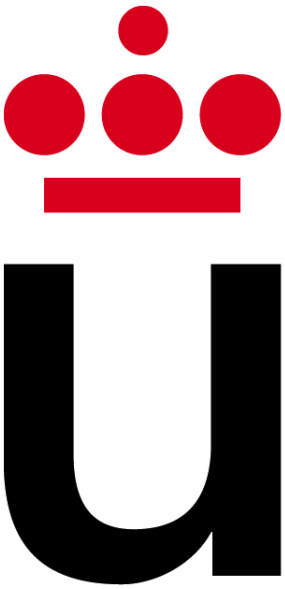
Práctica final

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 **??**.

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views

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

---

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:

   ```
   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 **??** muestra los resultados.

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

3. Los 10 usuarios con mayor reputación (`Reputation`).

   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, selecciona las 10 primeras filas y muestra el resultado en forma tabular. El resultado de la consulta se puede ver en la Figura 3.

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views

41067	43	2011-02-28T04:00:...	DVK	1806	976	2015-03-08T02:21:...			148259	5503	6224
893673	null				2765				118763	1934	3101
3776439	null	2011-09-06T19:47:...	Thaddeus	41	20774	2015-03-07T21:21:...	New York, NY		92783	3878	7577
1057622	null				3500		Hayward, CA		67739	841	3562
12203	32	2013-12-26T15:54:...	Richard	2288	656	2015-03-08T02:36:...	UK		62833	2230	1362
48648244				179	8719		Azkaban		57681	910	1515
375283	null	2011-11-22T14:47:...	Slytherincëss	44	1027	2015-03-08T00:08:...	Cincinnati, OH		49731	4875	2560
35978853				192	1693		null		48075	1919	1517
10264336		2011-02-01T22:00:...	Jeff D	1618	45	2015-03-08T02:28:...	null		47505	3799	1671
1170648	null			7	5184				47257	2003	1771
		2012-09-10T13:52:...	Darth Satan	242		2015-03-07T21:49:...	Richmond, VA, USA				
		2011-03-07T02:09:...	Keen	406		2015-03-08T01:44:...					
		2011-04-28T17:25:...	Tango Da-vRob60			2015-03-06T02:46:...	Salaberry-de-Vall...				
		2011-01-11T20:56:...	phantom42			2015-03-06T11:41:...	Orlando, FL				
		2012-03-06T21:45:...				2015-03-08T03:15:...					

---

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.

   Las Tablas 6 y 7 muestran los usuarios más jóvenes y viejos, respectivamente.

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views
452933	null	2014-06-20T12:01:...	ambient_memory	0	28494	2014-06-20T12:01:...	New York, United ...	1	0	0

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()
```

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

AccountId	Age	CreationDate	DisplayName	DownVotes	Id	LastAccessDate	Location	Reputation	UpVotes	Views

	95	2011-01-19T21:58:...		367	2011-02-13T13:03:...					0	
	95	2011-03-20T00:47:...		1266	2015-03-04T22:20:...					31	
	95	2011-05-03T23:27:...		1761	2015-02-20T01:48:...					1	0
	95	2011-05-20T08:27:...		1925	2011-05-20T17:58:...					1	0
	95	2011-06-04T09:59:...		2115	2015-02-22T17:09:...					3	
	95	2011-06-10T12:49:...		2181	2012-09-12T09:15:...					22	
	95	2011-06-13T08:12:...		2205	2011-06-10T12:49:...					1	
	95	2011-07-02T23:26:...		2341	2015-02-12T14:56:...					10	
	95	2011-07-27T03:05:...		2472	2012-08-29T05:03:...					0	
	95	2011-10-21T03:45:...		3120	2012-08-30T10:02:...					29	
	95	2011-10-29T00:25:...		3192	2011-11-02T00:13:...					2	
	95	2011-11-02T00:13:...		3213	2012-03-14T16:07:...					38	
	95	2012-03-14T16:07:...		5323	2015-03-05T16:45:...					0	8
	95	2012-05-16T19:20:...		6365	2015-03-02T16:56:...					0	0
	95	2012-05-17T23:24:...		6392	2014-01-16T23:46:...						
	95	2012-08-02T19:32:...		8091	2013-07-10T04:29:...						
	95	2012-08-29T05:03:...		8531	2015-03-04T22:47:...						
	95	2012-08-30T10:02:...		8554	2015-03-05T16:45:...						
	95	2012-12-29T01:52:...		11616	2015-03-02T16:56:...						
2913	95	2011-03-20T00:47:...	bill weaver	0							
9606	95	2011-05-03T23:27:...	Mark Bessey	1							
237100	95	2011-05-20T08:27:...	Kevin Sam Meldrum	0							
8802	95	2011-06-04T09:59:...	jufrpeji	0							
299848	95	2011-06-10T12:49:...	Keng	0							
564	95	2011-06-13T08:12:...	HackToHell	1							
413744	95	2011-07-02T23:26:...	Crazy Eddie	0							
114869	95	2011-07-27T03:05:...	Quinn C.	0							
388430	95	2011-10-21T03:45:...	Culver	0							
47040	95	2011-10-29T00:25:...	xiaohouzi79	18							
21774	95	2011-11-02T00:13:...	1.01pm	0							
128120	95	2012-03-14T16:07:...	JK.	11							
150461	95	2012-05-16T19:20:...	iamserious	0							
1399708	95	2012-05-17T23:24:...	JNat	0							
1196073	95	2012-08-02T19:32:...	MrEngineer13	0							
427266	95	2012-08-29T05:03:...	Merlin Pubby	0							
286976	95	2012-08-30T10:02:...	JanOlMajti	0							
989850		2012-12-29T01:52:...	Watch	0							
2187757											

										0	
										31	
										1	0
										1	0
										3	
										22	
										1	
										10	
										0	
										29	
										2	
										38	
										0	8
										0	0
367	2011-02-13T13:03:...										
1266	2015-02-20T01:48:...	United States	101	2							
1761	2011-05-20T17:58:...	California	1723	78							
1925	2015-03-04T22:20:...	Cincinnati, OH	0	3							
2115	2015-02-22T17:09:...	Chesham, United K...	101	0	22						
2181	2012-09-12T09:15:...	Spain	101	1	1						
2205	2011-06-10T12:49:...	Parts Unknown	101	0	10						
2341	2015-02-12T14:56:...	Mordor	101	9	0						
2472	2012-08-29T05:03:...	Gliese 581G	161	0	29						
3120	09-12T09:15:...	South Bend, IN	103	0	2						
3192	2011-11-02T00:13:...	Australia	431	8	38						
3213	06-10T12:49:...	Atlantis, FL	101	3	0	8					
5323	2015-03-05T16:45:...	Teresina, Brazil	944	22	0	0					
6365	02-22T17:09:...	London Europe	101	2							
6392	2014-01-16T23:46:...	United States	521								
8091	08-03T18:08:...	United Kingdom	101	352							
8531	2015-03-04T22:47:...	United States	188	9	4						
8554	01-20T16:08:...	Celje,Slovënia	101	9	0						
11616	2014-02-12T14:56:...	On someones wrist	101								
	2013-11-06T02:00:...		101								
	2014-01-12T13:06:...		89								
	2013-02-04T12:19:...										
	2013-02-13T08:37:...										

---

5. Conclusiones

6. Código Python