

## Activitat 1: Rein

### Introducció:

Per poder executar els programes que utilitzem a continuació, és important que tinguem el programa Elasticsearch en execució:

```
##descarregar el programa en l'ordinador
https://www.elastic.co/es/downloads/elasticsearch
##comanda per localitzar el programa
whereis elasticsearch
##executar-ho en funció on estigui: cas ordinadors d'universitat:
/usr/share/elasticsearch/bin/elasticsearch
```

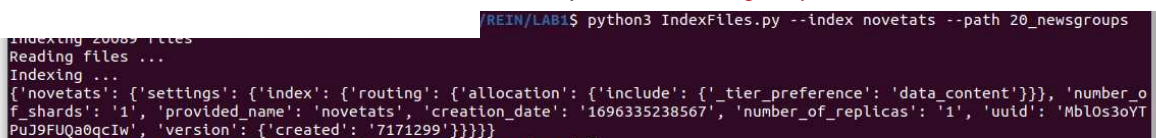
Si tot està correctament creat, a l'executar la següent instrucció no hauria de donar cap error:

```
%run elastic_test.py
```

Per poder fer els següents exercicis de l'activitat, primerament s'han d'indexar diferents documents per poder tenir un Índex on poder treballar-hi. Des de l'Elasticsearch es pot aconseguir el índex què es pot crear des de l'execució del fitxer *IndexFiles.py* donat i qualsevol col·lecció de documents.

Nosaltres tenim una carpeta anomenada *20\_newsgroups*, dins està guardat un corpus amb aproximadament 20000 documents de notícies ordenades dins de 20 carpetes, originalment creada per Ken Lang és un data set molt popular a l'hora de fer experiments en textos i en tractament de dades per la seva quantitat de textos i varietat de paraules:

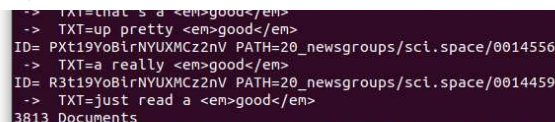
```
python3 IndexFiles.py --index novetats --path 20_newsgroups
```



```
REIN/LAB1$ python3 IndexFiles.py --index novetats --path 20_newsgroups
Indexing 20000 files
Reading files ...
Indexing ...
{'novetats': {'settings': {'index': {'routing': {'allocation': {'include': {'_tier_preference': 'data_content'}}}, 'number_of_shards': '1', 'provided_name': 'novetats', 'creation_date': '1696335238567', 'number_of_replicas': '1', 'uuid': 'Mb10s3oYT
PuJ9FUQa0qcIw', 'version': {'created': '7171299'}}}}}
```

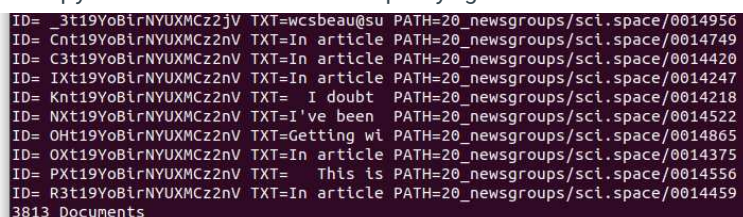
Per poder treballar en l'índex, tenim diferents fitxers per fer-hi consultes i trobar la freqüència de paraules concretes, per exemple:

```
python3 SearchIndex.py --index novetats --text good
```



```
-> TXT=that's a <em>good</em>
-> TXT=up pretty <em>good</em>
ID= Pxt19YoBirNYUXMcZ2nV PATH=20_newsgroups/sci.space/0014556
-> TXT=a really <em>good</em>
ID= R3t19YoBirNYUXMcZ2nV PATH=20_newsgroups/sci.space/0014459
-> TXT=just read a <em>good</em>
3813 Documents
```

```
python3 SearchIndex.py --index novetats --query good AND bad
```



```
ID= _3t19YoBirNYUXMcZ2jV TXT=wcsbeau@su PATH=20_newsgroups/sci.space/0014956
ID= Cnt19YoBirNYUXMcZ2nV TXT=In article PATH=20_newsgroups/sci.space/0014749
ID= C3t19YoBirNYUXMcZ2nV TXT=In article PATH=20_newsgroups/sci.space/0014420
ID= IXt19YoBirNYUXMcZ2nV TXT=In article PATH=20_newsgroups/sci.space/0014247
ID= Knt19YoBirNYUXMcZ2nV TXT= I doubt PATH=20_newsgroups/sci.space/0014218
ID= NXt19YoBirNYUXMcZ2nV TXT=I've been PATH=20_newsgroups/sci.space/0014522
ID= Oht19YoBirNYUXMcZ2nV TXT=Getting wi PATH=20_newsgroups/sci.space/0014865
ID= Oxt19YoBirNYUXMcZ2nV TXT=In article PATH=20_newsgroups/sci.space/0014375
ID= Pxt19YoBirNYUXMcZ2nV TXT= This is PATH=20_newsgroups/sci.space/0014556
ID= R3t19YoBirNYUXMcZ2nV TXT=In article PATH=20_newsgroups/sci.space/0014459
3813 Documents
```

Amb el fitxer donat *CountWords.py*, es pot saber la freqüència de cada paraula per després poder crear un fitxer .csv i guardar l'índex (ordenats per freqüència o alfabèticament):

```
python3 CountWords.py --index novetats> index_novetats.csv
```

1	1	aaaaarrgh	128628	345	existing	130284	75164	is
2	1	appeased	128629	345	functions	130285	86774	in
3	1	exasperatingly	128630	347	ready	130286	101530	and
4	1	institutionalize	128631	347	potential	130287	107310	a
5	1	interruptions	128632	347	escape	130288	116233	of
6	1	kickoff	128633	347	external	130289	129475	to
7	1	mzimmersc5sllk.lid9	128634	347	meg	130290	257240	the
8	1	nuisances	128635	348	isa	130291	-----	
9	1	cardinal's	128636	348	uk	130292	130290	Words

```
python3 CountWords.py --index novetats --alpha> index_novetats_alfa.csv
```

1	5785		88315	1	nyankees	130282	1	erale
2	11		88316	19	nz	130283	1	ete
3	1		88317	1	nz12	130284	1	ialittin
4	1		88318	2058	o	130285	4	n
5	8		88319	4	o'bedlam	130286	2	naustin
6	1		88320	13	o'brien	130287	1	u
7	1		88321	23	o'casey	130288	22	p
8	87	0.0	88322	60	o'clock	130289	1	y
9	49	0.00	88323	1	o'connell	130290	1	yhooked
10	10		88324	23	o'connor	130291	-----	
11	5	0.0005	88325	1	o'connor's	130292	130290	Words
			88326	1	o'd			
			88327	3	o'dea			

### Llei de Zipf:

$$f(r) = \frac{k}{r^\alpha}$$

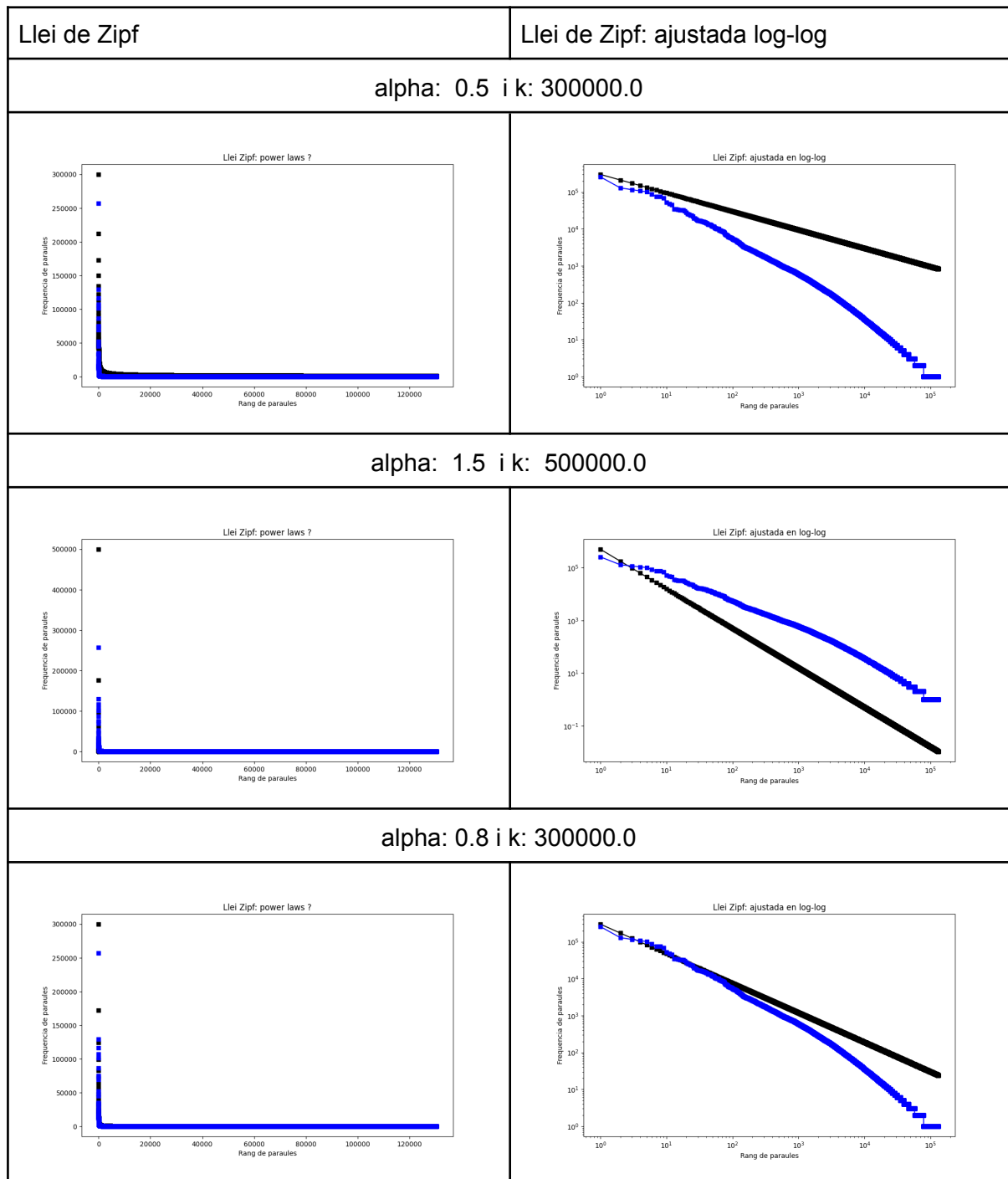
Sent f la freqüència de la paraula en el corpus i r el rang que ocupa la paraula en la llista de l'índex, hem d'estimar els paràmetres alpha i K i descriure la distribució de rang-freqüència de les dades, per comprovar que al fer una gràfica freqüència respecte rang es produeix una *power-law* tal com prediu la llei.

Per això hem creat un Script de Python nou anomenat *llel\_zipf.py* on es crea el gràfic segons les variables estimades donada per la funció *curve\_fit* amb un set precís de k i alpha i després on l'usuari pot entrar un parell d'alpha i k es corbes decreixents i també aquesta mateixa ajustada al log-log.

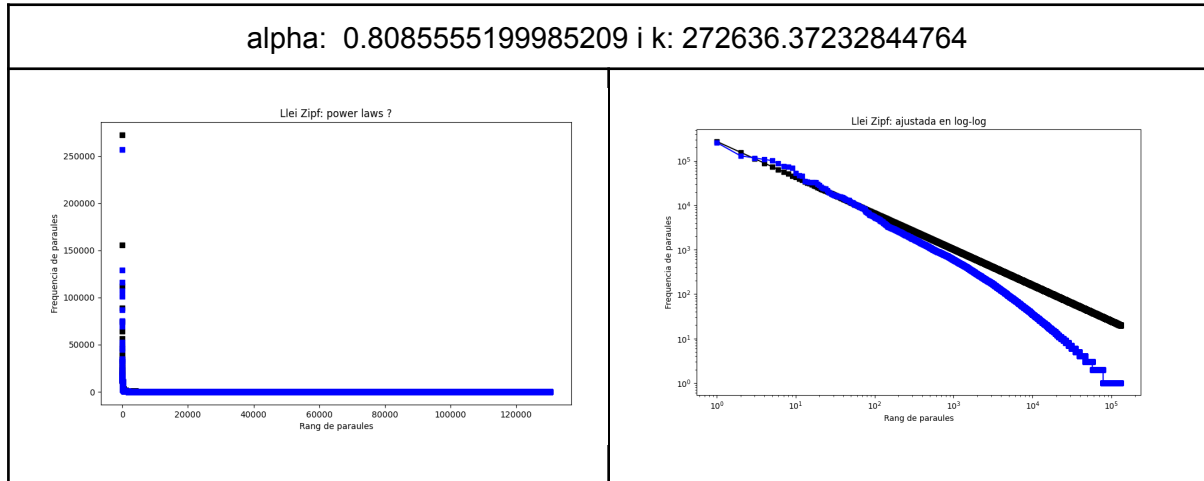
En quadrats negres són les freqüències que s'estimen segons la llei i en quadrats blaus la freqüència que realment està en el corpus executant:

```
python3 llel_zipf.py index_novetats.csv
```

Provant diferents alphas i K:



Si s'aplica l'alpha i la K calculada per la funció *curve\_fit*:



### Prediccions:

#### Paràmetres estimats:

k= 300000  
alpha= 0,8

#### Paràmetres reals:

k= 272.636,37  
aplha= 0,808555520

rang	freqüència estimada	freqüència real
57	11814,66387	10371,97554
345645	11,12287436	9,063453387
1000324	4,753447524	3,838282326
9844110	0,7630976012	0,6042440232
4492000	1,429451605	1,139506774

Si provem els nostres paràmetres per trobar la freqüència i la comparem amb la freqüència real, podem veure que varia molt poc, igualment que la freqüència sigui un valor diferent, els nostres paràmetres estimats s'ajusten molt als paràmetres reals.

### Llei de Heaps:

$$\log d = k_1 + \beta * \log N \Rightarrow d = k * N^\beta \text{ (on } k_1 = \log k)$$

Sent d el nombre de paraules diferents en tot el corpus i N el número total de totes les paraules, pels paràmetres  $\beta$  i K hem de trobar els paràmetres per descriure el nombre de termes diferents segons la llei de Heaps.

Per fer més manejable un corpus molt gran, es pot dividir en diferents subcorpus, com hem fet a continuació tenint 4 carpetes de documents:



Indexar cada carpeta per crear un índex propi i amb el fitxer *CountWords.py* i guardar en els seus corresponents .csv:

```
marc@marcperezg:~/Escritorio/REIN/lab1/1REIN-lab$ python3 IndexFiles.py --index textos1 --p
ath /home/marc/Escritorio/REIN/lab1/1REIN-lab/textos1/
Indexing 1998 files
Reading files ...
Indexing ...
{'textos1': {'settings': {'index': {'routing': {'allocation': {'include': {'_tier_preferenc
e': 'data_content'}}}, 'number_of_shards': '1', 'provided_name': 'textos1', 'creation_date'
: '1696361857490', 'number_of_replicas': '1', 'uuid': 'njd_dxNKQxlR2CMbcFAfSg', 'version':
{'created': '7171299'}}}}}}
marc@marcperezg:~/Escritorio/REIN/lab1/1REIN-lab$ python3 IndexFiles.py --index textos2 --p
ath /home/marc/Escritorio/REIN/lab1/1REIN-lab/textos2/
Indexing 3930 files
Reading files ...
Indexing ...
{'textos2': {'settings': {'index': {'routing': {'allocation': {'include': {'_tier_preferenc
e': 'data_content'}}}, 'number_of_shards': '1', 'provided_name': 'textos2', 'creation_date'
: '1696361929044', 'number_of_replicas': '1', 'uuid': 'AAMffIvzQKuokRkYct00GA', 'version':
{'created': '7171299'}}}}}}
marc@marcperezg:~/Escritorio/REIN/lab1/1REIN-lab$ python3 IndexFiles.py --index textos3 --p
ath /home/marc/Escritorio/REIN/lab1/1REIN-lab/textos3/
Indexing 5888 files
Reading files ...
Indexing ...
{'textos3': {'settings': {'index': {'routing': {'allocation': {'include': {'_tier_preferenc
e': 'data_content'}}}, 'number_of_shards': '1', 'provided_name': 'textos3', 'creation_date'
: '1696361941930', 'number_of_replicas': '1', 'uuid': 'IRck44wSSCmd-wte6I60IA', 'version':
{'created': '7171299'}}}}}}
marc@marcperezg:~/Escritorio/REIN/lab1/1REIN-lab$ python3 IndexFiles.py --index textos4 --p
ath /home/marc/Escritorio/REIN/lab1/1REIN-lab/textos4/
Indexing 8273 files
Reading files ...
Indexing ...
{'textos4': {'settings': {'index': {'routing': {'allocation': {'include': {'_tier_preferenc
e': 'data_content'}}}, 'number_of_shards': '1', 'provided_name': 'textos4', 'creation_date'
: '1696361941930', 'number_of_replicas': '1', 'uuid': 'X6jmaGkTRP-Ejy-NRMJX0g', 'version':
{'created': '7171299'}}}}}}
marc@marcperezg:~/Escritorio/REIN/lab1/1REIN-lab$
```

```
marc@marcperezg:~/Escritorio/REIN/lab1/1REIN-lab$ python3 CountWords.py --index textos1 > textos1.csv
marc@marcperezg:~/Escritorio/REIN/lab1/1REIN-lab$ python3 CountWords.py --index textos2 > textos2.csv
marc@marcperezg:~/Escritorio/REIN/lab1/1REIN-lab$ python3 CountWords.py --index textos3 > textos3.csv
marc@marcperezg:~/Escritorio/REIN/lab1/1REIN-lab$ python3 CountWords.py --index textos4 > textos4.csv
```



Executar un nou Script de Python anomenat *lle\_i\_heaps.py*, on llegeix tots els fitxers anomenats "textosX.csv", i hem fet que retorni això:

- Array del total de paraules trobades per cada carpeta de textos
- Array del nombre de paraules diferents per cada carpeta de textos
- Els paràmetres K i B estimats per la funció *curve\_fit*

```
/REIN/LAB1$ python3 llei_heaps_bona.py  
Total de paraules: [210649, 4267902, 839873, 1138622]  
Última paraula diferent: [12125, 72653, 48181, 47706]  
Estimat per K: 171.1783207519134 //Estimat per B: 0.3990444800633632
```

Si s'expandeix el corpus, segons la llei de Heaps, no hauria de pujar massa el nombre de paraules diferents, ja que si es té un índex prou complet és cada vegada més difícil trobar paraules que encara no estiguin. Per comprovar-ho nosaltres fem un índex nou amb nous documents:

```
/REIN/LAB1$ python3 llei_heaps_bona.py  
Total de paraules: [210649, 4267902, 839873, 1138622, 3147864]  
Última paraula diferent: [12125, 72653, 48181, 47706, 61825]  
Estimat per K: 194.14296166801012 //Estimat per B: 0.38879011510733
```

Amb la primera mostra de 4 col·leccions de textos ens dona: Estimat per K: 194.14296166801012 i Estimat per B: 0.38879011510733, i ara amb una col·lecció més ens dona Estimat per K: 194.14296166801012 Estimat per B: 0.38879011510733

En el nostre cas, quasi no varia perquè ja s'han indexat moltes paraules dels documents, però per qualsevol paraula nova trobada es modifica els valors, exemple que corrobora la llei de Heaps, ja que els paràmetres no tenen un canvi molt dràstic després de que se li afegixi molts més documents indexats.