CAIM Lab, Session 6:
MapReduce and Document clustering

Juan Sebastián Brito Rodríguez

David Sànchez Peña

Introducción

El objetivo de esta práctica es dividir un set de documentos en categorías basándonos en la similitud de los documentos. Para ello utilizaremos K-Means implementado con MapReduce.

Clustering documents with K-means

La primera parte de la práctica consistía en implementar el sistema de MapReduce. Para ello nos hemos basado en el esqueleto de código proporcionado.
Para calcular la similitud de dos documentos, se nos recomendaba utilizar el índice de Jaccard. Hemos decidido que la mejor manera de calcularlo era utilizando la siguiente formula:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Experimentación

Frecuencias

Para ver como afectan las frecuencias de las palabras al comportamiento del algoritmo, realizaremos varios experimentos cambiando los valores de las frecuencias máxima y mínima. Para todos los experimentos utilizaremos 10 clases y 5 iteraciones. Y para confirmar los resultados, lo ejecutaremos sobre dos corpus distintos.

Arxiv

El primer experimento lo realizamos con un vocabulario de 200 palabras. Estos son los resultados obtenidos:

Experimento	**1**		**2**		**3**		**4**		**5**		**6**	
	Min. Freq.	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**
	0.1	0.9	0.1	0.5	0.5	0.9	0.01	0.02	0.2	0.3	0.01	0.05
Tiempo	46.61s		42.26s		13.99s		38.83s		16.61s		42.17s	
Clases (de 10)	1		1		3		4		1		2	

El segundo experimento lo realizamos sobre un vocabulario de 800 palabras:

Experimento	**1**		**2**		**3**		**4**		**5**		**6**	
	Min. Freq.	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**
	0.1	0.9	0.1	0.5	0.5	0.9	0.01	0.02	0.2	0.3	0.01	0.05
Tiempo	44.43s		42.77s		13.48s		92.64s		16.23		162.26s	
Clases (de 10)	1		1		4		4		1		2	

Despues de realizar los experimentos, podemos ver que con frecuencias bajas aumenta el numero de clases. Esto es debido a que si la frecuencia es demasiado alta, los resultados serán demasiado similares, y todos acabarán formando parte del mismo cluster. Aunque cabe destacar que con el experimento número 3 ocurre algo diferente:,con frecuencias muy altas el número de clases resultante es alto, esto se debe a que con estas características solo se consiguió un vocabulario de tamaño 8.
Finalmente, hemos decidido utilizar 0,01-0,02 como rango de valores predeterminados para realizar el resto de los experimentos, ya que consideramos que es el que nos ofrece mayor equilibrio entre clases resultantes y tiempo de ejecución.

News

Realizamos los mismos experimentos con el corpus de News, con 200 y 800 palabras de vocabulario. Estos son los resultados respectivamente:

Experimento	**1**		**2**		**3**		**4**		**5**	
	Min. Freq.	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**
	0.1	0.9	0.1	0.5	0.5	0.9	0.01	0.02	0.2	0.3
Tiempo	16.81s		18.7s		7.49s		10.47s		8.16s	
Clases (de 10)	1		1		2		3		1	

Experimento	**1**		**2**		**3**		**4**		**5**	
	Min. Freq.	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**	**Min. Freq.**	**Max. Freq.**
	0.1	0.9	0.1	0.5	0.5	0.9	0.01	0.02	0.2	0.3
Tiempo	20.95s		15.39s		6.85s		5.59s		9.33s	
Clases (de 10)	2		1		1		1		1	

Mappers y reducers

Por defecto, el script utiliza dos mappers i dos reducers. Hemos modificado el script para poder cambiar estos valores y ver como afecta al tiempo de ejecución.

Primero realizamos el experimento sobre el dataset de 200 palabras con el corpus de Arxiv:

	Map	Red	Map	Red	Map	Red	Map	Red	Map	Red
Iteración	2	2	2	4	4	2	2	8	8	2
1	16.865288		16.209491		16.182425		17.254500		16.456844	
2	9.682042		9.474926		9.312292		9.505819		9.474282	
3	17.890940		18.079541		18.104679		19.840806		18.178183	
4	16.651075		16.673208		16.610684		17.092059		16.748991	
5	16.693454		17.002152		16.715179		16.943778		16.857855	

Después repetimos el experimento con el dataset de 800 palabras:

	Map	Red	Map	Red	Map	Red	Map	Red	Map	Red
Iteración	2	2	2	4	4	2	2	8	8	2
1	16.347885		17.050756		16.820460		16.621622		16.563709	
2	14.021145		14.144878		15.289024		14.050224		13.994550	
3	31.332494		32.017072		31.374404		31.704635		31.401931	
4	25.354518		23.851099		23.607943		23.556625		23.526570	
5	25.587138		23.731099		23.618890		23.655889		23.864235	

Se puede ver que cambiar el numero de mappers y reducers no influye demasiado en los tiempos de ejecución. Los tiempos son muy similares para cada iteración en

ambos datasets.

Palabras más frecuentes

Para encontrar las palabras más frecuentes en los documentos, ejecutaremos K-Means con 20 iteraciones, frecuencia mínima 0,01 y frecuencia máxima 0,02 sobre el dataset de 200 palabras. El resultado de la ejecución nos da los documentos agrupados en 2 clusters. En el primero (Clase 6) tenemos los términos *'count', 'specifi', 'look', 'protect' y 'disord'*. En el segundo cluster (Clase 7) tenemos 'trigger', 'emit', 'baryon', 'diverg', y 'neutrino'.

Dificultades

La principal dificultad que nos hemos encontrado ha sido entender como funcionaba el sistema de MapReduce, y a parte, perdimos tiempo solucionando un error que no entendíamos y que era debido a estar imprimiendo flags en pantalla.