CAIM - Lab1
ElasticSearch and Zipf's and Heaps' laws

Pau Núñez Amorós
Víctor Vallejo Rives

30/09/2019

1 Zipf's Law

En aquest apartat hem de comprovar si la *rank-frequency distribution* amb un text suficientment gran segueix una *power law*. En concret ens centrem en la *Zipf's Law*, definida per la funció $f = \frac{c}{(rank+b)^a}$ a la qual ajustarem els paràmetres `a`, `b` i `c` per tal d'apropar-nos al màxim possible a la *rank-frequency distribution*.

Com a *dataset* hem fet servir el **novels** proporcionat. Hem dotat el script `countWords.py` amb les modificacions adients per tal que només accepti paraules vàlides i no es quedi amb números i altres caràcters incorrectes. Al redireccionar la sortida hem obtingut un fitxer amb el número d'aparicions d'una paraula en el global de novel · les, seguit de la paraula com a tal. Al final d'aquest fitxer ens indica quantes són les paraules en total, en el nostre cas 56.583. Aquesta última informació l'hem esborrat per tal d'obtenir el fitxer amb el mateix format a totes les seves línies. Com que l'*script* ens donava les paraules ordenades ascendentment per número d'ocurrencies, només ha calgut llegir en ordre invers per tal de relacionar-les amb el seu *rank*.

Amb això ja hem pogut visualitzar la *rank-frequency distribution* i hem obtingut una corba decreixent. Observem que les primeres poques paraules en *rank* tenen una freqüència enormement més elevada que tot el conjunt que les segueix, que té una baixa freqüència molt similar.

Per comprovar que segueix una *power law*, hem d'optimitzar els paràmetres `a`, `b` i `c` de la funció de la *Zipf's Law*. Per tal de fer això, hem utilitzat el mètode `curve_fit` de la llibreria de python `scipy`. Primerament vam intentar que ajustés els tres paràmetres de forma automàtica, però ens vam trobar amb problemes relacionats amb la variable `a` així que vam optar per assignar-li valor nosaltres i que `curve_fit` només ens determinés les `b` i `c` òptimes. Així doncs vam declarar una `a = 1` i vam obtenir que els millors valors en relació eren `b = 0.774` i `c = 360913.113`.

Al dibuixar les dues corbes resultants, hem obtingut la següent gràfica:

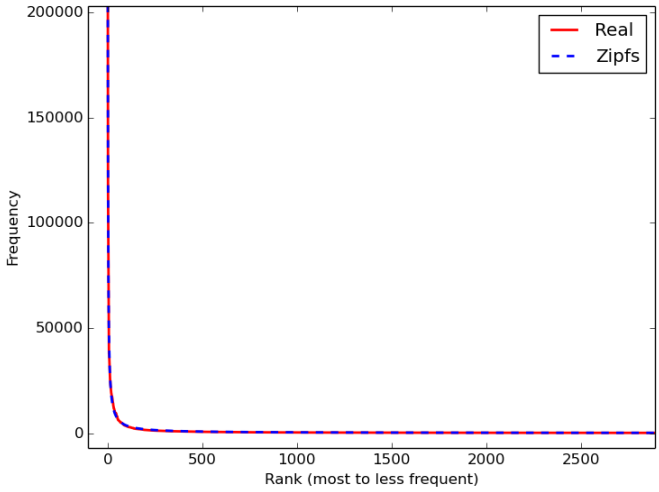

Figure 1: *Power law plot*: freq real i freq Zipf's *fitted*

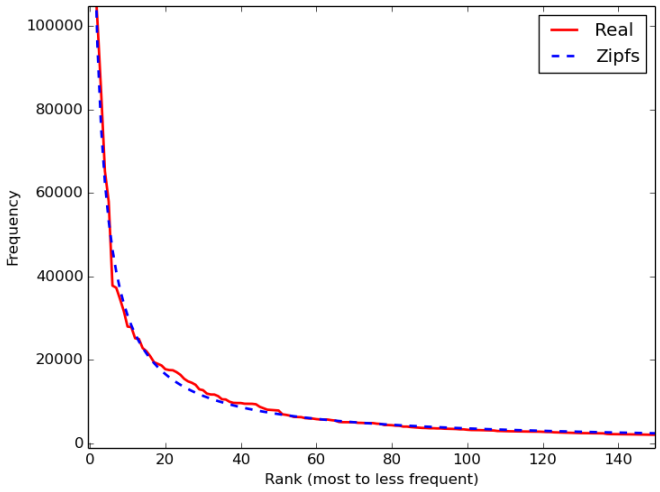

Figure 2: Zoom de la corba de la Figure 1

Com podem observar, els valors que hem assignat a les variables `a`, `b` i `c` han aconseguit que la funció de Zipf's s'aproximi de forma molt precisa a la *rank-frequency distribution*. En el nostre cas hem fet servir totes les paraules obtingudes del dataset, i són suficients per a determinar que la *rank-frequency distribution* segueix la *Zipf's Law*.

També hem volgut comprovar si aquests paràmetres ens permetien obtenir bons resultats quan es tracta d'una escala log-log, en comptes de la lineal anterior. Hem obtingut la següent gràfica:

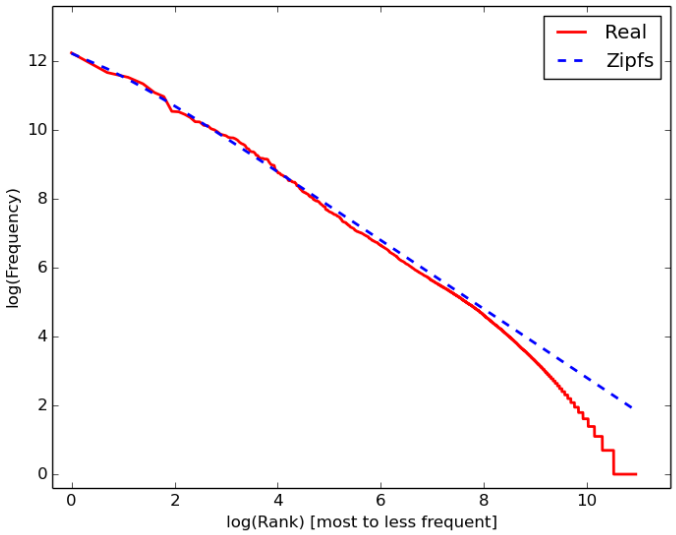

Figure 3: *Power law plot*: log-log Scale

Com mostra la figura, segueix ajustant prou bé el model, a excepció de l'últim tram, al qual la *rank-frequency distribution* pateix una disminució abrupta de valors que fa que s'allunyi de la predicció. Creiem que això és degut a la sintetització de les dades, ja que ara obtenim els valors en un rang molt menor al aplicar els logaritmes que fan que es visibilitzin més les diferències amb les paraules de menor *rank* que al cas de l'escala lineal passaven molt més desapercebuts per la distància tan gran entre la freqüència de paraules als alts i baixos *ranks*. Tot i així considerem que segueix acompanyant a la conclusió donada anteriorment.

2 Heaps' law

Per la llei de Heap cal veure si donat un cos de text amb N paraules es compleix que:

$$d = kN^{\beta}$$

$$d = \#paraules\ diferents,$$
$$k\ i\ \beta\ paràmetres\ lliures$$

Es recomana que indexem a *ElasticSearch* diversos índexs amb quantitats de text variables. Nosaltres hem creat 5 entrades a la base de dades i cada una conté ∼4MB menys de text que l'anterior entrada.

De la mateixa forma, per tots els índexs hem netejat el *pool* de paraules, quedant-nos només amb les reals i hem calculat `N` i `D`.

Tal i com hem fet amb Zipf hem usat `scipy.curve_fit` per ajustar els valors k i β al *plot* de les dades reals.

Hem obtingut $k = 83.129$ i $\beta = 0.437$

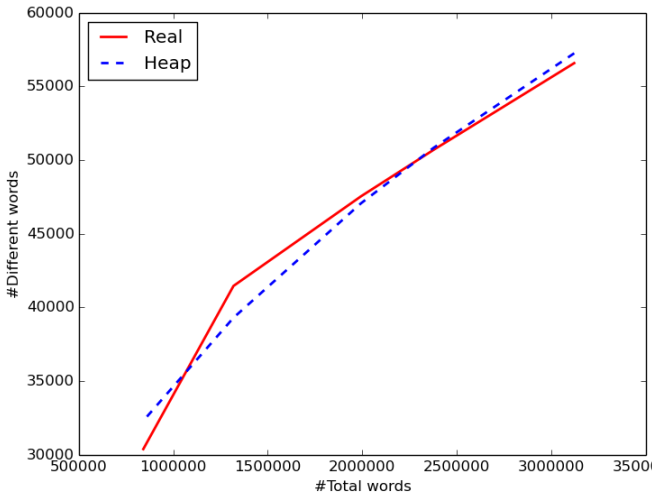

Figure 4: Relació real i relació Heap *fitted*

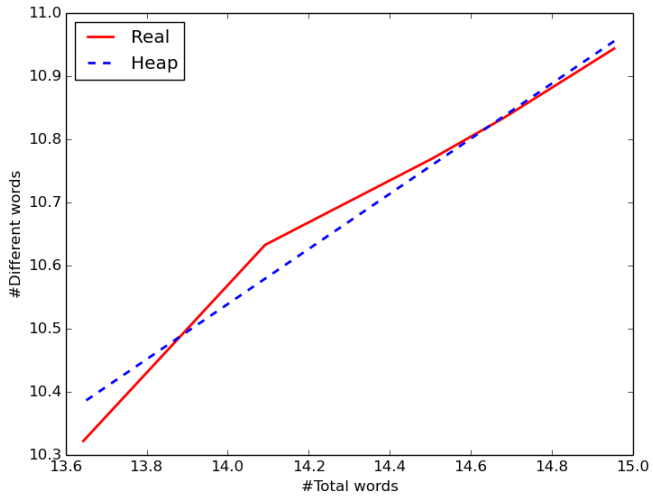

Figure 5: Figure 4 amb escala log-log

Hem de dir que aquesta vegada l'ajustament no és tan ideal, com es pot veure a les figures anteriors. Creiem que això es deu a la variabilitat de les novel · les eliminades (per exemple: és possible que Poe faci servir un lèxic diferent a Dickens) i també al *chunk* de 4MB eliminats a cada nou índex: amb valors més baixos la variància serà menys pronunciada i el *fitting* serà més acurat.