Session 2: Programming with ElasticSearch

Q1 2023/2024

Àlex Domínguez Rodríguez

Mario Fernández Simón

Introducción

En esta práctica estudiaremos el cálculo del peso tf-idf para representar documentos como vectores y la medida de similitud del coseno.

Comparación entre los tipos de tokens y filtros

Para estudiar los tipos de tokens y filtros utilizaremos los diferentes corpus utilizados en la sesión anterior, pero sobre todo, basaremos nuestras conclusiones en el de novelas.
Disponemos de 4 opciones distintas para "token":

- *Classic:* es el tokenizador más adecuado para la lengua inglesa.
- *Letter:* divide el texto en términos cuando encuentra un carácter que no es una letra.
- *Whitespace:* separa cuando encuentra un espacio en blanco.
- *Standard:* divide el texto en términos utilizando el algoritmo Unicode Text Segmentation. También elimina signos de puntuación.

Una vez expuesto a grandes rasgos los métodos de tokenización. Vamos a valorar los resultados obtenidos al contar los *tokens* y los términos que han generado cada uno de ellos. Hay que tener en cuenta que al no especificar el parámetro *filter* el que se usa por defecto es el *lowercase*.
Por un lado, lo que vemos claro en la *Figura 1* es que más o menos todos los métodos generan el mismo número de *tokens*, hay una diferencia menor al 2% entre lo que genera más y menos.
Por otro lado, en la *Figura 2* podemos ver lo que realmente nos interesa. En primer lugar, vemos como el método *Whitespace* es el menos agresivo al basarse solo en los espacios en blanco. *Letter* es la que genera menos términos. Eso es debido a que elimina palabras que contengan caracteres diferentes a las letras del alfabeto. Por último podemos ver como el método *Classic* y el *Standard* obtienen valores muy similares, tanto entre ellos, como con el método *Letter*. Para el estudio sobre todo nos fijaremos en *Letter*.

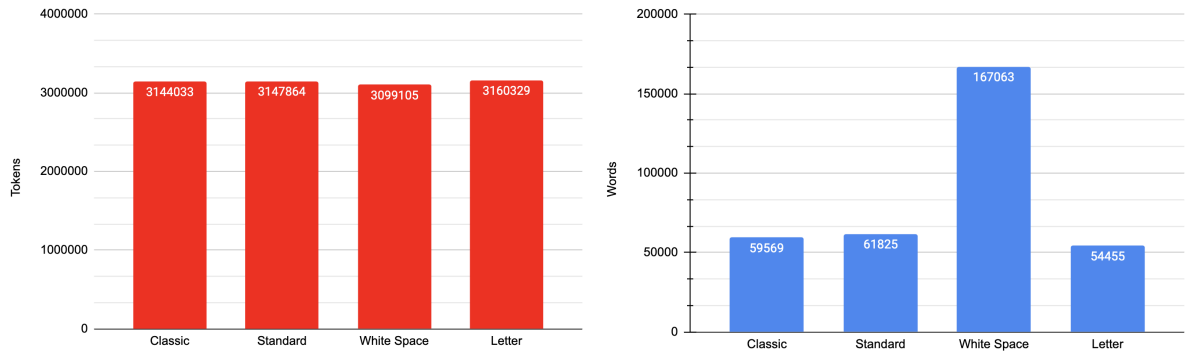

Figura 1. Gráfica Tokens y Words según el método de tokenizado

Al estudiar los ficheros de salida, nos hemos dado cuenta que la palabra más usada es “the”. A continuación, aplicaremos los filtros de distintas maneras y comentaremos los cambios relevantes.

En las siguientes figuras se han mirado todos los corpus, pero basaremos nuestras deducciones en las novelas como se comentó anteriormente.

En primer lugar, la columna azul “none” se han aplicado los filtros de “lowercase + asciifolding + stop”. Con esto conseguimos eliminar palabras con caracteres ASCII poco comunes y las stopwords que tenga registrado el filtro. Al hacer esto, la palabra más utilizada ha dejado de ser “the” y ahora es “i”.

Las otras 3 columnas implementan algoritmos diferentes cada una. Se puede observar que la que reduce más el número de palabras distintas es snowball. También vemos que *porter_stem* y *kstem* obtienen mejoras bastante similares.

Finalmente, podemos concluir que utilizando Letter como tokenizador y snowball cómo *filter* obtenemos un filtrado más agresivo.

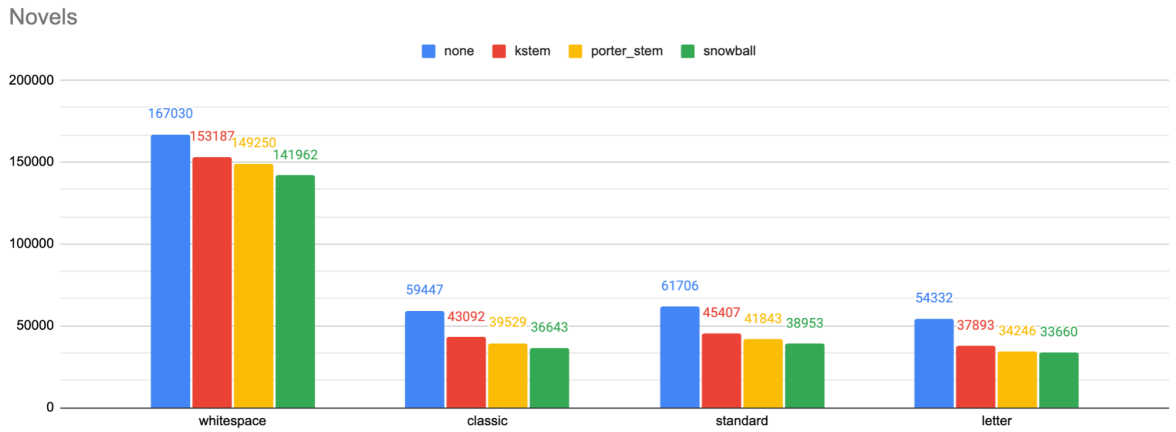

Figura 2. Comparación de técnicas de tokenizado y steamming sobre el conjunto 'novels'

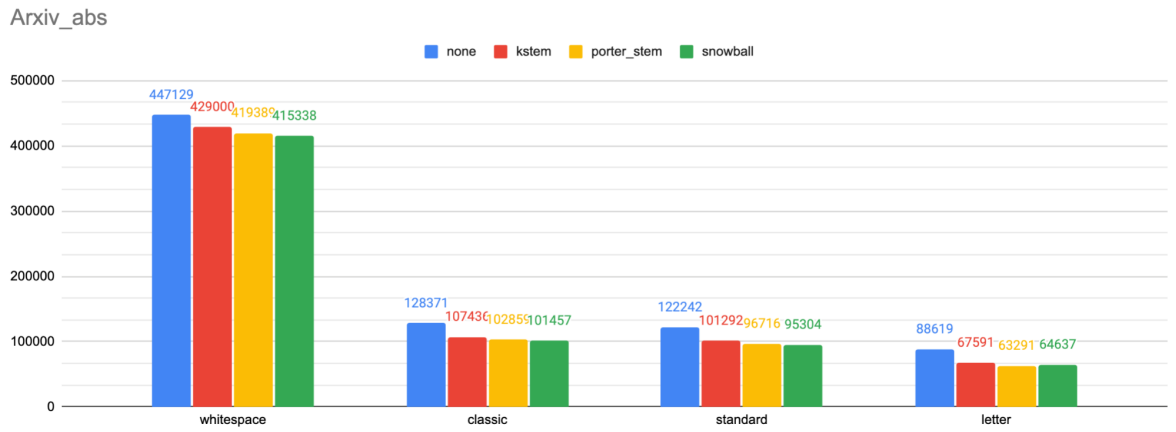

Figura 3. Comparación de técnicas de tokenizado y steamming sobre el conjunto 'arxive_abs'

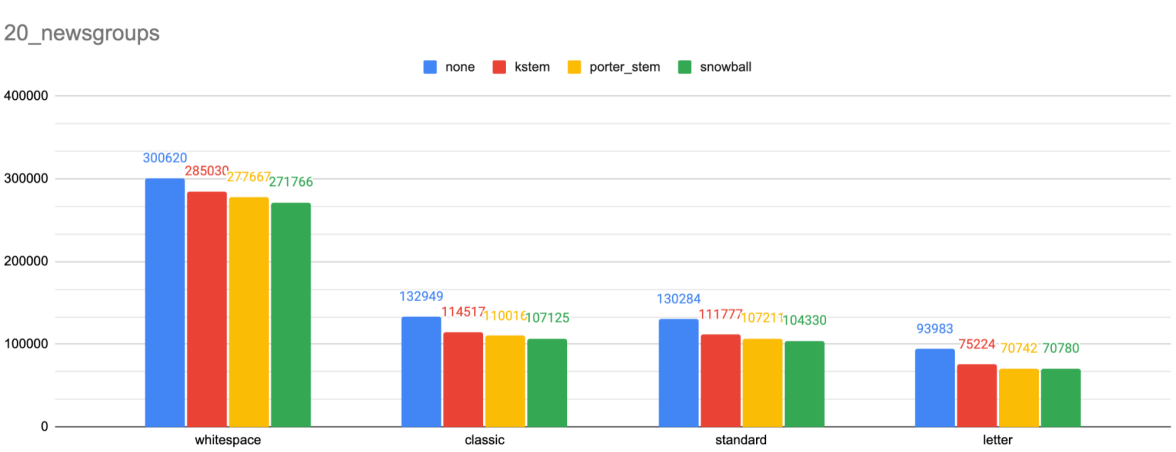

Figura 4. Comparación de técnicas de tokenizado y steamming sobre el conjunto '20_newsgroups'

Experimentación

Una vez acabada la implementación, empezaremos mirando los casos utilizados en las diapositivas de teoría, los cuales coinciden con el resultado que hemos obtenido. Además, hemos comparado un texto con sigo mismo y efectivamente obtenemos una similitud de 1.

A continuación, utilizaremos grupos de textos de *20_newsgroup* para seguir con el estudio. El primer grupo a estudiar que miraremos será el de *alt.atheism.* Concretamente, compararemos 5 documentos 2 a 2. Obtenemos la siguiente tabla:

ID DOCUMENTOS	**0000000**	**0000001**	**0000002**	**0000003**	**0000004**
0000000	1	0.36301	0.06337	0.07709	0.03164
0000001	0.36301	1	0.09038	0.13435	0.01090
0000002	0.06337	0.09038	1	0.04441	0.00688
0000003	0.07709	0.13435	0.04441	1	0.02054
0000004	0.03164	0.01090	0.00688	0.02054	1
Media	0.13377	0.14966	0.05126	0.06909	0.01749

Tabla 1. Similitud entre textos *alt.atheism.*

El segundo grupo que estudiaremos será *sci.med*. La tabla obtenida es la siguiente:

ID DOCUMENTOS	**0013000**	**0013001**	**0013002**	**0013003**	**0013004**
0013000	1	0.00484	0.01747	0.02028	0.00588
0013001	0.00484	1	0.00141	0.02563	0.00008
0013002	0.01747	0.00141	1	0.02558	0.00420
0013003	0.02028	0.02563	0.02558	1	0.01878
0013004	0.00588	0.00008	0.00420	0.01878	1
Media	0.01211	0.00799	0.01216	0.02256	0.00723

Tabla 2. Similitud entre textos *sci.med.*

La media obtenida en la Tabla 1 es de 0.08471 y en la Tabla 2 es de 0.01073. Por último, estudiaremos 2 documentos de *rec.autos* y 2 de *talk.politics.guns* entre ellos.

ID DOCUMENTOS	**Auto: 0006050**	**Auto: 0006060**	**Guns: 0011926**	**Guns: 0011939**
Auto: 0006050	1	0.06745	0.00000	0.01181
Auto: 0006060	0.06745	1	0.00266	0.01369
Guns: 0011926	0.00000	0.00266	1	0.04206
Guns: 0011939	0.01181	0.01369	0.04206	1
Media	0.02642	0.02793	0.01491	0.02252

Tabla 3. Similitud entre textos de distintos ámbitos.

Una vez acabado el análisis podemos extraer distintas conclusiones de los datos. En primer lugar, se puede observar como la media de similitud del primer grupo es comparablemente más grande que las otras 2. Esto es debido a que el tema del que tratan los textos de la Tabla 1 está mucho más relacionado entre sí que los de la Tabla 2 y lo mismo para la Tabla 3.

Por otro lado, algo que nos ha sorprendido es que no hay una diferencia muy grande entre la media de la Tabla 2 y la de la Tabla 3 pese a que esta última tiene textos mezclados. Además, hemos obtenido un valor de 0.0 lo cual tiene todo el sentido del mundo, ya que los textos son de temas distintos y pueden parecerse en nada. Este número nunca sería 0 del todo si hubiéramos dejado las *stopwords*.

Por último, al revisar los scripts proporcionados hemos encontrado el porque pese a que todos los campos se tokenizan no pasa con el path.

```
# configure the path field so it is not tokenized and we can do exact match search
client.indices.put_mapping(doc_type='document', index=index, include_type_name=True, body={
    "properties": {
        "path": {
            "type": "keyword",
        }
    }
})
```

Figura 5. Código extraído de IndexFilesPreprocess.py