Universitat Politècnica de Catalunya

Cerca i Anàlisis d'Informació Massiva

Bachelor Degree in Informatics Engineering

Session 2

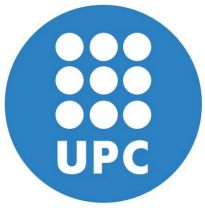
Programming with ElasticSearch

Autores:

Ruben Bagan
Ana Mestre

Profesor:
Javier Béjar Alonso

Q1 Curs 2017-2018



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Índice

1. Modifying ElasticSearch index behavior

En esta primera parte de la práctica la intención es averiguar cómo funcionan los diferentes flags que se pueden cambiar, es decir ***–token*** y ***–filter***. Con solo ejecutar diferentes pruebas queda ejemplificado que los resultados obtenidos son diferentes, pues se basan y aplican diversas metodologías.

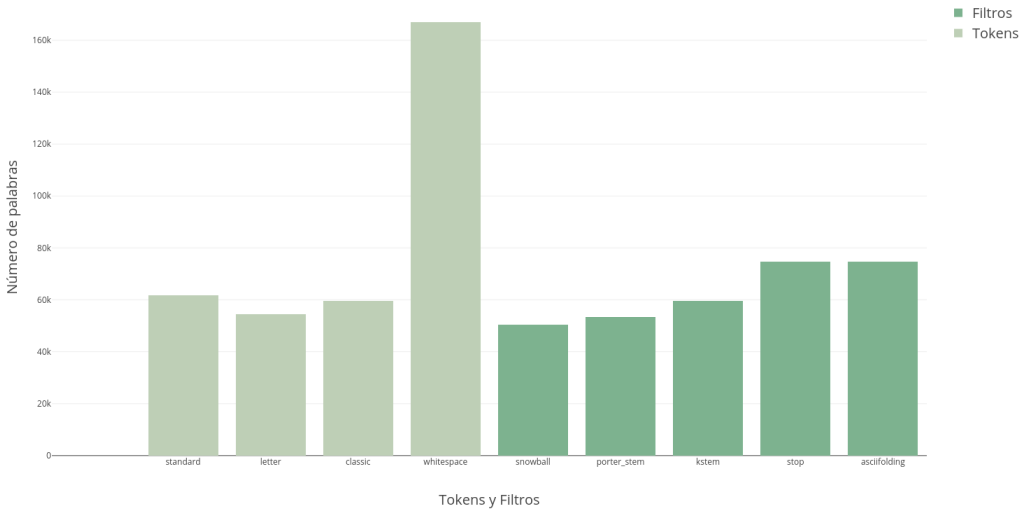


Figura 1: Gráficas del número de palabras según el filtro o token

A partir de la gráfica se puede concluir que el filtro más agresivo es *Snowball* y el token más agresivo es *Letter*.

No obstante, no solo se pueden hacer pruebas con estos dos parámetros por separado, los podemos combinar o incluso aplicar más de uno del mismo tipo a la vez. Esto es muy útil dependiendo de qué tipos de texto que se quieren comparar pero sobretodo habría que tener cuidado con el orden en que se aplican los filtros y tokens, puesto que no se obtienen los mismos resultados.

- Aplicando primero el token stop y luego el lowercase se obtienen 61823 palabras.
- Aplicando primero el token lowercase y luego el stop se obtienen 61790 palabras.

Se puede apreciar que el orden en que se ponen los filtros, es también el mismo en que se aplican. Porque al eliminar primero las stopwords y luego aplicar el lowercase, no hay ningún cambio respecto al filtro standard. En cambio, si invertimos el orden, primero estaríamos aplicando el lowercase, filtro que haría que palabras importantes como los acrónimos, sustantivos propios, ... pasen a ser stopwords y entonces, sean eliminados. Un claro ejemplo podría ser la palabra *IT* que en inglés significa Information Technologies (Campo de la informática) mientras que *it* en inglés es un pronombre.

2. Computing tf-idf 's and cosine similarity

En esta segunda parte consiste en ver que similitud hay entre dos documentos. Para ello tenemos que completar un código utilizando las formulas vistas en teoría y experimentar con ellas. La implementación de dichas formulas no supone ninguna dificultad exceptuando la interpretación que se puede hacer sobre el calculo del coseno de los documentos. Este calculo consiste en multiplicar los pesos de aquellos términos que estén en ambos documentos. Además sabemos que los documentos están ordenados alfabéticamente, por tanto podemos implementar la función de fusión del mergesort para hacer esta tarea de forma óptima dando un tiempo de $O(n)$, donde n es $min(|L_1|, |L_2|)$. El código adjunto con este informe esta comentado para entender bien las modificaciones.

2.1. Experimentación

La prueba mas básica es comprobar la similitud consigo mismo. Lógicamente el resultado será 1.0:

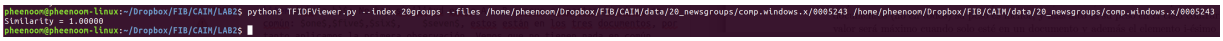


Figura 2: Ejecucción del tf-idf consigo mismo

En este primer caso no mostramos los cálculos intermedios porque compara dos vectores iguales y ocuparía mucho espacio en el documento.

Una segunda prueba consiste en ver que la implementación es correcta comparando los cálculos intermedios (lo que nos devuelve el programa) con la hoja de calculo. Para esta prueba, contamos con tres ficheros cuyo contenido es cuantas veces aparece un número que esta dentro del intervalo [1,9] y queremos ver la similitud que hay entre ellos. Para comprobar que las formulas están bien implementadas usaremos esta hoja de calculo como guía donde los cálculos están hechos a mano y comprobados que son correctos.

	one	two	three	four	five	six	seven	eight	nine	max	
a	2	1	2	0	2	2	1	0	0	2	
b	4	0	0	1	1	1	1	2	1	4	
c	3	0	2	2	1	3	1	0	0	3	
df	3	1	2	2	3	3	3	1	1		
A	0	0,7924812504	0,5849625007	0	0	0	0	0	0	Sum	Sqrt
Norm. A	0	0,6280265322	0,3421811272	0	0	0	0	0	0	0,970207659423014	0,9849911976
A / Norm. A	0	0,8045566826	0,5938758662	0	0	0	0	0	0		
B	0	0	0	0,1462406252	0	0	0	0,7924812504	0,3962406252	Sum	Sqrt
Norm. B	0	0	0	0,0213863205	0	0	0	0,6280265322	0,157006633	0,806419485669453	0,8980086223
B / Norm. B	0	0	0	0,1628499121	0	0	0	0,8824873511	0,4412436756		
C	0	0	0,3899750005	0,3899750005	0	0	0	0	0	Sum	Sqrt
Norm. C	0	0	0,152080501	0,152080501	0	0	0	0	0	0,304161001999954	0,5515079347
C / Norm.C	0	0	0,7071067812	0,7071067812	0	0	0	0	0		
										Sum (Sim total)	
sim(a,b)	0	0	0	0	0	0	0	0	0	0	
sim(a,c)	0	0	0,4199336522	0	0	0	0	0	0	0,41993365219094	
sim(b,c)	0	0	0	0,1151522771	0	0	0	0	0	0,115152277144804	

Figura 3: Captura de la demostración en excel

A continuación insertaremos tres figuras que corresponden al cálculo del $sim(a, b)$, $sim(a, c)$ y $sim(b, c)$ imprimiendo por la salida estándar los cálculos intermedios y observar que coinciden con los valores de la hoja de calculo, demostrando así su correcta implementación.

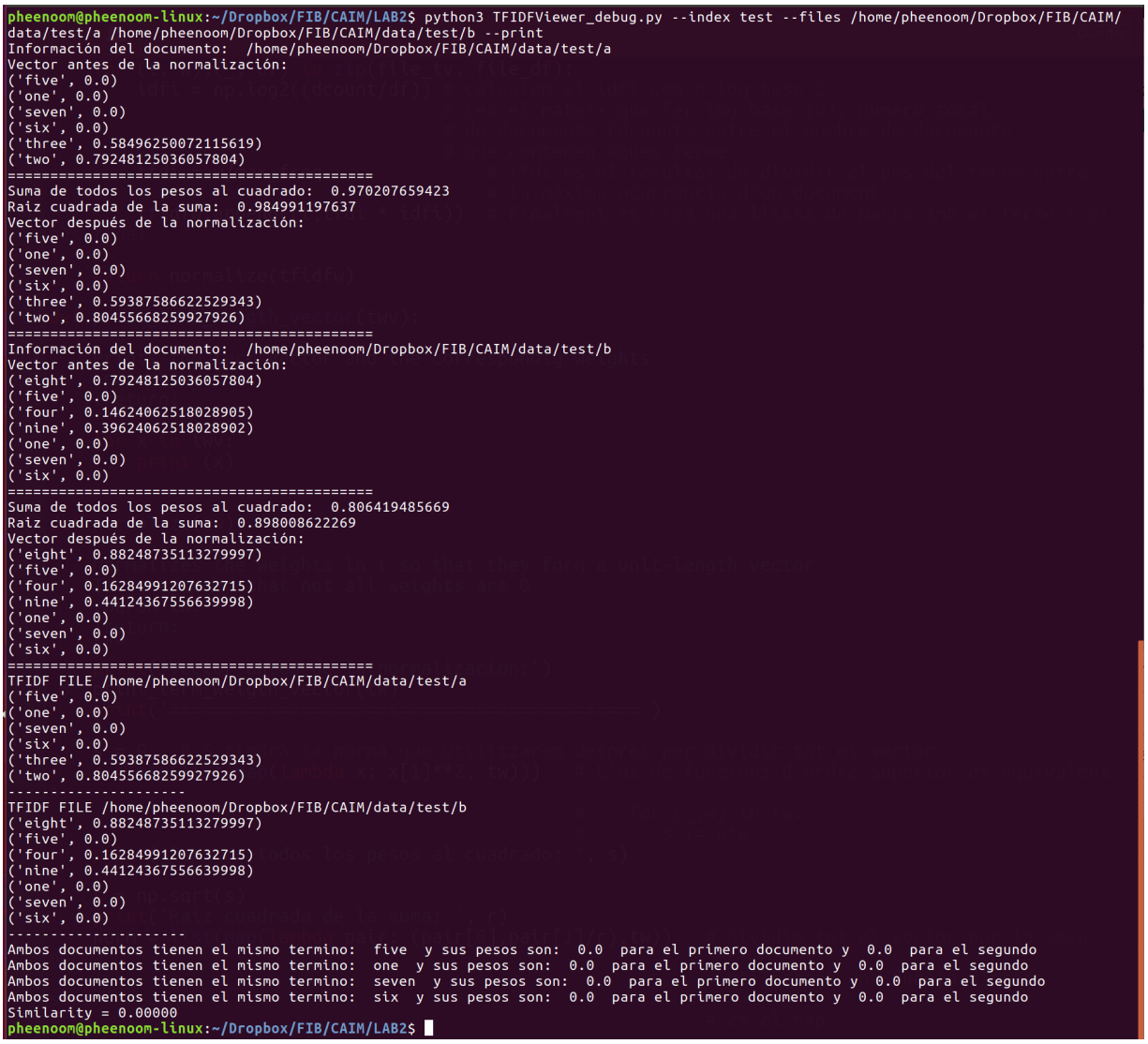


Figura 4: Ejecucción del tf-idf con el documento A y B

```
pheenoom@pheenoom-linux:~/Dropbox/FIB/CAIM/LAB2$ python3 TFIDFViewer_debug.py --index test --files /home/pheenoom/Dropbox/FIB/CAIM/
data/test/a /home/pheenoom/Dropbox/FIB/CAIM/data/test/c --print
Información del documento:  /home/pheenoom/Dropbox/FIB/CAIM/data/test/a
Vector antes de la normalización:
('five', 0.0)
('one', 0.0)
('seven', 0.0)
('six', 0.0)
('three', 0.58496250072115619)
('two', 0.79248125036057804)
==========================================
Suma de todos los pesos al cuadrado:  0.970207659423
Raiz cuadrada de la suma:  0.984991197637
Vector después de la normalización:
('five', 0.0)
('one', 0.0)
('seven', 0.0)
('six', 0.0)
('three', 0.59387586622529343)
('two', 0.80455668259927926)
==========================================
Información del documento:  /home/pheenoom/Dropbox/FIB/CAIM/data/test/c
Vector antes de la normalización:
('five', 0.0)
('four', 0.38997500048077077)
('one', 0.0)
('seven', 0.0)
('six', 0.0)
('three', 0.38997500048077077)
==========================================
Suma de todos los pesos al cuadrado:  0.304161002
Raiz cuadrada de la suma:  0.551507934666
Vector después de la normalización:
('five', 0.0)
('four', 0.70710678118654757)
('one', 0.0)
('seven', 0.0)
('six', 0.0)
('three', 0.70710678118654757)
==========================================
TFIDF FILE /home/pheenoom/Dropbox/FIB/CAIM/data/test/a
('five', 0.0)
('one', 0.0)
('seven', 0.0)
('six', 0.0)
('three', 0.59387586622529343)
('two', 0.80455668259927926)
--------------------
TFIDF FILE /home/pheenoom/Dropbox/FIB/CAIM/data/test/c
('five', 0.0)
('four', 0.70710678118654757)
('one', 0.0)
('seven', 0.0)
('six', 0.0)
('three', 0.70710678118654757)
--------------------
Ambos documentos tienen el mismo termino:  five  y sus pesos son:  0.0  para el primero documento y  0.0  para el segundo
Ambos documentos tienen el mismo termino:  one  y sus pesos son:  0.0  para el primero documento y  0.0  para el segundo
Ambos documentos tienen el mismo termino:  seven  y sus pesos son:  0.0  para el primero documento y  0.0  para el segundo
Ambos documentos tienen el mismo termino:  six  y sus pesos son:  0.0  para el primero documento y  0.0  para el segundo
Ambos documentos tienen el mismo termino:  three  y sus pesos son:  0.593875866225  para el primero documento y  0.707106781187  pa
ra el segundo
Similarity = 0.41993
pheenoom@pheenoom-linux:~/Dropbox/FIB/CAIM/LAB2$
```

Figura 5: Ejecucción del tf-idf con el documento A y C

```
pheenoom@pheenoom-linux:~/Dropbox/FIB/CAIM/LAB2$ python3 TFIDFViewer_debug.py --index test --files /home/pheenoom/Dropbox/FIB/CAIM/
data/test/b /home/pheenoom/Dropbox/FIB/CAIM/data/test/c --print
Información del documento:  /home/pheenoom/Dropbox/FIB/CAIM/data/test/b
Vector antes de la normalización:
('eight', 0.79248125036057804)
('five', 0.0)
('four', 0.14624062518028905)
('nine', 0.39624062518028902)
('one', 0.0)
('seven', 0.0)
('six', 0.0)
========================================
Suma de todos los pesos al cuadrado:  0.806419485669
Raiz cuadrada de la suma:  0.898008622269
Vector después de la normalización:
('eight', 0.88248735113279997)
('five', 0.0)
('four', 0.16284991207632715)
('nine', 0.44124367556639998)
('one', 0.0)
('seven', 0.0)
('six', 0.0)
========================================
Información del documento:  /home/pheenoom/Dropbox/FIB/CAIM/data/test/c
Vector antes de la normalización:
('five', 0.0)
('four', 0.38997500048077077)
('one', 0.0)
('seven', 0.0)
('six', 0.0)
('three', 0.38997500048077077)
========================================
Suma de todos los pesos al cuadrado:  0.304161002
Raiz cuadrada de la suma:  0.551507934666
Vector después de la normalización:
('five', 0.0)
('four', 0.70710678118654757)
('one', 0.0)
('seven', 0.0)
('six', 0.0)
('three', 0.70710678118654757)
========================================
TFIDF FILE /home/pheenoom/Dropbox/FIB/CAIM/data/test/b
('eight', 0.88248735113279997)
('five', 0.0)
('four', 0.16284991207632715)
('nine', 0.44124367556639998)
('one', 0.0)
('seven', 0.0)
('six', 0.0)
--------------------
TFIDF FILE /home/pheenoom/Dropbox/FIB/CAIM/data/test/c
('five', 0.0)
('four', 0.70710678118654757)
('one', 0.0)
('seven', 0.0)
('six', 0.0)
('three', 0.70710678118654757)
--------------------
Ambos documentos tienen el mismo termino:  five  y sus pesos son:  0.0  para el primero documento y  0.0  para el segundo
Ambos documentos tienen el mismo termino:  four  y sus pesos son:  0.162849912076  para el primero documento y  0.707106781187  par
a el segundo
Ambos documentos tienen el mismo termino:  one  y sus pesos son:  0.0  para el primero documento y  0.0  para el segundo
Ambos documentos tienen el mismo termino:  seven  y sus pesos son:  0.0  para el primero documento y  0.0  para el segundo
Ambos documentos tienen el mismo termino:  six  y sus pesos son:  0.0  para el primero documento y  0.0  para el segundo
Similarity = 0.11515
pheenoom@pheenoom-linux:~/Dropbox/FIB/CAIM/LAB2$
```

Figura 6: Ejecucción del tf-idf con el documento B y C

Es fácil que aparezcan bastantes ceros por la simplicidad del ejemplo, a continuación veremos dos observaciones que nos serán de ayuda para explicar las similitudes encontradas:

- Si un término aparece en los tres documentos, automáticamente tiene peso 0, independientemente de su número de ocurrencias, porque el Idf_i es $\log_2 \frac{D}{df_i}$ y el $\log_2 1$ es 0.
- El peso de un término viene dado por su rareza, es decir, si un término aparece en varios documentos (no en todos) su peso es menor, y más alto es su peso en el caso contrario. El valor será máximo cuando solo esté en un documento y además el elemento i-ésimo ($f_{d,i}$) sea máximo ($max_j f_{d,j}$).

Con estas observaciones podemos comprender el porqué de las similitudes dadas.

- $sim(a, b)$: No hay similitud entre A y B aunque compartan términos en común: one,$five$,six, $seven$, estos están en los tres documentos, por tanto aplicamos la primera observación. Vemos que no tienen nada en común.
- $sim(a, c)$: Estos documentos tienen una similitud aproximadamente de un 42 %, la mayor entre los documentos, en este caso aplicamos el segundo razonamiento. El único término en común con peso es three. Aunque en el texto compartan otros términos, como one, no se tiene en cuenta por la primera observación.
- $sim(b, c)$: Esta última similitud es la menor, porque el único término que comparten es four donde el documento B solo tiene una única ocurrencia.

Finalmente una última prueba seria ver qué relación puede existir entre diferentes temas. Nuestro ejemplo será comparar charlas de religión con la astrología.

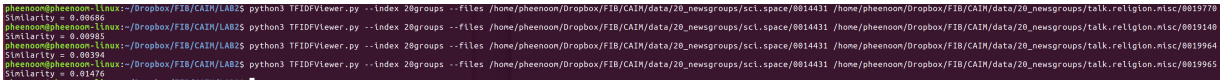


Figura 7: Ejecucción del tf-idf con el documento sci.space 0014431 con documentos de religión

Podemos observar que las similitudes entre un documento (en este caso el 0014431) con al menos 4 documentos diferentes de religión tiene una similitud máxima de un 1 %. Nos da indicios de que no hay relación entre estos dos temas.

3. Observaciones

En el desarrollo de la práctica no ha habido ninguna dificultad destacable exceptuando la mencionada en el segundo apartado. Para hacer las pruebas del primer y segundo apartado hemos pensado que juegos de pruebas o ejemplos que nos darian mas información de como funciona el ElasticSearch.

Por ejemplo, en el primer apartado mencionamos la importancia del orden de los parámetros. Descubrimos este hecho por error probando diferentes ejemplos y escribir los filtros en diferente orden. Nos dimos cuenta que para un mismo fichero y mismos filtros pero estos escritos en diferente orden nos daba un resultado diferente.

Para el segundo apartado hemos decidido buscar un ejemplo que nos diese que la similitud entre dos documentos fuera 0 y otro caso donde la similitud fuera aproximadamente del 50 %.