**[SRC] 04.1. Stream Tweets to Disk**

System.setProperty("twitter4j.oauth.consumerKey", "CJ63hFzuu3KLovHt8LmFMw")

System.setProperty("twitter4j.oauth.consumerSecret", "2W373bWM1Oca9zyDk38qeUoVamdJ78ZMfexq6mQsLb8")

System.setProperty("twitter4j.oauth.accessToken", "101229446-5qSO2hRsVfjO3nhopQGFT4FjPpb2XmBGDAHkxBg")

System.setProperty("twitter4j.oauth.accessTokenSecret", "D7TDqjm1TBpjF1Gs6OmIhWDC13WQJrLM86SoSznH8SQ")

val mapper = new com.fasterxml.jackson.databind.ObjectMapper

import org.apache.spark.streaming.\_

import org.apache.spark.streaming.twitter.\_

import java.util.concurrent.TimeUnit

import com.google.common.io.Files

val ssc = new StreamingContext(sc, Seconds(2))

val tweetStream = TwitterUtils.createStream(ssc, None)

val dstream = tweetStream.window(Seconds(6))

var collect = Array[String]()

dstream

.foreachRDD(rdd => {

collect ++= rdd.collect.map(t => mapper.writeValueAsString(t))

})

ssc.start

ssc.awaitTerminationOrTimeout(TimeUnit.SECONDS.toMillis(60))

val tempFolder = Files.createTempDir().getAbsolutePath + "/tweets"

println(s"Saving Tweets to folder $tempFolder")

sc.parallelize(collect).saveAsTextFile(tempFolder)

collect.take(5).map(println)

ssc.stop(true, true)

%sh

ls /tmp/1475931734044-0/tweets

%sh

head /tmp/1475931734044-0/tweets/part-00000

**[SRC] 04.2. Stream Tweets on a Map**

%angular

<h3>Hello {{name}}</h3>

<button class="btn btn-success" ng-click="name='Reader'">click</button>

z.angularBind("name", "Packt DSS")

z.angular("name")

var changed = 0

def aWatcher(before:Object, after:Object) = {

changed += 1

// println(changed)

}

z.angularWatch("name", aWatcher \_)

println("How many times the button has been triggered? " + changed + " times!")

import org.apache.spark.streaming.\_

import org.apache.spark.streaming.twitter.\_

import org.apache.spark.storage.StorageLevel

System.setProperty("twitter4j.oauth.consumerKey", "CJ63hFzuu3KLovHt8LmFMw")

System.setProperty("twitter4j.oauth.consumerSecret", "2W373bWM1Oca9zyDk38qeUoVamdJ78ZMfexq6mQsLb8")

System.setProperty("twitter4j.oauth.accessToken", "101229446-5qSO2hRsVfjO3nhopQGFT4FjPpb2XmBGDAHkxBg")

System.setProperty("twitter4j.oauth.accessTokenSecret", "D7TDqjm1TBpjF1Gs6OmIhWDC13WQJrLM86SoSznH8SQ")

val ssc = new StreamingContext(sc, Seconds(2))

val tweets = TwitterUtils.createStream(ssc, None).filter(\_.getGeoLocation() != null)

val twt = tweets.window(Seconds(6))

case class Tweet(createdAt:Long, text:String, longitude: Double, latitude: Double)

twt

.map(status => Tweet(status.getCreatedAt().getTime()/1000, status.getText(), status.getGeoLocation().getLongitude(), status.getGeoLocation().getLatitude()))

.foreachRDD(rdd => {

rdd.toDF().createOrReplaceTempView("tweets")

z.angularBind("locations", rdd.collect())

})

ssc.start()

%sql select \* from tweets limit 10

%sql select longitude, latitude from tweets limit 10

%angular

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.5/leaflet.css" />

<div id="map" style="height: 800px; width: 100%"></div>

<script type="text/javascript">

function initMap() {

var map = L.map('map').setView([30.00, -30.00], 3);

L.tileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

attribution: 'Map data &copy; <a href="http://openstreetmap.org">OpenStreetMap</a> contributors',

maxZoom: 12,

minZoom: 3

}).addTo(map);

var geoMarkers = L.layerGroup().addTo(map);

var el = angular.element($('#map').parent('.ng-scope'));

angular.element(el).ready(function() {

window.locationWatcher = el.scope().compiledScope.$watch('locations', function(newValue, oldValue) {

// geoMarkers.clearLayers(); -- if you want to only show new data clear the layer first

angular.forEach(newValue, function(tweet) {

var marker = L.marker([tweet.latitude, tweet.longitude])

.bindPopup(tweet.text)

.addTo(geoMarkers);

});

})

});

}

if (window.locationWatcher) {

// clear existing watcher otherwise we'll have duplicates

window.locationWatcher();

}

// ensure we only load the script once, seems to cause issues otherwise

if (window.L) {

initMap();

} else {

console.log('Loading Leaflet library');

var sc = document.createElement('script');

sc.type = 'text/javascript';

sc.src = 'https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.5/leaflet.js';

sc.onload = initMap;

sc.onerror = function(err) { alert(err); }

document.getElementsByTagName('head')[0].appendChild(sc);

}

</script>

**[SRC] 04.3. Cleanse and Build your Reference Dataset**

val r1 = sc.textFile("/tmp/1476122508344-0/tweets")

r1.count

val r2 = sc.textFile("/tmp/1476127831347-0/tweets")

r2.count

val r3 = sc.textFile("/tmp/1476127902785-0/tweets")

r3.count

import com.google.common.io.Files

val tweetsFolder = Files.createTempDir().getAbsolutePath + "/tweets"

val t = r1.union(r2).union(r3)

t.count

t.saveAsTextFile(tweetsFolder)

val dt = spark.read.json(tweetsFolder)

dt.printSchema

import org.apache.tika.language.LanguageIdentifier

def getLang(text: String): String = {

new LangouageIdentifier(text).getLanguage

}

import org.apache.spark.sql.functions.\_

val getLangUdf = udf(getLang \_)

val tweets = dt.withColumn("tikaLang", getLangUdf(col("text")))

tweets.createOrReplaceTempView("tweets")

%sql select tikaLang, count(\*) count from tweets group by tikaLang order by count desc

%sql select lang, count(\*) count from tweets group by lang order by count desc

%sql select count(lang), count(tikaLang) from tweets group by lang order by count desc

spark.sql("select count(\*) from tweets where tikaLang <> lang").first.get(0)

for (i <- 1 to 20) {

println("Text length < " + i + ": " + tweets.filter("length(text) < " + i).count + " Tweets.")

}

%sql select text, lang from tweets where length(text) < 3

val clean = tweets.filter("length(text) > 3").filter("lang = 'en'")

clean.count

val cleanFolder = Files.createTempDir().getAbsolutePath + "/tweets"

clean.write.parquet(cleanFolder)

%sh ls /tmp/1476158904873-0/tweets

**[SRC] 04.4. Query and Visualize Tweets with SQL**

val tweets = spark.read.parquet("/tmp/1476160043300-0/tweets")

tweets.printSchema

tweets.createOrReplaceTempView("tweets")

%sql select length(text), count(\*) from tweets group by length(text)

import org.apache.spark.sql.functions.\_

import java.sql.Timestamp

def timeStamp(i: Int): Timestamp = new Timestamp(i)

val toTimeStamp = udf(timeStamp \_)

val d = tweets.groupBy(minute(toTimeStamp(tweets("createdAt")))).agg(minute(toTimeStamp(tweets("createdAt"))), count("\*"))

d.show

d.createOrReplaceTempView("d")

%sql select \* from d

%sql select count(\*) from tweets where geoLocation is not null

%sql select count(\*) from tweets where geoLocation is null and user.location is not null

%sql select user.location from tweets where geoLocation is null and user.location is not null limit 10

%sql select geoLocation.longitude, geoLocation.latitude from tweets where geoLocation is not null

%sql select user.friendsCount, user.followersCount from tweets

%sql select distinct user.name from tweets where user.followersCount > 63000