Session 3: User Relevance Feedback

Q1 2023/2024

Àlex Domínguez Rodríguez

Mario Fernández Simón

Introducción

En esta práctica tendremos 2 objetivos principales. En primer lugar, programaremos en ElasticSearch un ciclo simple de “User Relevance Feedback”. En segundo lugar, experimentaremos y evaluaremos esta estrategia sobre un conjunto de documentos analizando distintos parámetros.

We will, we will Rocchio you

Después de seguir los pasos que nos marca el enunciado para implementar el archivo de python empezaremos con la experimentación, pero antes explicaremos porque hemos utilizado diccionarios en vez de vectores ordenados.
El motivo detrás de esta decisión reside en una mejor eficiencia al sumar diccionarios de gran tamaño que vectores. En el caso de los vectores, suponiendo que un vector tiene tamaño x1 y otro tiene tamaño x2, sumarlos daría un coste de O(x1+x2) y si repetimos este proceso k veces nos daría un coste de O(k * (x1 + x2)).
Por otro lado, el juntar 2 diccionarios no depende del número de elementos que haya sino del número de elementos únicos existentes. Esto origina que se reduzca drásticamente el tamaño de los elementos a juntar y más aún si son 2 textos de temas parecidos, ya que compartirán más palabras entre los 2. Por este motivo, hemos decidido utilizar diccionarios.

Experimentación

Para la experimentación iremos probando distintos valores para todos los parámetros y observaremos cómo cambia el resultado. Para el estudio hemos utilizado el corpus de *20_newsgroups/sci.med*.
Empezaremos viendo cómo cambia al tener valores de alfa y beta iguales, cuando alfa es más grande y en el caso contrario.

❖ Alfa = 1, Beta = 1, nrounds = 5, k = 4, r = 4, query = blood doctor

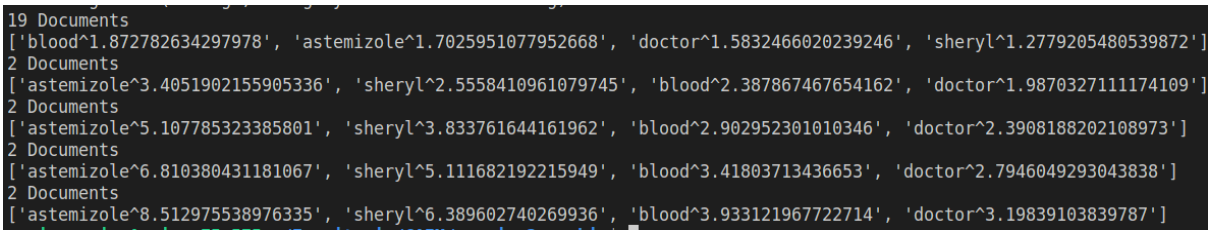


Figura 1: Resultado de consulta.

❖ Alfa = 2, Beta = 0.5, nrounds = 5, k = 4, r = 4, query = blood doctor

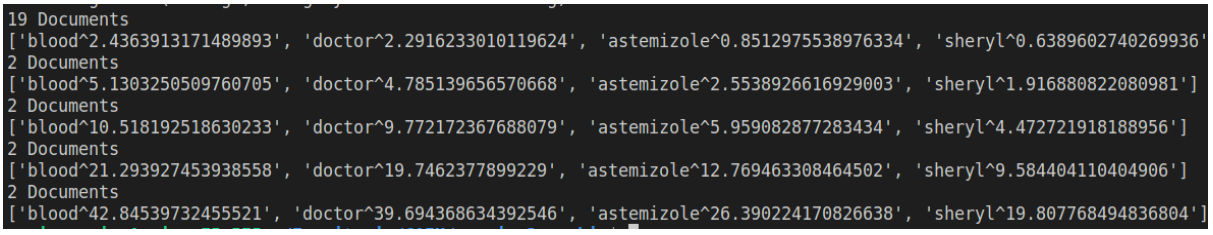


Figura 2: Resultado de consulta.

Vemos cómo al ser alfa más grande que beta, se mantiene tanto *blood* como *doctor* en las primeras posiciones, ya que estos resultados dependen más del resultado inicial y menos de los pesos calculados.

❖ Alfa = 0.5, Beta = 2, nrounds = 5, k = 4, r = 4, query = blood doctor

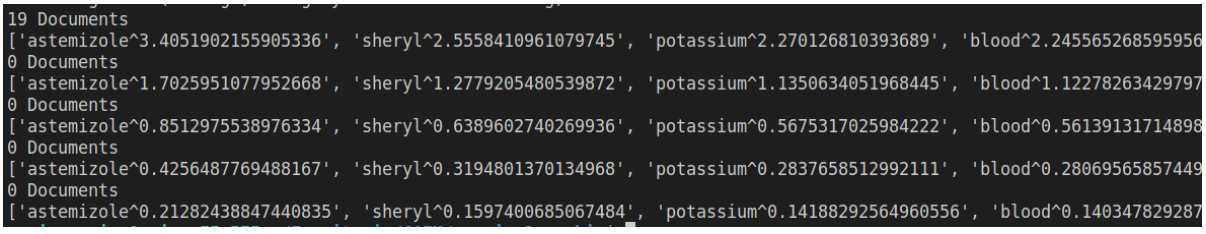


Figura 3: Resultado de consulta.

Aquí, por el contrario, vemos como *doctor* desaparece y aunque *blood* se mantiene se ve superado por otros términos como *potassium.*

❖ Alfa = 1, Beta = 0.5, nrounds = 3, k = 4, r = 4, query = blood doctor

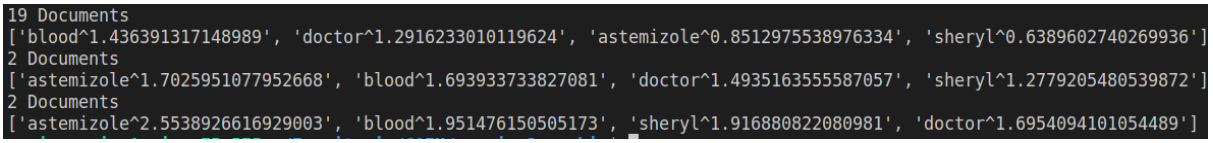


Figura 4: Resultado de consulta.

La figura anterior muestra cómo con pocas iteraciones los términos blood y doctor van perdiendo fuerza poco a poco pero se mantienen. Esto también se debe a que hemos suavizado los valores de alfa y beta con unos que hemos visto que responden mejor después de probar diferentes opciones.

- Alfa = 1, Beta = 0.5, nrounds = 7, k = 4, r = 4, query = blood doctor

Figura 5: Resultado de consulta.

Al aumentar el número de iteraciones, los términos iniciales sí que pierden importancia pero manteniendo un cierto peso.

- Alfa = 1, Beta = 0.5, nrounds = 4, k = 2, r = 4, query = blood doctor

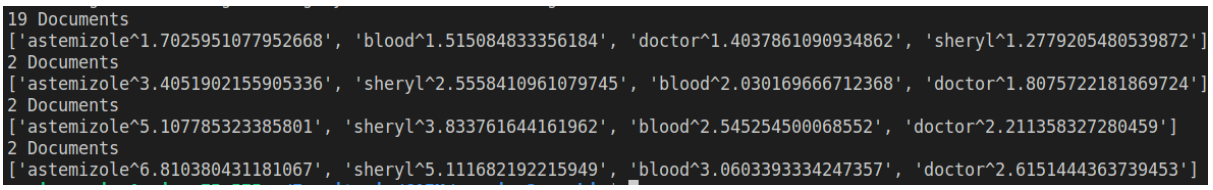


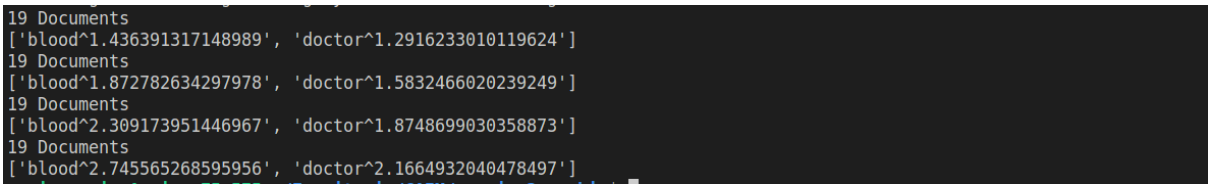
Figura 6: Resultado de consulta.

- Alfa = 1, Beta = 0.5, nrounds = 4, k = 6, r = 4, query = blood doctor

Figura 7: Resultado de consulta.

Como se deduce de la fórmula, cuando la k es pequeña los pesos de los términos en el conjunto de k vectores tienen más importancia y si con un k grande pasa al revés.

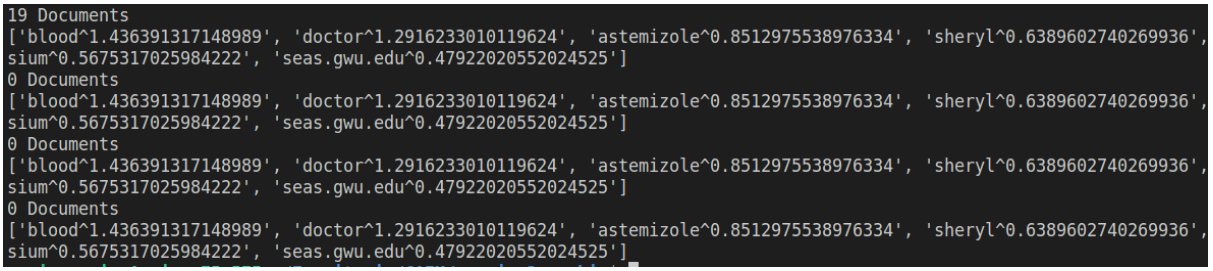
- Alfa = 1, Beta = 0.5, nrounds = 4, k = 2, r = 2, query = blood doctor



```
19 Documents
['blood^1.436391317148989', 'doctor^1.2916233010119624']
19 Documents
['blood^1.872782634297978', 'doctor^1.5832466020239249']
19 Documents
['blood^2.309173951446967', 'doctor^1.8748699030358873']
19 Documents
['blood^2.745565268595956', 'doctor^2.1664932040478497']
```

Figura 8: Resultado de consulta.

- Alfa = 1, Beta = 0.5, nrounds = 4, k = 2, r = 6, query = blood doctor



```
19 Documents
['blood^1.436391317148989', 'doctor^1.2916233010119624', 'astemizole^0.8512975538976334', 'sheryl^0.6389602740269936',
sium^0.5675317025984222', 'seas.gwu.edu^0.47922020552024525']
0 Documents
['blood^1.436391317148989', 'doctor^1.2916233010119624', 'astemizole^0.8512975538976334', 'sheryl^0.6389602740269936',
sium^0.5675317025984222', 'seas.gwu.edu^0.47922020552024525']
0 Documents
['blood^1.436391317148989', 'doctor^1.2916233010119624', 'astemizole^0.8512975538976334', 'sheryl^0.6389602740269936',
sium^0.5675317025984222', 'seas.gwu.edu^0.47922020552024525']
0 Documents
['blood^1.436391317148989', 'doctor^1.2916233010119624', 'astemizole^0.8512975538976334', 'sheryl^0.6389602740269936',
sium^0.5675317025984222', 'seas.gwu.edu^0.47922020552024525']
```

Figura 9: Resultado de consulta.

Por último, cuando tenemos una r pequeña no se tiene la posibilidad de obtener nuevos términos o ver como los que hay se van ajustando con cada ronda. En cambio, al poner una r más grande no ocurre esto. Lo que sí observamos es que a partir de la segunda iteración no encontramos ningún documento. Esto puede deberse a que quizás las palabras buscadas en la query no tienen tanta relación entre ellas.

Conclusiones

Valores Alfa y Beta:

Para valores grandes de Alfa, vemos como el resultado se relaciona más con la consulta inicial, ya que este valor se multiplica en la fórmula por el valor del resultado inicial. Si es pequeño pasa todo lo contrario.
Para valores grandes de Beta, vemos como el resultado se relaciona con el peso de los vectores tf-idf de los documentos, esto provoca que aparezcan nuevos términos con más facilidad. Si es pequeño pasa todo lo contrario.
Por lo tanto, vemos como ambos parámetros están inversamente relacionados.

Valor Nrounds:

Con valores pequeños, se observa como el resultado de la primera consulta es muy parecido al resultado final, lo cual es lógico porque se ha aplicado menos veces la regla de Rocchio. Para *nrounds* más grandes vemos como sí que cambia hasta cierta ronda, donde a partir de ahí, ya los valores apenas se modifican.

Valores R y K:

Para valores de R grandes vemos como la consulta no devuelve ningún documento, ya que la Query se trata con una AND y al tener más términos que tener en cuenta es difícil que encuentre textos apropiados y menos en temas como medicina donde hay una gran variedad de textos que no tienen nada que ver entre ellos.
Por otro lado, se aprecia como la K está muy relacionada con el recall. Debido a que cuanto más grande sea su valor más documentos se recuperarán. No obstante, vemos como en nuestros experimentos ocurre todo lo contrario, se obtienen menos documentos que con K más pequeñas. Esto puede deberse a valores de otros parámetros y los temas de los que tratan los documentos. En nuestro caso, tenemos un K = 6 y R = 4, quizás con valores más pequeños de R podríamos haber encontrado más documentos al buscar menos términos en común.