Alexandru-Ilie Popa
Pol Casacuberta Gil

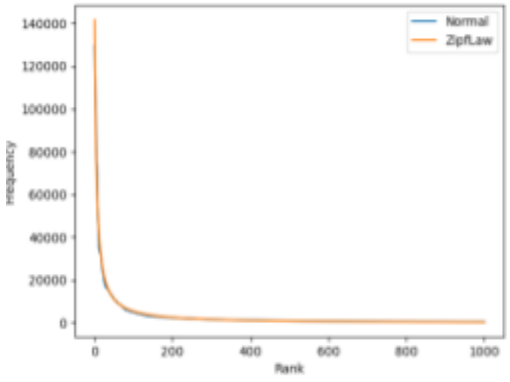
Indexes i queries

En primer lloc, indexem cada conjunt de dades (20 newsgroups, arxiv abs, novels) amb un índex diferent (en concret: news, abs i novels). Després fem servir l'script *CountWords.py* per obtenir els fitxers (words_news, words_abs, words_nov) que conté les paraules i les freqüències dels conjunts de dades.
Ara generem la llista de paraules ordenades lexicogràficament associada a cada fitxer que hem obtingut prèviament, traient els strings que no siguin estrictament paraules. Per aconseguir això, utilitzem l'script *filter_words.py* que hem creat i n'extraiem en un csv (un per cada fitxer: data_novels, data_words_abs, data_words_news) les dades de les paraules amb les corresponents freqüències.

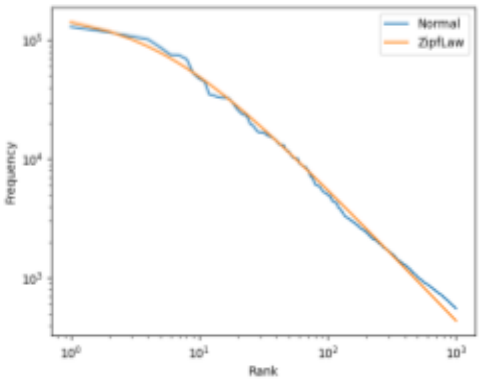
Un cop completat el preprocessat de les dades passem a analitzar si compleixen la llei de Zipf, $f = \frac{c}{(rank+b)^a}$ ajustant els paràmetres perquè la fórmula descrigués les dades de la millor manera possible.
Per a l'anàlisi hem escrit l'script ZipfLaw.py. Aquí mitjançant la llibreria *scipy* de Python fem servir el mètode *curve_fit* per ajustar la funció f, a les dades.
Hem deixat que *Python* ajusti els valors de cada variable. Per aconseguir això, hem posat en el mètode *curve_fit* els següents límits per a(0,5 a 2,0), b(-500000,0 a 500000,0) i respectivament c(-500000,0 a 500000,0).
En consideració al nombre de paraules, primer hem provat amb tot el conjunt de dades, però la gràfica no s'ajustava al que esperàvem trobar. Hem provat de variar el nombre de paraules amb què fem els experiments i hem observat que els paràmetres varien en funció del nombre de paraules que fem servir. Per aquesta raó hem decidit mostrar els gràfics amb les 1000 paraules més freqüents.
Els nostres resultats es veuen reflectits en els dos tipus de gràfiques següents, que corresponen a cada conjunt de dades:

En les següents gràfiques i valors donats als paràmetres, podrem observar una peculiaritat dels registres de la llengua emprats en els textos. En la base de dades de 20_newsgroups el registre és més aviat col·loquial, degut a això es fan servir més paraules repetides, i menys de diferents, podem veure que el paràmetre "a" s'allunya bastant del valor 1. Si ens fixem en les dades d'arxiv_abs en el que es fa servir un llenguatge científic, es fan servir més paraules diferents, ja que utilitzen més paraules tècniques i si mirem a l'altra banda de l'espectre, en l'àmbit literari en la base de dades novels, en el que es busca de forma deliberada la bellesa del llenguatge i la varietat i riquesa d'aquest, la repetició de paraules és encara menor.

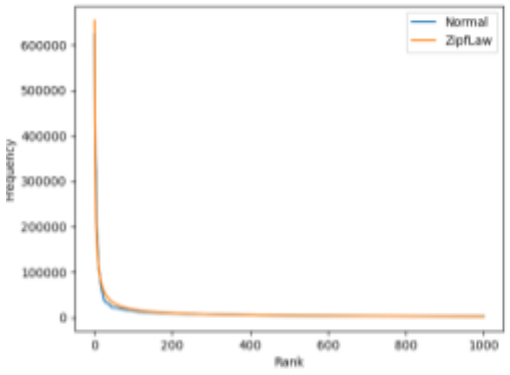
Alexandru-Ilie Popa
Pol Casacuberta Gil



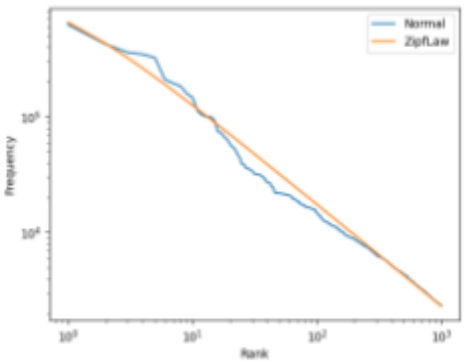
20_newsgroups no log



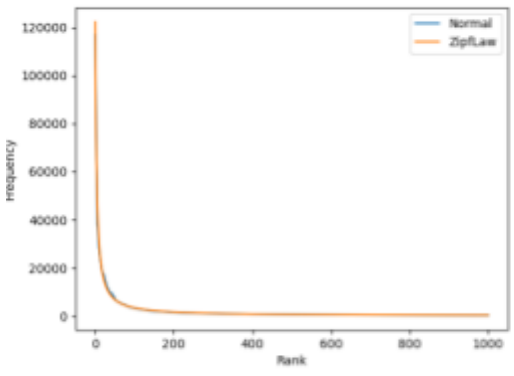
20_newsgroups log



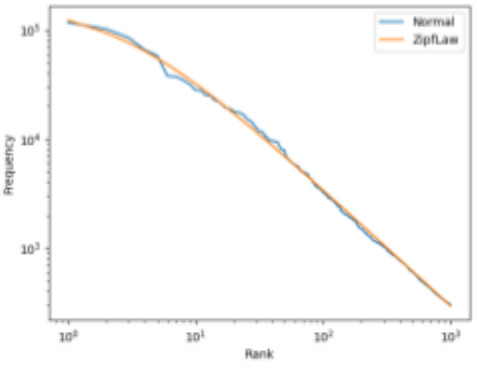
arxiv_abs no log



arxiv_abs log



novels no log



novels log

Els valors dels paràmetres són els següents:

20_newsgroups: a = 1,228118, b = 6,133342, c = 1547738,615658
arxiv_abs: a = 1,117946, b = 2,180405, c = 2271015,793695
novels: a = 1,071542, b = 2,600829, c = 482506,443516

Amb aquests experiments hem pogut concloure que la distribució

rang-freqüència segueix una llei potencial, en concret la llei de Zipf i que els fitxers de text compleixen aquesta llei.

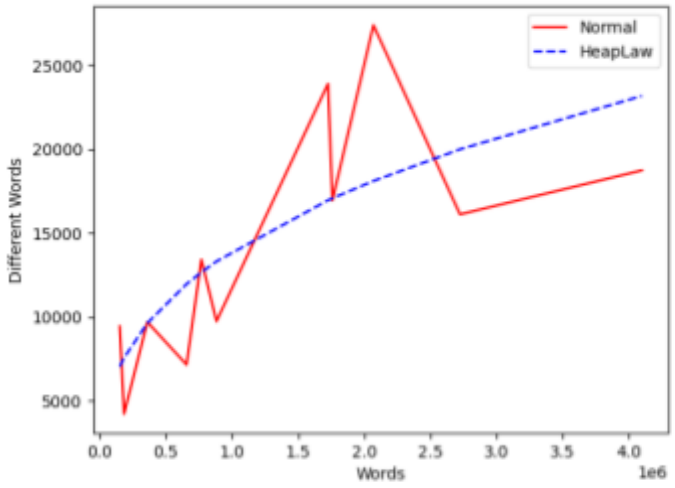
Heap's law

Per a aquest experiment, en primer lloc, vam intentar separar els llibres en 8 arxius diferents i indexar aquests 8 arxius de cada llibre en índexs diferents, extraient la informació necessària per a gràfica els resultats. Després vam pensar a ordenar els llibres per la mida de cada arxiu, fem grups de 3 llibres dels 33 llibres totals i ens queden 11 grups ordenats en ordre creixent per mida.
Després passem a indexar cada grup. Els índexs són números sencers positius de 0 a 10, així l'índex 0 fa referència als 3 fitxers més petits del conjunt de dades novels, l'1 als següents 3 fitxers que són més grans que els de l'índex 0, així fins a l'índex 10 que són els 3 arxius que tenen la mida més gran. Tot aquest preprocès ho hem fet amb l'script *Split_and_Index_Novels.py.*
A *HeapLaw.py* hem adaptat el codi de *CountWords.py* per comptar mitjançant l'elastic search el nombre de paraules totals i el nombre de paraules diferents de cada índex que prèviament hem afegit a la base de dades.
En tenir totes les dades ara passem a comprovar si es compleix la llei de Heap:

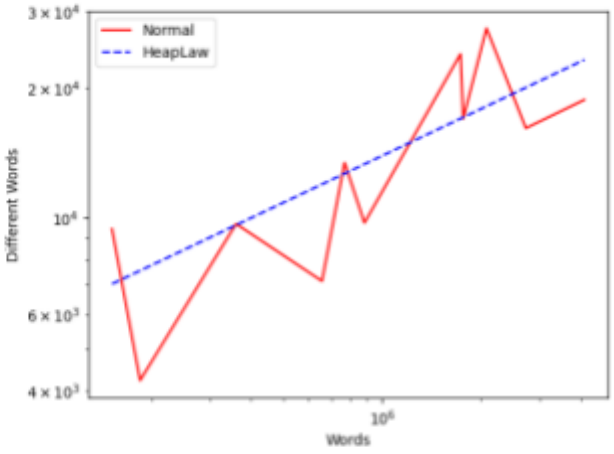
$$d = k * N^{\beta}$$

d es el número de paraules diferents que té el text
N: és el nombre total de paraules al text.
K,Beta: paràmetres de la funció.

Per trobar els valors de K i de Beta fem servir el mètode *curve_fit*, amb els següents rangs: K de 0 a 1000000.0, Beta de 0.1 a 2.



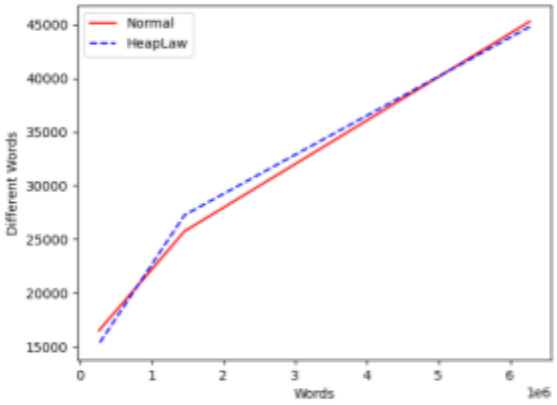
novels no log



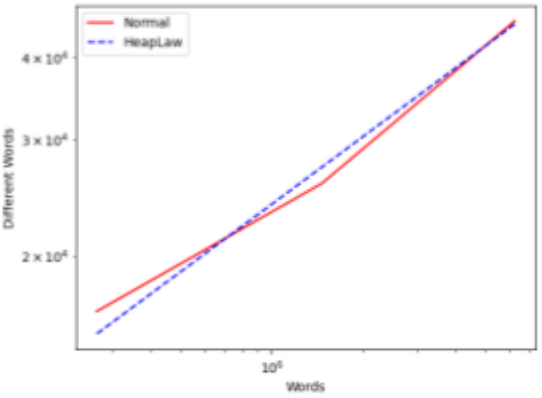
novels log

k = 93.972419, B = 0.361779

En aquestes darreres gràfiques observem resultats que no necessàriament són els que esperàvem, però que parlen del conjunt de dades amb el que estem tractant. Esperaríem, en condicions normals que amb un augment de les paraules, el nombre de paraules diferents augmenti, no obstant no observem això en les dades en tots els punts. Això pot ser degut a diverses coses com que agrupem els llibres de diferents autors en un mateix índex per a calcular-ne les paraules diferents o a causa del fet que hi ha poesia i prosa en un mateix índex, també podria ser degut al simple fet d'agrupar llibres diferents en comptes de comptar-los d'un en un.



novels no log



novels log

k = 115.074104, B = 0.374954

Finalment, hem provat de reduir el nombre d'índexs de manera que mantenim més informació en cada un dels índexs, d'aquesta manera veiem que les dades es comporten de manera prou semblant a la de la llei de Heap.