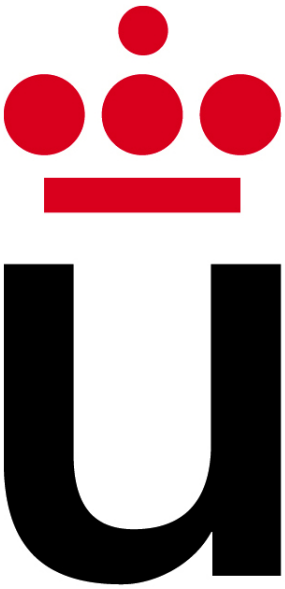
Práctica final

Análisis de Big Data

Análisis de una comunidad virtual de Stack Exchange

José Ignacio Escribano



Móstoles, 10 de abril de 2016

Índice de tablas

Índice

1. Introducción

En esta práctica final resolveremos una serie de cuestiones relativas a los usuarios que han participado en la comunidad `http://scifi.stackexchange.com`, dedicada a resolver preguntas sobre temas de ciencia ficción y fantasía.

El archivo `users.xml` contiene información sobre el perfil de un grupo del total de usuarios de la comunidad que han participado en ella, hasta marzo de 2015.

Utilizaremos Apache Spark para realizar el análisis de este archivo, utilizando un DataFrames.

2. Carga de los datos

Para cargar los datos, usamos el siguiente comando:

```
df = sqlContext.read.format('com.databricks.spark.xml').options(rowTag='row').load(
```

Este creará un DataFrame a partir del fichero `users.xml`, que se leerá como archivo `xml`, y se guardará los hijos de un elemento `row` como una columna, quedando como una tabla.

3. Ejercicio 1

Inspeccionamos las primeras filas del archivo para ver cuáles son los elementos de los que consta cada uno de ellos.

```
<users>
  <row>
    <Id>-1</Id>
    <Reputation>1</Reputation>
    <CreationDate>2011-01-11T19:19:36.483</CreationDate>
    <DisplayName>Community</DisplayName>
    <LastAccessDate>2011-01-11T19:19:36.483</LastAccessDate>
    <Location>on the server farm</Location>
    <Views>0</Views>
    <UpVotes>2587</UpVotes>
    <DownVotes>3953</DownVotes>
    <AccountId>-1</AccountId>
  </row>
  <row>
    <Id>2</Id>
    <Reputation>101</Reputation>
    <CreationDate>2011-01-11T19:50:40.620</CreationDate>
```

```
    <DisplayName>Geoff Dalgas</DisplayName>
    <LastAccessDate>2015-02-05T00:03:28.030</LastAccessDate>
    <Location>Corvallis, OR</Location>
    <Views>21</Views>
    <UpVotes>1</UpVotes>
    <DownVotes>0</DownVotes>
    <Age>38</Age>
    <AccountId>2</AccountId>
  </row>
  ...
</users>
```

Observamos que tenemos 11 elementos en cada fila:

- AccountId: Id de la cuenta
- Age: edad del usuario
- CreationDate: fecha de creación de la cuenta
- DisplayName: nombre de usuario
- DownVotes: votos a favor
- Id: Id de usuario
- LastAccessDate: fecha del último acceso
- Location: localización
- Reputation: puntos de reputación
- UpVotes: votos a favor
- Views: visitas a su página de perfil

Otra forma de comprobar el número de columnas de nuestro fichero es imprimir el esquema usando el comando

```
df.printSchema()
```

La salida de este comando es

```
root
  |-- AccountId: long (nullable = true)
  |-- Age: long (nullable = true)
  |-- CreationDate: string (nullable = true)
  |-- DisplayName: string (nullable = true)
```

```
|-- DownVotes: long (nullable = true)
|-- Id: long (nullable = true)
|-- LastAccessDate: string (nullable = true)
|-- Location: string (nullable = true)
|-- Reputation: long (nullable = true)
|-- UpVotes: long (nullable = true)
|-- Views: long (nullable = true)
```

Además de las columnas, este comando nos devuelve el tipo de cada variable.

Si queremos ver el contenido del fichero, usamos el comando, que muestra las 10 primeras filas del fichero.

```
df.show(10)
```

El comando devuelve la Tabla 1.

Tabla 1: Primeras líneas del fichero `users.xml`

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views
-1	null	2011-01-11T19:19:...	Community	3953	-1	2011-01-11T19:19:...	on the server farm	1	2587	0
2	38	2011-01-11T19:50:...	Geoff Dalgas	0	2	2015-02-05T00:03:...	Corvallis, OR	101	1	21
7598	30	2011-01-11T19:55:..	Nick Craver	0	3	2015-03-01T13:49:...	Winston-Salem, NC	101	3	10
1998	29	2011-01-11T20:17:...	Emmett	0	4	2013-05-06T20:52:...	San Francisco, CA	101	0	7
29738	null	2011-01-11T20:18:...	Kevin Montrose	0	5	2015-02-15T05:27:...	New York City, Ne...	101	32	11
32917	30	2011-01-11T20:29:...	David Fullerton	4	6	2015-03-06T22:57:...	New York, NY	99	63	13
33603	30	2011-01-11T20:37:...	Sorantis	0	7	2014-07-23T12:00:...	Sweden	146	5	15
51549	38	2011-01-11T20:37:...	GAThrawn	0	8	2015-02-18T23:17:...	United Kingdom	103	19	98
3874	48	2011-01-11T20:38:...	Rodger Cooley	3	9	2015-03-04T19:47:...	Houston, TX	1816	211	85
15651	42	2011-01-11T20:38:...	MatthewMartin	2	10	2015-03-06T01:30:...	United States	1787	78	48

4. Ejercicio 2

A continuación, resolveremos una serie de consultas sobre el fichero XML.

1. Número total de registros de usuarios en el fichero.

   La consulta que realizamos es la siguiente:

   ```
   totalUsers = df.count()
   print(totalUsers)
   ```

   La consulta devuelve 20333. Es decir, tenemos que el fichero tiene 20333 usuarios.

2. Todos los datos disponibles sobre el usuario con `DisplayName` igual a `ambient_memory`.

   La consulta que realizamos es la siguiente:

Tabla 2: Información del usuario `ambient_memory`

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views
4529331	null	2014-06-20T12:01:...	ambient_memory	0	28494	2014-06-20T12:01:...	New York, United ...	1	0	0

```
df.filter(df["DisplayName"] == "ambient_memory").show()
```

La consulta busca las filas que tienen `ambient_memory` como `DisplayName` y mostramos el resultado en forma tabular. La Tabla 2 muestra los resultados.

3. Los 10 usuarios con mayor reputación.

   La consulta es la siguiente:

```
df.sort(df.Reputation.desc()).limit(10).show()

''' O de forma equivalente '''

df.sort("Reputation", ascending=False).limit(10).show()
```

   La consulta ordena la columna `Reputation` de forma descendente y selecciona las 10 primeras filas y el resultado se muestra en forma tabular. La Tabla 3 muestra los resultados de esta consulta.

Tabla 3: Los 10 usuarios con más reputación

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views
41067	43	2011-02-28T04:00:...	DVK	1806	1806	2015-03-08T02:21:...	New York, NY	148259	5503	6224
893673	null	2011-09-06T19:47:...	Thaddeus	41	2765	2015-03-07T21:21:...	Hayward, CA	118763	1934	3101
3776439	null	2013-12-26T15:54:...	Richard	2288	20774	2015-03-08T02:36:...	UK	92783	3878	7577
1057622	null	2011-11-22T14:47:...	Slytherincess	179	3500	2015-03-08T00:08:...	Azkaban	67739	841	3562
12203	32	2011-02-01T22:00:...	Jeff	44	656	2015-03-08T02:28:...	Cincinnati, OH	62833	2230	1362
486482	44	2012-09-10T13:52:...	Darth Satan	192	8719	2015-03-07T21:49:...	null	57681	910	1515
375283	null	2011-03-07T02:09:...	Keen	1618	1027	2015-03-08T01:44:...	null	49731	4875	2560
359788	53	2011-04-28T17:25:...	Tango	7	1693	2015-03-06T02:46:...	Richmond, VA, USA	48075	1919	1517
102643	36	2011-01-11T20:56:...	DavRob60	242	45	2015-03-06T11:41:...	Salaberry-de-Vall...	47505	3799	1671
1170648	null	2012-03-06T21:45:...	phantom42	406	5184	2015-03-08T03:15:...	Orlando, FL	47257	2003	1771

4. Los usuarios con la fecha de creación más antigua y la más reciente, respectivamente.

   La consulta es la siguiente:

```
''' Usuario más antiguo '''
df.sort("CreationDate", ascending=False).limit(1).show()

''' Usuario más reciente '''
df.sort("CreationDate", ascending=True).limit(1).show()
```

La consulta ordena descendentemente (ascentemente) por la columna `CreationDate`, se selecciona la primera fila y se muestra de forma tabular. La primera consulta devuelve el usuario más antiguo y la segunda, el usuario más reciente.

Las Tablas 4 y 5 muestra los usuarios con fecha de creación de su cuenta más antigua y más reciente, respectivamente.

Tabla 4: Usuario más antiguo

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views
4529331	null	2014-06-20T12:01:...	ambient_memory	0	28494	2014-06-20T12:01:...	New York, United ...	1	0	0

Tabla 5: Usuario más reciente

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views
-1	null	2011-01-11T19:19:...	Community	3953	-1	2011-01-11T19:19:...	on the server farm	1	2587	0

5. El usuario más joven y el más viejo (de los que han indicado una edad válida en el campo `Age`).

   La consulta es la siguiente:

```
''' Importamos las funciones `max` y `min` '''
from pyspark.sql.functions import min, max

''' Calculamos la edad máxima y mínima de la columna `Age` '''
ages = df.select(min("Age").alias("min"), max("Age").alias("max")).collect()
```

   La consulta devuelve

```
[Row(min=14, max=95)]
```

   Es decir, la edad mínima es 14 y la máxima es 95.

   Para conocer la información de los usuarios que tienen estas edades, hacemos la siguiente consulta:

```
''' Usuarios con la edad mínima guardada en la variable `ages` '''
df.filter(df.Age == ages[0].min).show()

''' Usuarios con la edad máxima guardada en la variable `ages` '''
df.filter(df.Age == ages[0].max).show()
```

La consulta muestra los usuarios que tienen la edad mínima o máxima y se muestra de forma tabular. Las Tablas 6 y **??** muestran a los usuarios más jóvenes y viejos, respectivamente.

Tabla 6: Usuarios más jovenes

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views
2178726	14	2013-05-01T20:06:...	danilka1	0	14208	2013-05-01T20:06:...	planet earth	101	0	0
198592	14	2013-08-29T15:11:...	Ramchandra Apte	0	16999	2014-12-24T13:08:...	Bangalore, India	103	9	3

5. Conclusiones

6. Código Python