# [SRC] 07.1. Word Cloudify your Dataset

val tweets = spark.read.parquet("/dataset/twitter/reference")

import org.apache.spark.ml.feature.Tokenizer

val tokenizer = new Tokenizer()

.setInputCol("text")

.setOutputCol("words")

val words = tokenizer.transform(tweets)

import org.apache.spark.ml.feature.StopWordsRemover

val remover = new StopWordsRemover()

.setInputCol("words")

.setOutputCol("noStopWords")

val noStopWords = remover.transform(words)

val counts = noStopWords.rdd

.flatMap(w => w.getAs[Seq[String]]("noStopWords"))

.map(\_ -> 1)

.reduceByKey(\_ + \_)

.sortBy(\_.\_2, ascending = false)

.toDF("word", "count")

counts.persist

val select = counts.rdd.filter(\_.getAs[String]("word").length > 2)

val w = select.take(30).map(r => r.getAs[String]("word"))

z.put("words", w)

val wCount = select.take(30).map(r => r.getAs[Int]("count"))

z.put("wordsCount", wCount)

%r {"imageWidth": "500px"}

library(wordcloud)

library(RColorBrewer)

words <- unlist(z.get("words"))

wordsCount <- unlist(z.get("wordsCount"))

wordcloud(as.factor(words), wordsCount, random.order = FALSE, colors = brewer.pal(8, "Dark2"))

val expand = select.take(30).map( r => {

val sb = new StringBuilder

val w = r.getAs[String]("word")

for (i <- 0 to r.getAs[Int]("count")) {

sb ++= w

sb ++= " "

}

sb.toString()

})

z.put("words", expand.mkString(" "))

%pyspark

import matplotlib

import io

matplotlib.use('Agg')

import matplotlib.pyplot as plt

def show(p):

img = io.StringIO()

p.savefig(img, format='svg')

img.seek(0)

print("%html <div style='width:600px'>" + img.getvalue() + "</div>")

from os import path

from wordcloud import WordCloud

text = z.get("words")

wordcloud = WordCloud().generate(text)

plt.imshow(wordcloud)

plt.axis("off")

# lower max\_font\_size

wordcloud = WordCloud(max\_font\_size=40).generate(text)

plt.figure()

plt.imshow(wordcloud)

plt.axis("off")

show(plt)

import org.apache.commons.codec.binary.Base64

import wordcloud.{CollisionMode, PolarBlendMode, PolarWordCloud}

import wordcloud.bg.CircleBackground

import wordcloud.font.scale.SqrtFontScalar

import wordcloud.nlp.FrequencyAnalizer

import wordcloud.bg.PixelBoundryBackground

import wordcloud.palette.ColorPalette

import java.io.ByteArrayOutputStream

import javax.imageio.ImageIO

import java.net.URL

import java.io.FileInputStream

import java.awt.Color

import scala.io.Source

import collection.JavaConverters.\_

import collection.JavaConversions.\_

val frequencyAnalizer = new FrequencyAnalizer()

frequencyAnalizer.setWordFrequencesToReturn(750)

frequencyAnalizer.setMinWordLength(4)

val stopwords = scala.io.Source.fromURL("file:///dataset/stop-words/stop-words.txt").getLines().toList.asJava

frequencyAnalizer.setStopWords(stopwords)

val wordFrequencies = frequencyAnalizer.load(expand.toList)

val wordCloud = new PolarWordCloud(300, 300, CollisionMode.PIXEL\_PERFECT, PolarBlendMode.BLUR)

wordCloud.setPadding(2)

val backgroundFile = new FileInputStream("/dataset/image/datalayer-logo-02.png")

wordCloud.setBackground(new PixelBoundryBackground(backgroundFile))

wordCloud.setBackgroundColor(Color.WHITE);

wordCloud.setFontScalar(new SqrtFontScalar(10, 40))

wordCloud.build(wordFrequencies)

val baos = new ByteArrayOutputStream()

ImageIO.write(wordCloud.getBufferedImage(), "png", baos)

val image = new String(Base64.encodeBase64(baos.toByteArray()))

println("%img " + image)

z.angularBind("words", expand.mkString(" "))

%angular

<!DOCTYPE html>

<meta charset="utf-8">

<body>

<script src="http://d3js.org/d3.v3.min.js"></script>

<script src="https://rawgit.com/jasondavies/d3-cloud/master/build/d3.layout.cloud.js"></script>

<script>

// Simple animated example of d3-cloud - https://github.com/jasondavies/d3-cloud

function wordCloud(selector) {

var fill = d3.scale.category20();

var svg = d3.select(selector).append("svg")

.attr("width", 500)

.attr("height", 500)

.append("g")

.attr("transform", "translate(250,250)");

function draw(words) {

var cloud = svg.selectAll("g text")

.data(words, function(d) { return d.text; })

cloud.enter()

.append("text")

.style("font-family", "Impact")

.style("fill", function(d, i) { return fill(i); })

.attr("text-anchor", "middle")

.attr('font-size', 1)

.text(function(d) { return d.text; });

cloud

.transition()

.duration(600)

.style("font-size", function(d) { return d.size + "px"; })

.attr("transform", function(d) {

return "translate(" + [d.x, d.y] + ")rotate(" + d.rotate + ")";

})

.style("fill-opacity", 1);

cloud.exit()

.transition()

.duration(200)

.style('fill-opacity', 1e-6)

.attr('font-size', 1)

.remove();

}

return {

update: function(words) {

d3.layout.cloud().size([500, 500])

.words(words)

.padding(5)

.rotate(function() { return ~~(Math.random() \* 2) \* 90; })

.font("Impact")

.fontSize(function(d) { return d.size; })

.on("end", draw)

.start();

}

}

}

function getWords(words) {

return words

.replace(/[!\.,:;\?]/g, '')

.split(' ')

.map(function(d) {

return {text: d, size: 10 + Math.random() \* 60};

})

}

function showNewWords(words, vis) {

vis.update(getWords(words));

// setTimeout(function() { showNewWords(words, vis)}, 2000);

}

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

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

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

showNewWords(newValue, wordCloud('#word-cloud'));

})

});

</script>

<div id="words" ng-model="words"/>

<div id="word-cloud"/>

# [SRC] 07.2. Locate Users and Display Heatmaps with GeoHash

object GeoHash extends Serializable {

val base32 = "0123456789bcdefghjkmnpqrstuvwxyz"

def decodeBounds( geohash:String ):((Double,Double),(Double,Double)) = {

def toBitList( s:String ) = s.flatMap{

c => ("00000" + base32.indexOf(c).toBinaryString ).

reverse.take(5).reverse.map('1' == ) } toList

def split( l:List[Boolean] ):(List[Boolean],List[Boolean]) ={

l match{

case Nil => (Nil,Nil)

case x::Nil => ( x::Nil,Nil)

case x::y::zs => val (xs,ys) =split( zs );( x::xs,y::ys)

}

}

def dehash( xs:List[Boolean] , min:Double,max:Double):(Double,Double) = {

((min,max) /: xs ){

case ((min,max) ,b) =>

if( b )( (min + max )/2 , max )

else ( min,(min + max )/ 2 )

}

}

val ( xs ,ys ) = split( toBitList( geohash ) )

( dehash( ys ,-90,90) , dehash( xs, -180,180 ) )

}

def decode( geohash:String ):(Double,Double) = {

decodeBounds(geohash) match {

case ((minLat,maxLat),(minLng,maxLng)) => ( (maxLat+minLat)/2, (maxLng+minLng)/2 )

}

}

def encode( lat:Double, lng:Double ):String = encode(lat,lng,12)

def encode( lat:Double, lng:Double, precision:Int ):String = {

var (minLat,maxLat) = (-90.0,90.0)

var (minLng,maxLng) = (-180.0,180.0)

val bits = List(16,8,4,2,1)

(0 until precision).map{ p => {

base32 apply (0 until 5).map{ i => {

if (((5 \* p) + i) % 2 == 0) {

val mid = (minLng+maxLng)/2.0

if (lng > mid) {

minLng = mid

bits(i)

} else {

maxLng = mid

0

}

} else {

val mid = (minLat+maxLat)/2.0

if (lat > mid) {

minLat = mid

bits(i)

} else {

maxLat = mid

0

}

}

}}.reduceLeft( (a,b) => a|b )

}}.mkString("")

}

}

def geohash(lat: Double, lng: Double): String = GeoHash.encode(lat, lng)

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

val geohashUdf = udf((lat: Double, lng: Double) => geohash(lat, lng))

import org.apache.spark.sql.Row

import org.apache.spark.sql.functions.explode

val tweets = spark.read.json("/dataset/twitter/london")

val lngLat = tweets.select(explode(tweets("place.boundingBoxCoordinates"))).rdd.map(a => {

val sw = a.getAs[Seq[Row]](0)(0)

val ne = a.getAs[Seq[Row]](0)(2)

val meanLng = sw.get(1).asInstanceOf[Double] + (ne.get(1).asInstanceOf[Double] - sw.get(1).asInstanceOf[Double]) / 2

val meanLat = ne.get(0).asInstanceOf[Double] + (sw.get(0).asInstanceOf[Double] - ne.get(0).asInstanceOf[Double]) / 2

(meanLng, meanLat)

}).toDF("lng", "lat")

lngLat.count

lngLat.show

val geohashed = lngLat.withColumn("geohash", geohashUdf(col("lat"), col("lng")))

geohashed.show

val londons = geohashed.select(col("geohash").substr(0, 5)).rdd.countByValue()

val terms = londons.map(r => s"""{ "term": "${r.\_1(0)}", "count": ${r.\_2} }""").mkString(", ")

val json = s"""

{

"\_type": "terms",

"terms": [

${terms}

]

,

"other": 406994,

"missing": 50

}

"""

z.angularBind("londons", json);

%angular

<style>

.leaflet-tile-pane {

opacity: 0.3;

}

svg {

width: auto !important;

height: auto !important;

}

</style>

<link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.7.5/leaflet.css" type="text/css" media="screen" />

<link rel="stylesheet" href="http://platform.datalayer.io/leaflet-dvf/dist/css/dvf.css" type="text/css" media="screen" />

<link rel="stylesheet" href="http://platform.datalayer.io/leaflet-dvf/examples/css/example.css" type="text/css" media="screen" />

<div class="container-fluid">

<div class="row-fluid">

<div id="map" class="span9"></div>

<div id="legend" class="span3"></div>

</div>

</div>

<script type="text/javascript" src="http://cdn.leafletjs.com/leaflet-0.7.5/leaflet.js"></script>

<script type="text/javascript" src="http://platform.datalayer.io/leaflet-dvf/examples/lib/jsts/javascript.util.js"></script>

<script type="text/javascript" src="http://platform.datalayer.io/leaflet-dvf/examples/lib/jsts/jsts.js"></script>

<script type="text/javascript" src="http://platform.datalayer.io/leaflet-dvf/examples/lib/date.format.js"></script>

<script type="text/javascript" src="http://platform.datalayer.io/leaflet-dvf/examples/lib/geohash.js"></script>

<script type="text/javascript" src="http://maps.stamen.com/js/tile.stamen.js?v1.2.4"></script>

<script type="text/javascript" src="http://platform.datalayer.io/leaflet-dvf/dist/leaflet-dvf.min.js"></script>

<script type="text/javascript">

function render(geohashData) {

geohashData = JSON.parse(geohashData);

var map;

map = L.map('map').setView([0.0, 0.0], 2);

var baseLayer = new L.StamenTileLayer('toner', {

detectRetina: true

});

baseLayer.addTo(map);

var colorFunction = new L.HSLHueFunction(new L.Point(357,200), new L.Point(34677,0), {outputSaturation: '100%', outputLuminosity: '25%'});

var fillColorFunction = new L.HSLHueFunction(new L.Point(357,200), new L.Point(34677,0), {outputSaturation: '100%', outputLuminosity: '50%'});

var options = {

recordsField: 'terms',

geohashField: 'term',

displayOptions: {

count: {

color: colorFunction,

fillColor: fillColorFunction,

gradient: true

}

},

layerOptions: {

fillOpacity: 0.7,

opacity: 1,

weight: 1,

gradient: true

}

};

var layer = new L.GeohashDataLayer(geohashData, options);

map.addLayer(layer);

$('#legend').append(layer.getLegend({

numSegments: 20,

width: 80,

className: 'well'

}));

}

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

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

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

render(newValue);

})

});

</script>

<div id="londons" ng-model="londons"/>