CAIM - Session 1

ElasticSearch and Zipf's and Heaps' laws

Santiago Arxé i Carbona
Bryan Leonardo Salto Salao

6/10/2021

1. Introducción y gestión de los datos

La práctica en cuestión se centra en el estudio de un par de leyes (*Zipf's* y *Heap's*) dedicadas a estudiar ciertas características de las distribuciones de palabras (distintas o no) en grandes textos.

Para realizar ambos estudios, se han llevado a cabo varios procesos comunes que han servido para recolectar los datos, limpiarlos y tratarlos de forma adecuada.

En primer lugar, se ha utilizado *elastic search* para indexar todos los textos que se querían estudiar. Después, utilizando ***CountWords.py***, contar las ocurrencias de cada "palabra". Como solo se necesitan palabras propias y no números o palabras carentes de sentido para este estudio, es conveniente pasar un filtro. Para descartar las palabras que no se consideran oportunas, se ha ejecutado el script ***delete_noise.py***.

Una vez se han obtenido las palabras, se ha pasado a representar los datos reales y a realizar aproximaciones mediante diversas funciones y la función ***scipy.optimize.curve_fit***, que permitía encontrar los parámetros óptimos para cada función. El programa encargado de realizar los *plots* y el ajuste de las funciones es ***plot_elastic_search_laws.py***.

Los principales problemas que se han encontrado al realizar los dos códigos han sido: por una parte distinguir qué palabras se podrían o no eliminar, ya que era extremadamente complicado encontrar unas reglas que "limpiasen" los datos de manera ideal.

2. Zipf's Law

En este primer apartado se quiere demostrar si para un gran número de textos (en este caso novelas y *news*) se cumple la ley de Zipf's. En otras palabras, se quiere comprobar si el *rank-frequency distribution* tiende a seguir una *power-law.* Para ello, hay que ver si las frecuencias siguen ($f = \frac{c}{(rank+b)^{a}}$) para ciertos valores (a, b, c).

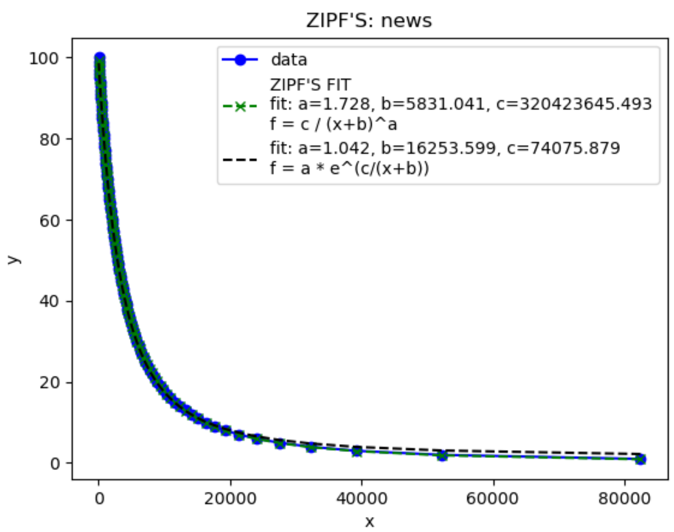

Figura 1: Datos originales (azul), ajuste con, Zipf's (verde) y ajuste con función exponencial (negro) para *news*

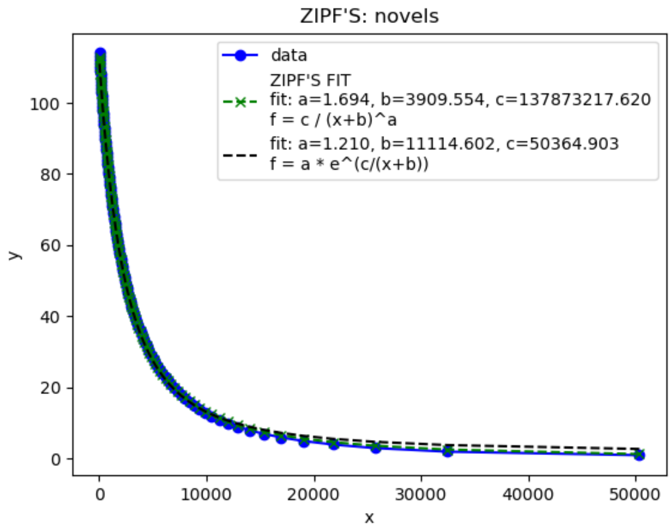

Figura 2: Datos originales (azul), ajuste con, Zipf's (verde) y ajuste con función exponencial (negro) para *novels*

Como se puede observar en las figuras 1 y 2, al final se han obtenido muy buenos resultados. Se puede observar claramente que las función correspondiente Zipf's encajan a la perfección con la *datal* para los valores a = 1.728, b = 5831.041, c = 320423645.493 en el caso de *news*. Para el caso de *novels*, los valores son a = 1.694, b = 3909.554, c = 137873217.620.

Más allá de eso, también se ha intentado comparar con otras funciones, obteniendo resultados muy cercanos a la *data* real aunque ninguno se acerca tanto como Zipf's. Es más, las funciones que eran capaces de aproximarse más a los datos reales, eran siempre funciones exponenciales. Aunque la segunda función de ajuste (negra) representada en las figuras 1 y 2 no tenga exactamente la misma forma que *Zipf's*, lo que está claro es que una función exponencial es la mejor manera de simular la realidad en este caso.

3. Heap's Law

Después de analizar la relación entre la frecuencia de las palabras y su *ranking* en cuanto a dicha frecuencia se refiere, pasó a analizarse otra teórica relación entre palabras y, en este caso, la densidad de palabras diferentes.

Esto es lo que pretende anunciar, precisamente, la *Ley de Heap*. Dicha ley predice que, en un texto con Npalabras totales, podrán encontrarse $k \cdot N^{\beta}$ palabras distintas (para unas ciertas ky β.

Para analizar dicha relación, se procedió a separar las novelas proporcionadas para obtener archivos con diferentes números de palabras y distribuciones. A partir de ahí, se analizó la relación entre palabras totales y palabras distintas a partir de un estudio gráfico y un ajuste de una curva.

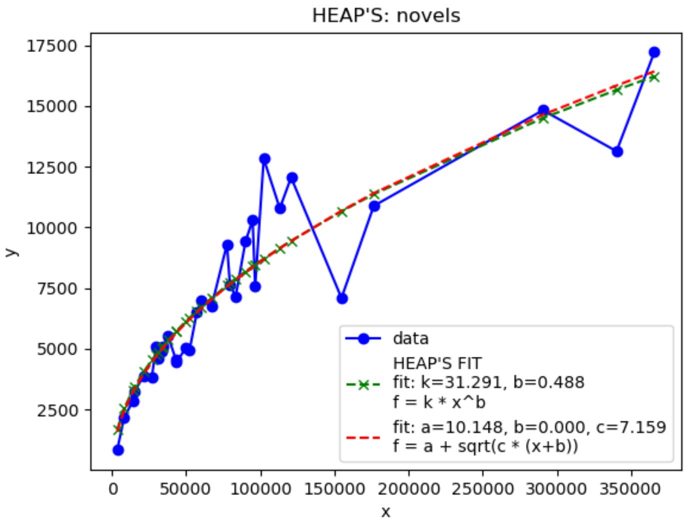

Figura 3: Número de palabras distintas en función de palabras totales (azul) y dos ajustes posibles (verde: *Heap's*, rojo: raíz cuadrada)

Como puede apreciarse en la *Figura 3*, la función de *Heap's* es capaz de ajustarse de manera muy cercana a los datos reales cuando los parámetros son los correctos ($k = 31,29 \mid \beta = 0.488$), sobre todo en los valores más pequeños para N.

No obstante, y pese a que el ajuste sea tan bueno a simple vista, cabe destacar dos cosas:

- Cuanto más crece la N, más variabilidad puede encontrarse en la proporción de las palabras distintas y sus repeticiones. Sin embargo, tampoco puede decirse que se rompa la tendencia.
- Aunque el ajuste sea muy cercano al propuesto en la *Ley de Heap*, y es realmente bueno, no puede asegurarse que esa sea exactamente la tendencia seguida.

En relación al segundo punto, se intentó hacer otro ajuste (visto en rojo en la *Figura 3*) a raíz de lo que se asemejaba la gráfica a la forma de una raíz cuadrada. Visto esto, se ajustaron los parámetros de manera adecuada para una función de raíz cuadrada y pudo obtenerse un ajuste también extremadamente similar.

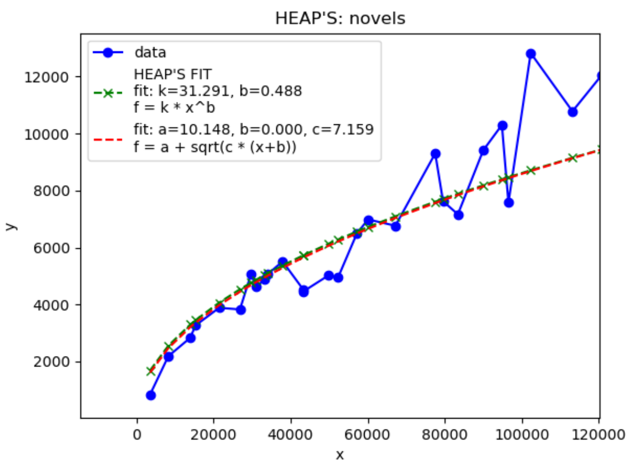

Figura 4: *Zoom* sobre los valores más pequeños de *N*para comprobar la calidad del ajuste para dichos valores.

¿Es eso un error? ¿Cómo puede saberse qué ajuste es mejor? Lo cierto es que… No podrían ser mejores noticias. Y es que la función utilizada en la raíz cuadrada puede simplificarse de la siguiente manera:

$$a + \sqrt{c \cdot (N + b)} = a + (c \cdot (N + b))^{\frac{1}{2}} \rightarrow \{b = 0\} \rightarrow a + (c \cdot N)^{\frac{1}{2}} = 10,48 + 2,67 \cdot N^{0,5}$$

Aunque la fórmula obtenida no sea exactamente la misma, el principal cuerpo de dicha fórmula es $N^{0,5}$, muy similar al $N^{0,488}$obteniendo al ajustar la *Ley de Heap*. Lo que en un principio parecía poder refutar dicha ley, no hace otra cosa que darle aún más fuerza.