CAIM - Lab6
MapReduce and Document clustering

Víctor Vallejo
Pau Núñez Amorós

02/12/2019

1 Generació de Datasets

Abans de poder començar a executar l'algorisme *K-means amb MapReduce (MRK)* i experimentar amb ell hem seguit una sèrie de passos per preparar l'entrada necessària. Primer de tot hem indexat el dataset `arxiv` a ElasticSearch per poder fer recompte de paraules:

```
$ python IndexFiles.py --path ../texts/arxiv --index arxiv
```

Seguidament cal extreure la informació que acabem d'indexar amb:

```
$ python ExtractData.py --index arxiv [--minfreq MIN --maxfreq MAX --numwords WORDS]
```

el qual ens genera dos arxius importants:

- `vocabulary.txt` → Que per cada token indica quants cops apareix en total
- `documents.txt` → Que per cada document indica quins tokens s'ha seleccionat

Per últim ens resta generar els prototips inicials per *K-means*, és a dir un conjunt inicial de clusters alearotirs amb certs tokens i la seva freqüència assignats (al fitxer `prototypes.txt`):

```
$ python GeneratePrototypes.py [--nclust N]
```

2 Experimentació

Una vegada acabada la generació inicial dels fitxers necessaris procedim amb el l'experimentació de l'algorisme *MRK*. Estan descomposats segons l'interés i objectiu que té cada un:

2.1 Experiment 1

En aquest primer experiment el que volem comprovar és quin efecte té la freqüència de cada token, és a dir, el ràtio d'aparició per document = $[(\#docs \mid token \in doc) \ / \ \#docs]$, sobre el temps d'execució i especialment sobre el nombre de clústers restants quan l'algorisme acaba. A la següent taula podem veure aquesta relació segons un rang de freqüències mínima i màxima que hem escollit:

	Freqüència											
iteració	min 0.01	max 0.05	min 0.05	max 0.1	min 0.1	max 0.3	min 0.3	max 0.5	min 0.5	max 0.7	min 0.7	max 1.0
1	7.24s		9.38s		14.81s		6.85s		4.68s		4.44s	
2	30.30s		32.45s		38.24s		8.62s		4.61s		3.36s	
3	33.15s		34.49s		19.31s		5.54s		4.42s		3.69s	
4	12.26s		19.94s		14.99s		4.31s		4.14s		3.57s	
5	9.88s		13.45s		15.26s		4.79s		4.04s		3.56s	
Total	85.59s		100.33s		87.80s		23.26s		17.21s		14.18s	
Mitjana	21.40s		25.10s		21.95s		5.82s		4.30s		3.54s	
#clusters	2		2		3		2		3		2	

Table 1: Acotat a les 200 paraules més freqüents

Hi ha dues observacions importants sobre aquest experiment:

Com més freqüents són les paraules al corpus, menys temps tarda l'algorisme *MRK* en acabar l'execució. Això es deu a que com més comunes són les paraules més s'assemblen entre elles, ja que són menys especialitzades o concretes d'un camp en particular. Per tant l'algorisme convergeix molt més ràpid, ja que pot assignar ràpidament una bona quantitat de tokens a un prototip.

Una cosa que no esperàvem però és que el #clusters es mantingués tan estable a mesura que augmenta la freqüència dels tokens: la nostra intuïció ens feia pensar que a menor freqüència el #clusters seria força elevat i aniria disminuïnt a mesura que les paraules són més comunes i més semblants entre elles. És possible que això es degui al factor aleatori que té la generació inicial dels prototips.

Un altre punt a destacar és la curta durada de la primera iteració de l'algorisme en general. Creiem que això es degut a que a la 1ª iteració no s'ha de comprovar la similitud de les assignacions a cada clúster amb la iteració anterior per decidir la condició de parada i això agilitza molt l'execució d'aquesta primera iteració. No hem tingut en compte aquest valor a l'hora de calcular la el *Total* i la *Mitjana* a la taula.

2.2 Experiment 2

Després del primer experiment fixem els següents paràmentres així: `Freq[min = 0.1, max = 0.3]`, és a dir treballarem amb les paraules que apareixen entre en un 10% i un 30% dels documents. El motiu de l'elecció és que aquest rang és un *sweet spot* entre paralues massa específiques i massa generals. Creiem que d'aquesta manera neutralitzem possibles desviacions en experiments posteriors.

En aquest segon experiment el que volem comprovar és quin efecte té el paràmetre `ncores`, que afecta directament el nombre de processos assignats als diferents *mappers* i *reducers*, sobre caracterísitiques com el temps d'execució, el nombre del clústers, etc

Iteració	# Words								
	100			250			500		
	# Cores								
	2	4	8	2	4	8	2	4	8
1	4.70s	10.74s	12.18s	9.80s	10.33s	10.07s	9.10s	9.02s	9.58s
2	8.96s	18.88s	21.27s	21.47s	24.39s	23.06s	21.54s	20.75s	23.12s
3	5.28s	10.70s	18.52s	10.48s	10.61s	19.23s	7.61s	10.47s	21.65s
4	3.52s	9.27s	15.48s	7.88s	8.13s	17.34s	7.39s	10.01s	19.67s
5	3.76s	9.23s	15.29s	6.48s	8.03s	17.06s	7.35s	8.17s	17.72s
Total	21.52s	48.08s	70.56s	46.31s	51.16s	76.69s	43.89s	49.40s	82.16s
Mitjana	5.38s	12.02s	17.64s	11.58s	12.79s	19.17s	10.97s	12.35s	20.54s
#clusters	1	3	6	2	3	6	2	3	6

Com podem observar a la taula resultant, el nombre de *words* seleccionat té un impacte significant en quant al temps d'execució. Això és bastant lògic degut a que augmenta el nombre de comparacions necessàries. Aquesta penalització es veu més present quan es tracta d'un conjunt més reduït de *cores*.

Sobre el número de *cores*, ens ha sorprés el marcat augment de temps d'execució a mesura que apujàvem el paràmetre `ncores`. Ens va semblar paradoxal que al intentar parel · lelitzar l'execució el rendiment baixés. Més tard vam adonar-nos que estàvem fent servir el corpus `arxiv` i que l'ordinador sobre el qual executàvem els scripts només tenia 2 cores. Això explicaria que l'augment de cores no només no ens beneficia sinó que causa un *overhead* de comunicació entre processos del programa produint un notable alentiment.

2.3 Experiment 3

Per tal de dur a terme l'última experimentació modifiquem `iter = 20`, per tal d'obtenir uns resultats més concrets hem fixat un rang de `freq = [0.01, 0.03]` i acotem `words = 250`. Com a resultat hem obtingut un total de **12 clústers**, d'entre els quals destaquem els següents pel seu grau de relació temàtica de les paraules associades a cada un d'ells:

CLASS8 = ['collaps', 'flat', 'supernova', 'neutron', 'molecul', 'deform', 'singular', 'spheric', 'fermi', 'kpc']

temàtica = Ciència

CLASS9 = ['integ', 'variant', 'mobil', 'outer', 'secur,' 'servic', 'protocol', 'digit', 'capac', 'conjectur']

temàtica = Seguretat informàtica

CLASS12 = ['kinet', 'thin', 'uv', 'kinemat', 'beam', 'astrophys', 'solid', 'plasma', 'lessim', 'thick']

temàtica = Astrofísica

No obstant, també observem que obtenim clusters que no semblen tenir gaire relació entre els elements que els formen, i no són més que una miscel · lània de termes.