Universidad de Sonora

División de Ciencias Exactas y Naturales

"Análisis de Global Terrorism Database con PySpark"

Licenciatura en Ciencias de la Computación

Autor: Cesar Ernesto Salazar Buelna

Hermosillo Sonora, Diciembre del 2018

Introducción

En el mundo se acumulan cada vez más datos en formato digital, el problema es que se encuentran poco estructurados y en cantidades ingentes. Uno de los objetivos del uso de las tecnologias Big Data es el transformar los datos en conocimiento útil. Para ello se necesitan herramientas que nos ayudan a analizar, procesar y almacenar todos los datos recogidos. Actualmente existe un gran número de estas. Muchas de las mejores herramientas son **Open Source**, pero tambien existen buenas alternativas de pago.

En este documento se muestra un análisis de la **Global Terrorism Database** con la herramienta **Apache Spark**.

Global Terrorism Database

Los datos fueron tomados de https://www.kaggle.com/START-UMD/gtd

La Global Terrorism Database es una base de datos de codigo abierto que incluye información sobre ataques terroristas en todo el mundo desde 1970 hasta el 2017. Hasta ahora incluye más de 180,000 ataques. La base de datos es sostenida por investigadores del National Consortium for the Study of Terrorism and Responses to Terrorism (START) con sede en la Universidad de Maryland.

Contiene más de 100 variables, entre estas, el año, mes, dia, país, region, las coordenadas, los tipos de ataques, el número de muertos, etc. La fuente son articulos de medios no clasificados.

PySpark

Apache Spark es un motor de procesamiento de datos de código abierto realmente rápido.

Creado por Matei Zaharia en la Universidad de Berkley, se considera el primer software de open source que hace programación distribuida (muy en esncia, consiste en distribuir el trabajo entre un grupo de ordenadores, “cluster”, que trabajan como uno) realmente accesible a los científico de datos.

Se pueden programar aplicaciones usando diferentes lenguajes como java, Scala, Python o R, pudiendo ser, según el programa, hasta 100 veces más rápido en memoria o 10 veces más en disco que Hadoop MapReduce.

Para este análisis se utilizara Apache Spark. Esta escrito en Scala. Para adminir Python con Spark, la comunidad Apache Spark lanzó una herramienta, PySpark. Se hara un sencillo análisis de las columnas más importantes y algunas visualizaciones para intentar tener una idea de lo que contiene el dataset.

Análisis

Primero se tiene que importar pyspark, e inicializar el contexto para poder usar sus herramientas.

```
import pyspark as spark
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("Python Spark SQL basic example") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()
```

Tambien se tiene que leer el archivo de los ataques terroristas

```
df = spark.read.format("csv").option("header", "true").load("terrorismo.csv")
```

Para mostrar la primer linea de los datos leidos usamos `df.head()`

```
Row(eventid='197000000001', iyear='1970', imonth='7', iday='2',
approxdate=None, extended='0', resolution=None, country='58',
country_txt='Dominican Republic', region='2', region_txt='Central America &
Caribbean', provstate=None, city='Santo Domingo', latitude='18.456792',
longitude='-69.951164', specificity='1', vicinity='0', location=None,
summary=None, crit1='1', crit2='1', crit3='1', doubtterr='0',
alternative=None, alternative_txt=None, multiple='0', success='1',
suicide='0', attacktype1='1', attacktype1_txt='Assassination',
attacktype2=None, attacktype2_txt=None, attacktype3=None,
attacktype3_txt=None, targtype1='14', targtype1_txt='Private Citizens &
Property', targsubtype1='68', targsubtype1_txt='Named Civilian', corp1=None,
target1='Julio Guzman', natlty1='58', natlty1_txt='Dominican Republic',
targtype2=None, targtype2_txt=None, targsubtype2=None, targsubtype2_txt=None,
corp2=None, target2=None, natlty2=None, natlty2_txt=None, targtype3=None,
targtype3_txt=None, targsubtype3=None, targsubtype3_txt=None, corp3=None,
target3=None, natlty3=None, natlty3_txt=None, gname='MANO-D', gsubname=None,
gname2=None, gsubname2=None, gname3=None, gsubname3=None, motive=None,
guncertain1='0', guncertain2=None, guncertain3=None, individual='0',
nperps=None, nperpcap=None, claimed=None, claimmode=None, claimmode_txt=None,
claim2=None, claimmode2=None, claimmode2_txt=None, claim3=None,
claimmode3=None, claimmode3_txt=None, compclaim=None, weaptype1='13',
weaptype1_txt='Unknown', weapsubtype1=None, weapsubtype1_txt=None,
weaptype2=None, weaptype2_txt=None, weapsubtype2=None, weapsubtype2_txt=None,
weaptype3=None, weaptype3_txt=None, weapsubtype3=None, weapsubtype3_txt=None,
weaptype4=None, weaptype4_txt=None, weapsubtype4=None, weapsubtype4_txt=None,
weapdetail=None, nkill='1', nkillus=None, nkillter=None, nwound='0',
nwoundus=None, nwoundte=None, property='0', propextent=None,
propextent_txt=None, propvalue=None, propcomment=None, ishostkid='0',
nhostkid=None, nhostkidus=None, nhours=None, ndays=None, divert=None,
kidhijcountry=None, ransom='0', ransomamt=None, ransomamtus=None,
ransompaid=None, ransompaidus=None, ransomnote=None, hostkidoutcome=None,
hostkidoutcome_txt=None, nreleased=None, addnotes=None, scite1=None,
scite2=None, scite3=None, dbsource='PGIS', INT_LOG='0', INT_IDEO='0',
INT_MISC='0', INT_ANY='0', related=None)
```

Como se ve, son muchas columnas, por esa razon nos quedaremos con algunas columnas mas importantes.

Para tomar las columas mas interesantes utilizamos el siguiente código (tambien se eliminaron los renglones que tienen valores nulos) :

```
guardar solo las columas año, mes, pais, region, y el tipo d ataque
datos_seleccionados =
df.select("iyear","imonth","country_txt","region_txt","attacktype1_txt","nkill
")
#eliminar nulos
datos_seleccionados = datos_seleccionados.dropna(how='any')
#mostrar los primeros 10
datos_seleccionados.show(10)
```

Este código tambien muestra los primeros 10 renglones del dataset leido.

```
+-----+------+-----------------+--------------------+--------------------+-----+
|iyear|imonth|      country_txt|          region_txt|    attacktype1_txt|nkill|
+-----+------+-----------------+--------------------+--------------------+-----+
| 1970|     7|Dominican Republic|Central America &...|       Assassination|    1|
| 1970|     0|           Mexico|       North America|Hostage Taking (K...|    0|
| 1970|     1|      Philippines|      Southeast Asia|       Assassination|    1|
| 1970|     1|    United States|       North America|       Armed Assault|    0|
| 1970|     1|          Uruguay|       South America|       Assassination|    0|
| 1970|     1|    United States|       North America|   Bombing/Explosion|    0|
| 1970|     1|    United States|       North America|Facility/Infrastr...|    0|
| 1970|     1|    United States|       North America|Facility/Infrastr...|    0|
| 1970|     1|    United States|       North America|   Bombing/Explosion|    0|
| 1970|     1|    United States|       North America|Facility/Infrastr...|    0|
+-----+------+-----------------+--------------------+--------------------+-----+
only showing top 10 rows
```

Un analísis de cada columna se muestra a continuación:

```
print("Paises distintos: ",datos_seleccionados.select("country_txt").distinct().count())
Paises distintos:  202
print("Regiones distintas: ",datos_seleccionados.select("region_txt").distinct().count())
Regiones distintas:  12
print("Ataques distintos:",datos_seleccionados.select("attacktype1_txt").distinct().count())
Tipos de ataques distintos:  20
datos_seleccionados.describe("iyear","imonth","nkill").show()
+-------+------------------+------------------+------------------+
|summary|             iyear|            imonth|             nkill|
+-------+------------------+------------------+------------------+
|  count|            170793|            170793|            170793|
|   mean|2003.070518112569|6.479000895821257|  2.4038225968104|
| stddev|12.951105702923726|3.3877264610887305|11.554775970212424|
|    min|              1970|                 0|              RPGs|
|    max|              2017|                 9|          wrenches|
+-------+------------------+------------------+------------------+
```

Ahora se realizó un conteo por columna para ver cuales eran el pais, año, región y tipo de ataque mas comunes entre todos los renglones.

Años

```
years=datos_seleccionados.groupBy('iyear').count()
years.orderBy(years['count'].desc()).show(10)
```

```
+-----+-----+
|iyear|count|
+-----+-----+
| 2014|15855|
| 2015|14147|
| 2016|12764|
| 2013|11614|
| 2017|10237|
| 2012| 8208|
| 1992| 5046|
| 2011| 5025|
| 2010| 4790|
| 2008| 4698|
+-----+-----+
```

Contando el número de ataques por año, se observa que los últimos 5 años son los que tienen mas ataques terroristas registrados, siendo el 2014 el que tiene más con 15,855.

Paises

```
country=datos_seleccionados.groupBy('country_txt').count()
country.orderBy(country['count'].desc()).show(10)
```

```
+--------------+-----+
|   country_txt|count|
+--------------+-----+
|          Iraq|23781|
|      Pakistan|14086|
|   Afghanistan|12351|
|         India|11694|
|      Colombia| 7826|
|   Philippines| 6649|
|          Peru| 5455|
|United Kingdom| 5050|
|        Turkey| 4136|
|   El Salvador| 3939|
+--------------+-----+
```

Contando los ataques por paises, se ve que los tres primeros lugares con mas ataques son Iraq, Pakistan y Afhganistan. Este resultado era quizas esperado por los conflictos en los que han estado estos paises a lo largo de los años.

Regiones

```
region=datos_seleccionados.groupBy('region_txt').count()
region.orderBy(region['count'].desc()).show(10)
```

```
+--------------------+-----+
|          region_txt|count|
+--------------------+-----+
|Middle East & Nor...|47981|
|          South Asia|43913|
|       South America|17267|
|  Sub-Saharan Africa|15901|
|      Western Europe|15666|
|      Southeast Asia|12135|
|Central America &...| 8021|
|      Eastern Europe| 4951|
|       North America| 3192|
|           East Asia|  757|
+--------------------+-----+
```

La región con mas ataques es la del Medio Este y Norte de África siguiendolo de cerca el Sur de Asia

Tipo de ataque

```
attacktype=datos_seleccionados.groupBy('attacktype1_txt').count()
attacktype.orderBy(attacktype['count'].desc()).show(10)
```

```
+--------------------+-----+
|     attacktype1_txt|count|
+--------------------+-----+
|   Bombing/Explosion|83975|
|       Armed Assault|40209|
|       Assassination|19202|
|Facility/Infrastr...| 9634|
|Hostage Taking (K...| 8554|
|             Unknown| 6556|
|     Unarmed Assault|  987|
|Hostage Taking (B...|  894|
|           Hijacking|  603|
| the Red Hand Def...|    1|
+--------------------+-----+
```

Por ultimo de todos los ataques cerca del 50% de todos son ataques con bombas y explosiones.

Análisis de Coordenadas

Para visualizar este conjunto de datos, se implementará k-means en PySpark para agrupar las diferentes ubicaciones geográficas de los ataques terroristas.

Para utilizar solo los campos de latitud y longitud del conjunto de datos, se utilizó tambien la librería pandas.

Primero se importan la librerias necesarias para graficar, y realizar varias operaciones.

```
.import pandas as pd
```

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
```

Despues usando pandas obtenemos las columnas “latitud” y “longitud” de entre las más de 100 y las guardamos en un archivo csv. Tambien se quitaron los renglones donde se encontraron valores faltantes.

```
#se lee el archivo con los datos
datos = pd.read_csv('terrorismo.csv', encoding='latin-1')
#se filtra para soo obtener las coordenadas
lat_lng_datos = pd.DataFrame(datos, columns=['latitude', 'longitude'])
datos_limpios = lat_lng_datos.dropna(axis=0, how='any')

# guardar los datos en CSV
has_header = True # False when uploading to cluster
datos_limpios.to_csv('datos_limpios_header.csv', columns = ["latitude",
"longitude"], index=False, header=has_header)
```

Utilizando pyspark podemos visualizar los datos que se guardaron anteriormente.

```
coordenadas = sc.textFile("datos_limpios.csv")
#primero se muestran los primeros 5 renglones
print("Primeros 5 renglones:\n",coordenadas.take(5))
#despues se cuenta los renglones que se tienen
print("\nHay ", coordenadas.count(), " pares de coordendas.")
```

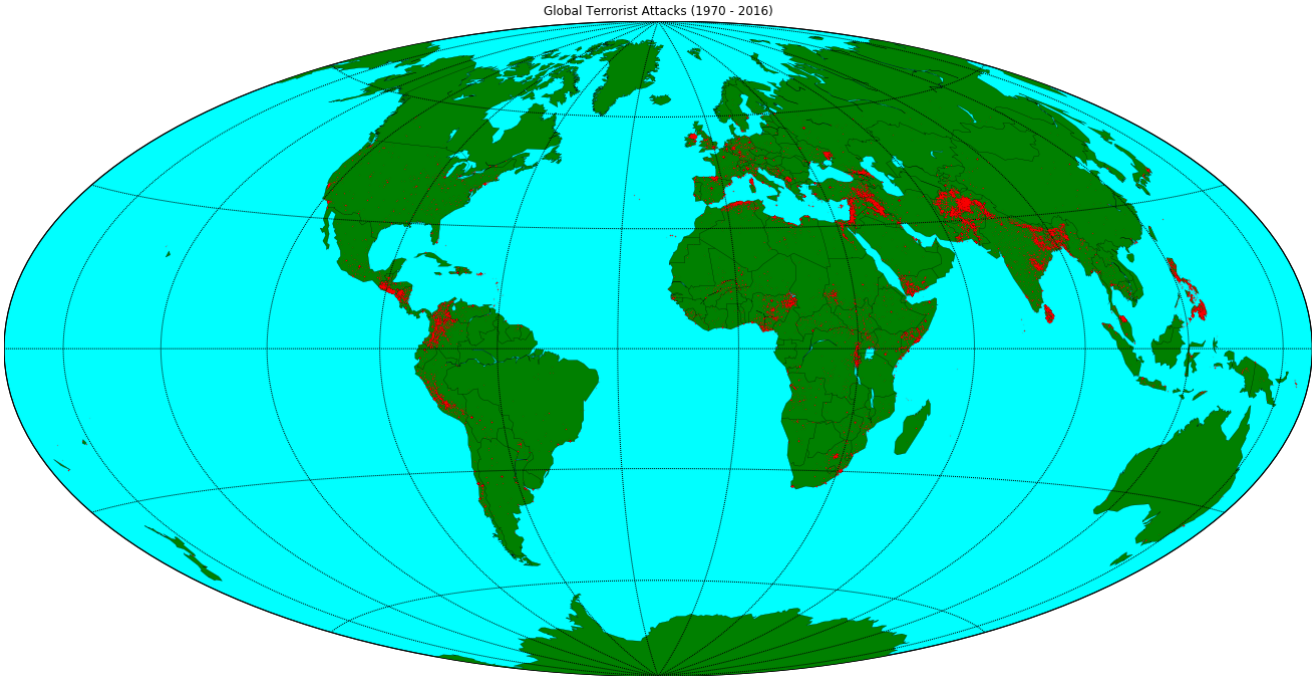
```
Primeros 5 renglones:
 ['18.456792,-69.951164', '19.371887,-99.086624',
'15.478598000000002,120.599741', '37.99749,23.762728', '33.580412,130.396361']

Hay  177134  pares de coordendas.
```

Creando el mapa con todos los ataques terroristas

Para ver la distribuicion de los ataques terroristas en un mapa mundial utilizamos los dataframes de pandas y la librería matplotlib .

```
clean_data_df = pd.read_csv('datos_limpios_header.csv')
clean_lat = clean_data_df['latitude']
clean_lon = clean_data_df['longitude']
latG = clean_lat.tolist()
lonG = clean_lon.tolist()

plt.figure(figsize=(24,12))

map = Basemap(projection='hammer',lon_0=-20)

map.drawcoastlines(linewidth=0.25)
map.drawcountries(linewidth=0.25)
map.fillcontinents(color='green',lake_color='aqua')

map.drawmapboundary(fill_color='aqua')
```

```
# draw lat/lon grid lines every 30 degrees.
map.drawmeridians(np.arange(0,360,30))
map.drawparallels(np.arange(-90,90,30))

x,y = map(lonG, latG)
map.plot(x, y, 'ro', markersize=.05)

title_string = "Global Terrorist Attacks (1970 - 2016)"
plt.title(title_string)

plt.show()
```



K-Means

Como ejemplo ilustrativo de agrupamiento en este conjunto de datos, se establece k = 5 porque se correspondería aproximadamente con un agrupamiento para cada continente.

Primero se importa de la librería mllib.

```
from pyspark.mllib.clustering import Kmeans
```

Esta librería tiene implementado el algoritmo KMeans. A continuación se leen los datos limpios.

```
text = sc.textFile("datos_limpios.csv")
# Use the below commands for files where each line is a lat, long pair
separated by commas.
split = coordenadas.map(lambda line: line.split(','))
data = split.map(lambda line: (float(line[0]), float(line[1])))
data.persist()
```

Se inicia el algoritmo K-Means.

```
clusters = KMeans.train(data, 5, maxIterations=100000, initializationMode="random")
```

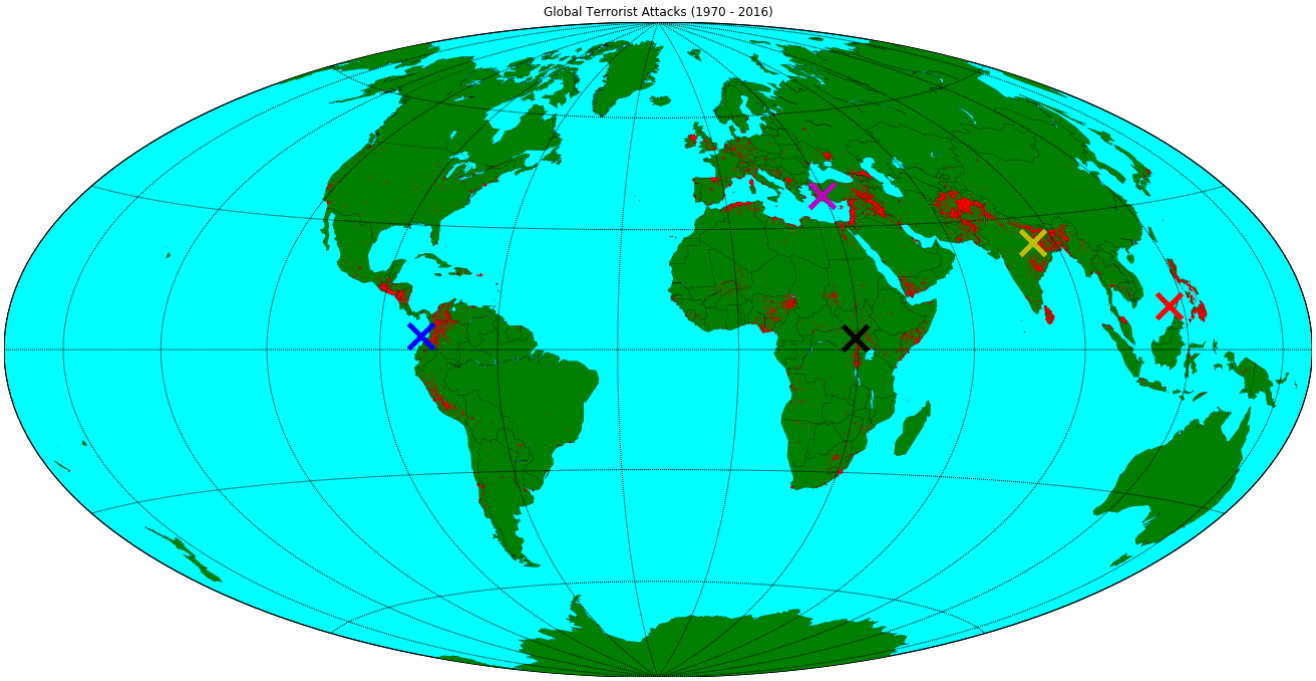
A continuación se muestran los 5 centroides resultantes:

```
[array([  3.41065696, -79.7535064 ]),
 array([  2.86456376,  29.20992461]),
 array([  9.20774026, 115.80481474]),
 array([ 38.20999153,  29.06040902]),
 array([ 24.28047414,  83.44332289])]
```

Creando el mapa con todos los ataques terroristas con los centroides resultantes

Ahora se mostrará en el mapa todos los ataques terroristas ademas marcando los 5 centroides resultantes del algoritmo Kmeans.

```
colors = ['b', 'k', 'r', 'm', 'y', 'g', 'c']
centroid_colors = zip(centroids, colors)

plt.figure(figsize=(24,12))
map = Basemap(projection='hammer',lon_0=-20)

map.drawcoastlines(linewidth=0.25)
map.drawcountries(linewidth=0.25)
map.fillcontinents(color='green',lake_color='aqua')

map.drawmapboundary(fill_color='aqua')

# draw lat/lon grid lines every 30 degrees.
map.drawmeridians(np.arange(0,360,30))
map.drawparallels(np.arange(-90,90,30))

x,y = map(lonG, latG)
map.plot(x, y, 'ro', markersize=.05)

for (lat, lon), color in centroid_colors:
    x,y = map(lon, lat)
    marker_style = color + 'x'
    map.plot(x, y, marker_style, markersize=25, mew=5)

title_string = "Global Terrorist Attacks (1970 - 2016)"
plt.title(title_string)

plt.show()
```



La visualización de los centroides en el mapa muestra que los lugares donde se centran los ataques terroristas son: el centro de África, Asia del sur, Medio oriente, el norte de America del sur y la parte norte de Oceanía.

Aquí se encuantra todo el código y los resultados del análisis.

Conclusión

Cuando se trabaja con una gran cantidad de datos, se necesita mucho poder de computo, para ello, Apache Spark es una buena alternativa.

Cuenta con herramientas que permiten visualizar, realizar calculos estadisticos, y tambien tiene la librería MlLib para poder utilizar *machine learning* con *big data*, lo que es muy útil hoy en dia. Además, hace todo esto en paralelo mediante el uso de diferentes clusters lo que lo hace más eficiente.