Session 6: MapReduce and Document clustering

Q1 2023/2024

Àlex Domínguez Rodríguez

Mario Fernández Simón

Introducción

En esta práctica aprenderemos principalmente dos cosas: en primer lugar, a como se usa el map-reduce de la libreria de MrJob de python y, en segundo lugar, a implementar un “document clustering algorithm” para usarlo con la finalidad de explorar un conjunto de documentos.

Implementación

En la implementación, hemos modificado los 2 documentos que nos pedían. En primer lugar, tenemos MRKmeans.py que ejecuta k veces MRKmeansStep.py y genera los outputs.
En segundo lugar, teníamos el MRKmeansStep.py, el cual cuenta principalmente con 3 funciones. La primera es la que utiliza la fórmula de Jaccard para calcular la similitud. La segunda llamada mapper calcula el cluster más cercano a un documento y devuelve ese cluster. Por último, la tercera calcula para todos los tokens del cluster su frecuencia y las devuelve ordenadas.

Experimentación

Para los experimentos empezaremos usando 5 clusters y 8 iteraciones máximas del proceso y 100 palabras. Luego aumentaremos el número de clusters, cambiaremos la frecuencia y el número de palabras.

Con 5 clusters

Frecuencia 0-10%:

El algoritmo no converge y tiene un tiempo medio por iteración de aproximadamente 6. Los clusters dan resultados dispersos, quizás por el uso de tan solo 5 de ellos.

Frecuencia 10-30%:

El algoritmo no converge y tiene un tiempo medio por iteración de aproximadamente 6.5. Vemos cómo al subir la frecuencia los clusters son parecidos entre ellos, ya que cada vez tienen palabras más comunes.

Frecuencia 30-70%:

El tiempo medio ha disminuido, en contra de nuestros pensamientos iniciales. Esta disminución puede ser debido a que como cada vez la frecuencia es más alta el algoritmo requiere de menos cálculos debido a la repetición.

Con 20 clusters y frecuencias diferentes

A priori, algo que podemos esperar es que aumente mucho el tiempo de iteración.

Frecuencia 0-10%:

Como hemos predicho anteriormente, el tiempo de iteración ha aumentado drásticamente siendo incluso casi medio minuto para algunos. Pese a ese aumento en el tiempo de cálculo, se observa cómo a partir de la 6 o 7 iteración los resultados siguen cambiando aunque más levemente lo cual nos hace pensar que quizás aumentar el número de iteraciones podría llevarnos a resultados más ajustados sacrificando tiempo.

Frecuencia 70-100%:

Ahora vemos como el algoritmo ha convergido en la tercera iteración. Esto tiene todo el sentido al aumentar tanto el número de clusters y subir tanto la frecuencia.

Con 20 clusters y número palabras = 250

Usaremos la frecuencia 10-30 ya que es la que hemos encontrado como la que tiene más sentido.

Frecuencia 10-30%:

A diferencia de antes, vemos como en la primera iteración aparecen palabras distintas entre cluster, esto seguramente sea debido al aumento del número de palabras. Otra observación, es que a partir de la 4 o 5 iteración los resultados obtenidos varían muy poco respecto a las siguientes iteraciones. Esto puede ser debido al aumento conjunto del número de palabras y el número de clusters junto a una frecuencia adecuada. Pese a estas mejoras, también vemos como el tiempo de ejecución ha aumentado. Por eso se debe valorar si el aumento en la precisión de los resultados vale la pena el sacrificar tiempo de ejecución.