [Download Link](https://jar-download.com/explore-java-source-code.php?a=spark-streaming-twitter_2.11&g=org.apache.bahir&v=2.0.1&downloadable=1)
   2. Stanford-CoreNLP – [Download Link](https://nlp.stanford.edu/software/stanford-corenlp-full-2014-08-27.zip)
   3. OpenNLP – [Download Link](https://rsc.workfusion.com/nexus/content/repositories/ml-sdk/org/apache/opennlp/opennlp-tools/1.6.0/opennlp-tools-1.6.0.jar)
2. From the downloads, copy the following jars to “C:\Syncfusion\BigData\<version>\BigDataSDK\SDK\Spark\jars”;
   1. stanford-corenlp-3.4.1.jar
   2. stanford-corenlp-3.4.1-models.jar
   3. ejml-0.23.jar
   4. opennlp-tools-1.6.0.jar
   5. twitter4j-core-4.0.4.jar
   6. twitter4j-stream-4.0.4.jar
   7. spark-streaming-twitter\_2.11-2.0.1.jar
3. Open Big Data Studio [Spark](https://help.syncfusion.com/bigdata/bigdata-studio/spark) tab and restart Spark-Scala shell so that the shell takes effect of all the newly added jars. Spark context is already created for you when the shell is started.



1. Create a new empty script.



1. Copy below code snippet and paste them in the text editor under Big Data Studio Spark tab. Below code is based on spark streaming support used to fetch twitter tweets at a real time and move the JSON data from Twitter into given HDFS location.

It requires Consumer Key, Consumer Secret, Acess token, Access Token Secret. You can generate them by following this [reference](https://themepacific.com/how-to-generate-api-key-consumer-token-access-key-for-twitter-oauth/994/).

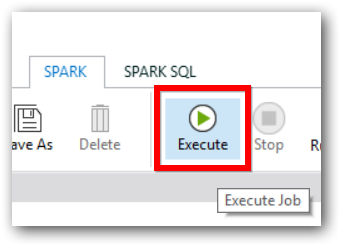
Note: Mention your brand name or Keyword based on which you want to filter tweets in the Variable ‘Key’

We have created a Spark Streaming context **ssc** using Spark context **sc**

|  |
| --- |
| import org.apache.spark.SparkConf  import twitter4j.conf.ConfigurationBuilder  import twitter4j.auth.OAuthAuthorization  import twitter4j.Status  import twitter4j.\_  import com.google.gson.Gson  import org.apache.spark.SparkContext  import org.apache.spark.streaming.{Seconds, StreamingContext}  import org.apache.spark.streaming.dstream.DStream  import org.apache.spark.streaming.twitter.TwitterUtils  //Set the following parameters  var ConsumerKey="consumer key"  var ConsumerSecretKey="consumer secret key"  var AccessToken="access token"  var AccessTokenSecret="access token secret"  var key="brandname" // Set your filter keyword to collect tweets.    // Set the Spark StreamingContext to create a DStream for every 5 seconds  val ssc = new StreamingContext(sc, Seconds(5))  // Pass the filter keywords as arguements  val cb = new ConfigurationBuilder  cb.setDebugEnabled(true).setOAuthConsumerKey(ConsumerKey).setOAuthConsumerSecret(ConsumerSecretKey).setOAuthAccessToken(AccessToken).setOAuthAccessTokenSecret(AccessTokenSecret)  val auth = new OAuthAuthorization(cb.build)  var filters = Array(key)  val tweetStream: DStream[String] = TwitterUtils.createStream(ssc, Some(auth),filters).map(new Gson().toJson(\_))  var numTweetsCollected = 0L  tweetStream.foreachRDD { (rdd, time) =>  rdd.saveAsTextFile("/Data/TwitterSparkStreaming/"+key+"/")  }  ssc.start()  ssc.awaitTerminationOrTimeout(60000) // Run this for till the specified time  ssc.stop() |

1. Execute the script.

The spark context will be in running state until the timeout specified(60000) and the tweets will be retrieved at this time. The pulled tweets will get stored into HDFS location "/Data/TwitterSparkStreaming/BrandName" in JSON format.



1. **You need to restart the spark shell now, as the spark context will be terminated after the specified timeout.**

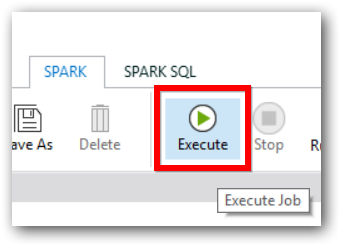
## Analyzing tweets

1. Create another new script file and copy the following into it. This code is to analyze the tweets using NLP. See to that you specify the same key which is specified in previous code snippet in variable ‘key’.

The data will be parsed per tweet and sent to ‘detectSentiment’ function. Finally, the sentiment is added as an additional column along with the actual data and created as a Spark table.

|  |
| --- |
| import java.util.Properties  import java.util.Properties  import org.apache.spark.sql.SQLContext  import edu.stanford.nlp.ling.CoreAnnotations  import edu.stanford.nlp.neural.rnn.RNNCoreAnnotations  import edu.stanford.nlp.pipeline.StanfordCoreNLP  import edu.stanford.nlp.sentiment.SentimentCoreAnnotations  import scala.collection.mutable.ArrayBuffer  import scala.collection.JavaConversions.\_  import scala.collection.mutable.ListBuffer  import sqlContext.implicits.\_  val nlpPropts = {  val propts = new Properties()  propts.setProperty("annotators", "tokenize, ssplit, pos, lemma, parse, sentiment")  propts  }  def detectSentiment(message: String): String = {  val pipeline = new StanfordCoreNLP(nlpPropts)  val annotation = pipeline.process(message)  var sentiments: ListBuffer[Double] = ListBuffer()  var sizes: ListBuffer[Int] = ListBuffer()  var longest = 0  var mainSentiment = 0  for (tweetMsg <- annotation.get(classOf[CoreAnnotations.SentencesAnnotation])) {  val parseTree = tweetMsg.get(classOf[SentimentCoreAnnotations.AnnotatedTree])  val tweetSentiment = RNNCoreAnnotations.getPredictedClass(parseTree)  val partText = tweetMsg.toString  if (partText.length() > longest) {  mainSentiment = tweetSentiment  longest = partText.length()  }  sentiments += tweetSentiment.toDouble  sizes += partText.length  }  val weightedSentiments = (sentiments, sizes).zipped.map((sentiment, size) => sentiment \* size)  var weightedSentiment = weightedSentiments.sum / (sizes.fold(0)(\_ + \_))  if (weightedSentiment <= 0.0)  "NOT\_UNDERSTOOD"  else if (weightedSentiment < 1.6)  "NEGATIVE"  else if (weightedSentiment <= 2.0)  "NEUTRAL"  else if (weightedSentiment < 5.0)  "POSITIVE"  else "NOT\_UNDERSTOOD"  }  val sqlContext = new org.apache.spark.sql.SQLContext(sc)  var key = "brandname"  val branddatadf = sqlContext.read.json("/Data/TwitterSparkStreaming/"+key)  var branddata=branddatadf.withColumn("brand",lit(key))  var dataarray =branddata.collect().map(t => (detectSentiment(t.getAs[String]("text")).toString,t.getAs[Long]("id")))  val rdd = sc.parallelize(dataarray)  var twitterdataframe= rdd.toDF()  var newdf = branddata.join(twitterdataframe, branddata.col("id") === twitterdataframe.col("\_2"))  var result=newdf.select("id","createdAt","isFavorited","retweetCount","favoriteCount","text","user.name","user.screenName","user.friendsCount","user.followersCount","user.statusesCount","lang","brand","\_1")  result.write.mode(org.apache.spark.sql.SaveMode.Append).saveAsTable("TwitterAnalysis") |

1. Execute the script. Sentimental analysis of twitter feeds started and Sentiment scores are defined in this sample as ‘positive’, ‘negative’ and ‘neutral’.



1. Once the Spark job execution completed, the result will be stored in a Spark Table **‘TwitterAnalysis’** which we are going to be visualized in Dashboard**.**

**Note:** You can generate sentiments of multiple brands in same Spark table ‘TwitterAnalysis’ appended to it. Just change the brand name in both scripts, restart the shell and repeat to run them.

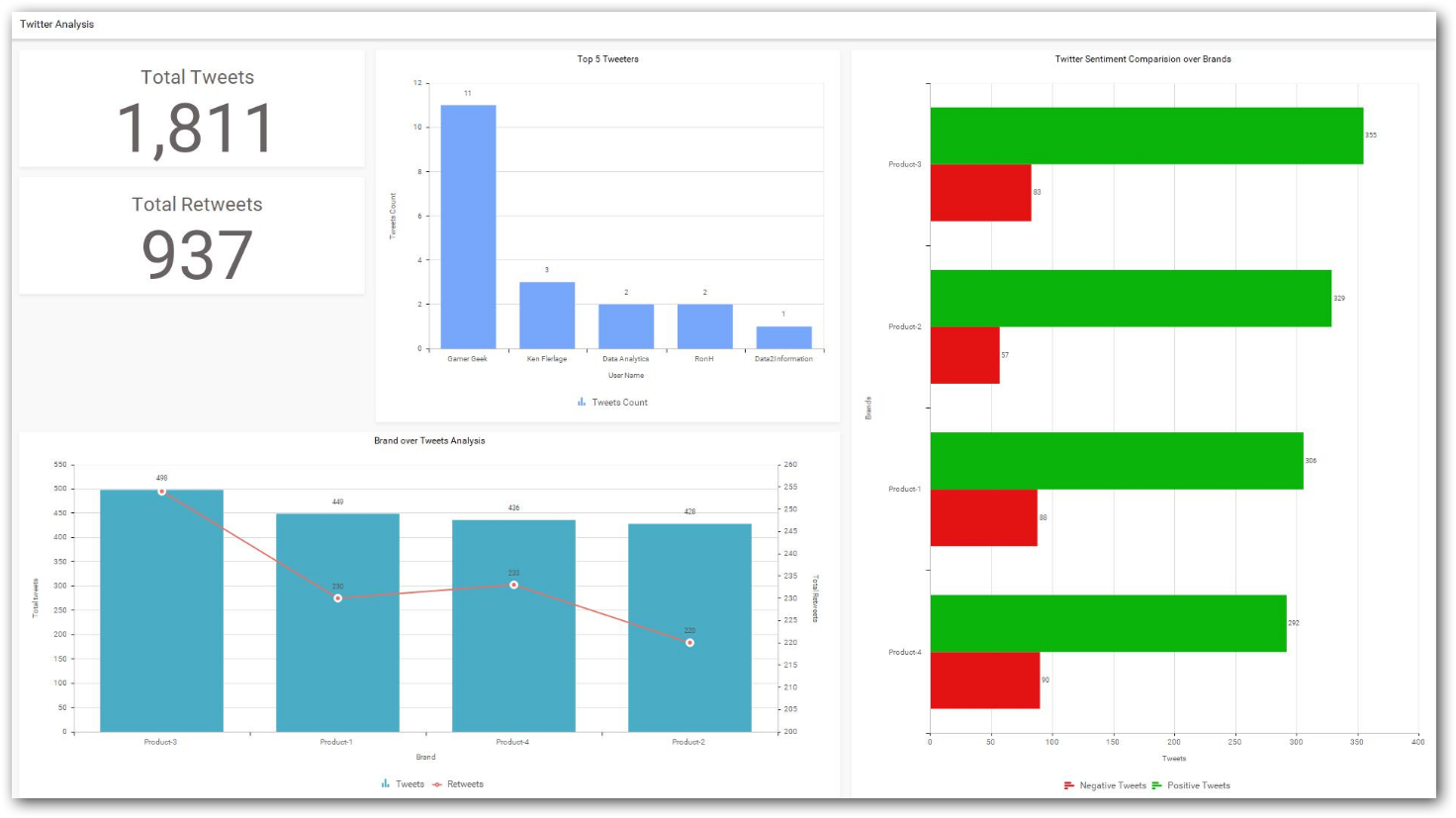
# Visualize result in Dashboard

Visualization is a key thing to showcase the data in a far better way. Most of the people especially from the management sector use this for decision making*.*

1. We’re going to use Syncfusion Dashboard Platform for Visualizing data from Spark. Have a quick start about this,

<https://help.syncfusion.com/dashboard-platform/quick-start>

1. Download the [Dashboard Designer .sydx file](sydx%20file%20from%20syncfusion%20repository) from here and launch the file which shows the pre-built widgets in Dashboard Designer.
2. Click ‘Preview’ to view the result from the Spark table.



**Note:** The Dashboard is designed in such a way as the cluster is running as ‘localhost’ instance. If you have a cluster with IP address, please edit the following connection string with Namenode IP.

<<Connection String>>

Well, it went good I hope!

We have now got twitter tweets via Spark streaming, did sentimental analysis using NLP and finally visualized the result in a Dashboard.

You can prepare the same environment such that you get near real-time result about your brand immediately. We like helping people who need any kind of Big Data / Data stream / Dashboard solution like this. Feel free to [contact us](https://www.syncfusion.com/company/contact-us) if you need help.

# References

<https://stanfordnlp.github.io/CoreNLP/>

<https://spark.apache.org/streaming/>

<https://www.oreilly.com/ideas/training-and-serving-nlp-models-using-spark-mllib>

<https://www.safaribooksonline.com/library/view/apache-spark-for/9781785880100/ch06.html>