CAIM Lab, Session 1: ElasticSearch and Zipf's and Heaps' laws

Adrián Sánchez Albanell

ElasticSearch

El propósito de los primeros apartados de la práctica parece ser trastear un poco con **ElasticSearch**. Para poder trabajar con mi equipo (con sistema operativo Windows 10) he usado **ElasticSearch** y **Kibana**, que ofrece la pagina oficial en versión portable para Windows, sin la necesidad de instalar nada.

En general no ha habido ningún problema para hacer los dos primeros apartados de la sesión ni nada a comentar, **ElasticSearch** nos permite operar con facilidad con una base de datos no relacional, las librerías que ofrece **Python** y se nos proponen en la práctica están bien documentadas y són fáciles de usar y, adicionalmente aunque no fuera necesario para esta sesión, usar *Kibana* permite visualizar los datos que tenemos, ayudando a ver fácilmente que está pasando por detrás mientras trabajamos, lo cual me ha parecido bastante útil para ver si es lo que esperábamos.

Para acabar con está parte de la práctica he indexado los tres conjuntos de ficheros, con los indices *news*, *novels* y *arxiv* respectivamente, y visto un poco el funcionamiento de las *queries* con el código que se nos proporciona y la documentación oficial, donde podemos ver como usar los objetos ***Q*** y ***Search***, no solo para simplificar las busquedas en **ElasticSearch**, también por ejemplo para filtrar u ordenar los resultados obtenidos.

Zipf Law

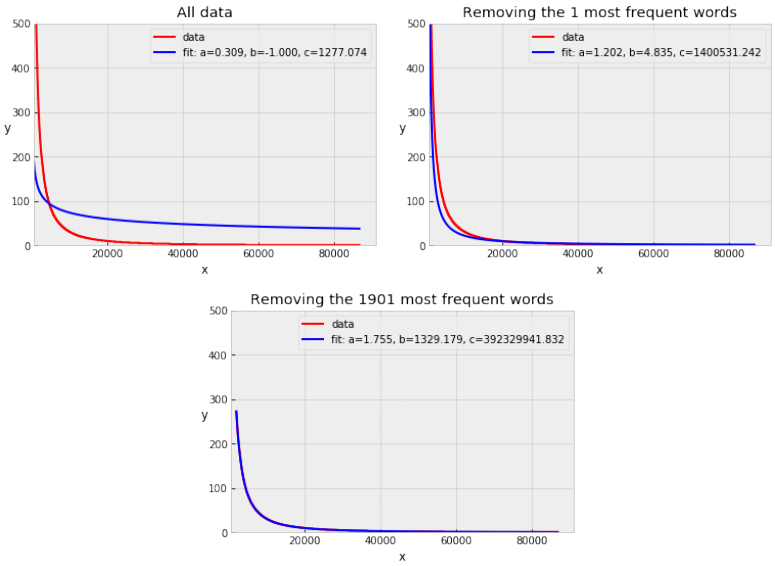
Con los índices necesarios creados como explico en el apartado anterior, he usado el fichero *CountWords.py* para generar un documento por cada uno de ellos. Seguidamente he generado un script tipo notebook con *jupyter*.

Al leer los ficheros resultantes de *CountWords* me encontré con un problema, seguramente derivado de trabajar con Windows, ya que hacia la lectura de los caracteres '\x00' y 'ÿþ' en partes del texto donde no debería, haciendo que las expresiones regulares que estaba usando no funcionaran como esperaba y errores inesperados en el código, principalmente el carácter '\x00', ya que dependiendo de como hagas el output en pantalla se ve exactamente como un espacio en blanco, dándome más de un dolor de cabeza para entender que estaba pasando. Una vez detectado fue fácil arreglarlo eliminando dichos caracteres del texto leído.

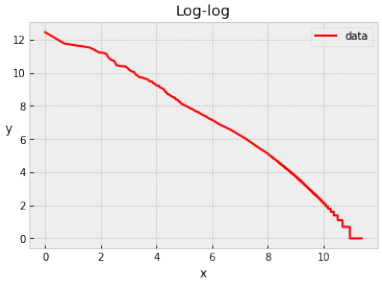
Una vez recogidos los datos que necesitavamos, he usado los recursos recomendados y adicionalmente *numpy*. Con la función *curve_fit* de *scipy*, una vez sacados los datos necesarios, ajustar la función para conseguir los parámetros es casi automático y con *matplotlib* es senzillo representar visualmente los datos iniciales respecto a la función encontrada para ajustarlos.

Como dice en el enunciado que no nos tenemos que fijar en las palabras más frecuentes, he decidido hacer varios experimentos para intentar ajustar cuantos debia eliminar de ellos y como variaba el ajuste de la función respecto a los datos. Primero he representado los datos sin eliminar nada, luego he hecho diferentes iteraciones eliminando al empezar solo un elemento y luego cien más en cada iteración. En cada iteración he calculado el error cuadrático medio de la función con los parámetros encontrados en esa iteración respecto los datos reales. He parado en cuanto el error cuadrático medio no variaba demasiado respecto al encontrado en la iteración anterior.

Como ejemplo muestro aquí los gráficos resultantes de experimentar con los datos del índice *news*. Como se puede apreciar visualmente en los dos primeros gráficos, hay una gran diferencia solo con eliminar el elemento más frecuente respecto a los datos en crudo. Finalmente en la tercera gráfica tenemos la ultima iteración, se ve que los datos cuadran casi a la perfección con los resultados de la powerlaw con los parámetros encontrados. Para una comparación seguramente más objetiva, el *mse* encontrado es 1177982.93, 9864.28 y 0.15 respectivamente. Si observaramos las iteraciones intermedias veriamos como el *mse* varia menos cada vez entre una iteración y la siguiente.



Como dato adicional he pintado los datos en la gráfica *log-log*. El resultado ha sido el esperado. Los resultados con los otros dos índices han sido muy parecidos a los encontrados con *news*, aunque varien un poco se comportan de la misma manera.



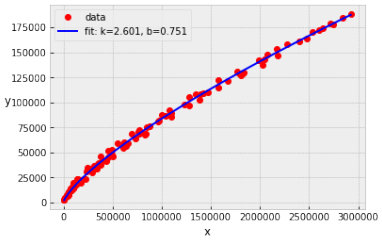
En la gráfica logarítmica se ve que sigue el comportamiento esperado en una power law. Esto es una condición necesaria, pero no suficiente para decir que se trata de una. He encontrado un par de papers muy interesantes sobre el comportamiento de distribuciones power law, uno de ellos justamente sobre por que no es suficiente para decir que se trata de una, y como hacerlo de la forma adecuada. Sin embargo al tratarse de un documento con una formalidad matemática mayor a la que estamos acostumbrados en la carrera y más extenso que la práctica en si, aunque parecia muy interesante y justamente lo necesario para hacer la demostración formal, no me a parecido lo correcto sumergerme ni aprofundizar más en ello.

Heaps Law

Para este apartado he seguido el procedimiento que dice el enunciado.

Primero he creado un fichero **Python**, *IndexFilesHeaps.py* para generar diferentes indices con diferentes cantidades de texto. Para hacerlo he cojido subconjuntos aleatorios de tamaño creciente de novelas de forma iterativa y generado un indice para cada subconjunto. Cuanto mayor el tamaño del subconjunto menos subconjuntos de ese tamaño (si hay 33 novelas y genero un subconjunto de 30 novelas, seguramente no sea muy diferente de otro de 30 novelas, así que solo genero uno de ese tamaño. Si se trata de uno de 5 novelas en cambio es menos probable que coincida con otro subconjunto aleatorio de 5 novelas, así que creare más subconjuntos de este tamaño). La base para generar este script ha sido el codigo de *IndexFiles.py*

Una vez generados los indices, he generado un script en *jupyter*, ayudandome del codigo de *CountWords.py* modificado para recojer los datos que necesitaba, muy parecido al de la parte de **Zipf** pero sin la parte iterativa y de prueba y error, consiguiendo los resultados que se muestran en la siguiente gráfica:



Lo primero que me ha llamado la atención han sido los valores para k y β, ya que en la Wikipedia pone que los valores de k normalmente oscilan entre 10 y 100, y los de β entre 0.4 y 0.6. Aunque siendo Wikipedia y no algun sitio más fiable o los apuntes de clase me llama más la atención que parecerme algo preocupante.

Por otro lado, me ha sorprendido que el error cuadrático medio encontrado ha sido 6239567.67. Sin embargo se ha de tener en cuenta que trabajamos con valores mucho más grandes en este apartado (centenares de miles) y se trata de un valor cuadrático.