Session 1: ElasticSearch and Zipf's and Heaps' laws

Q1 2022-23

Integrantes:
Pol Pérez Castillo
Daniel Ruiz Jiménez

Fecha de entrega: 20-09-2022

Índice

Introducción

En esta práctica vamos a observar el comportamiento de la distribución de las palabras sobre tres documentos de vocabulario orientados a 3 ámbitos diferentes: textos coloquiales, textos literarios y textos científicos. En el enunciado de la práctica ya se nos avanza que los textos literarios probablemente presenten unos mejores resultados y más limpios, pero nosotros estudiaremos los tres casos por separado.

Para este estudio haremos uso de la ley de Zipf, con la cual intentaremos comprobar si la distribución rango-frecuencia de las palabras sigue una ley potencial y qué parámetros cumplen esta ley.

$$f = \frac{c}{(rank + b)^a}$$

Imagen 1: Ley de Zipf

Más tarde con la ley de Heaps, la cual describe el comportamiento de las palabras en un texto, estudiaremos qué relación existe entre el número total de palabras y el número de palabras diferentes en los textos literarios.

$$V_R(n) = Kn^{\beta}$$

Imagen 2: Ley de Heaps

Preparación previa

Antes de estudiar las características de los textos proporcionados, hemos indexado cada uno de los 3 documentos en su correspondiente índice haciendo uso del código dado en *IndexFiles.py*.
Pero hay un problema con el vocabulario encontrado en los textos, y este es que los documentos contienen términos tales como números, url's, fechas y demás palabras ilegibles que nos impiden el análisis correcto de estos textos.
Por lo tanto, hemos modificado parte del código en *CountWords.py* de tal forma que cualquier string que contenga números o carácteres especiales no pase el filtro como palabra válida (no hemos borrado las stopwords). Para este proceso nos hemos ayudado de la función isalpha() de Python. Además también hemos invertido la lista final de palabras de forma que la más frecuente sea la primera en ser escrita en el output. De este modo, el pendiente de la recta que se formará al generar la imagen de la recta log-log que mejor se ajuste será negativo. El fichero pasa a llamarse *CountWordsMod.py*

```
for v in voc:
    if v.isalpha():
        lpal.append((v.encode("utf-8", "ignore"), voc[v]))

results = {}
for pal, cnt in sorted(lpal,
                       key=lambda x: x[0 if args.alpha else 1],
                       reverse=True):
    print(f'{cnt}, {pal.decode("utf-8")}')
```

Imagen 3: Fragmento de código añadido a CountWords.py

Habiendo implementado el código pertinente, ya podemos empezar con el estudio de la ley de Zipf.

Ley de Zipf

Gracias a la función *curve-fit* de la librería Scipy de Python, hemos podido generar unos valores ajustados a, b y c, indicándoles nosotros unos ciertos límites superiores e inferiores. Para el valor *a*, haciendo cálculos previos con valores pequeños vimos que nos daría alrededor de 1 (que corresponde con el valor teórico correcto).

Para los otros dos parámetros hemos estudiado los posibles valores negativos o positivos que pudieran dar valores no esperados o incoherentes. Por esta misma razón hemos decidido por el límite inferior de b = -499, ya que solo cogemos los rangos del 1 al 500 y así intentar evitar que el denominador sea negativo, lo cual nos obliga a poner el límite inferior de c = 0.0 para evitar números de frecuencia negativos.

Por último, el programa implementado en *Zipf.py* lee los outputs ya producidos anteriormente por *CountWords.py* y produce las siguientes tablas para cada uno de los casos:

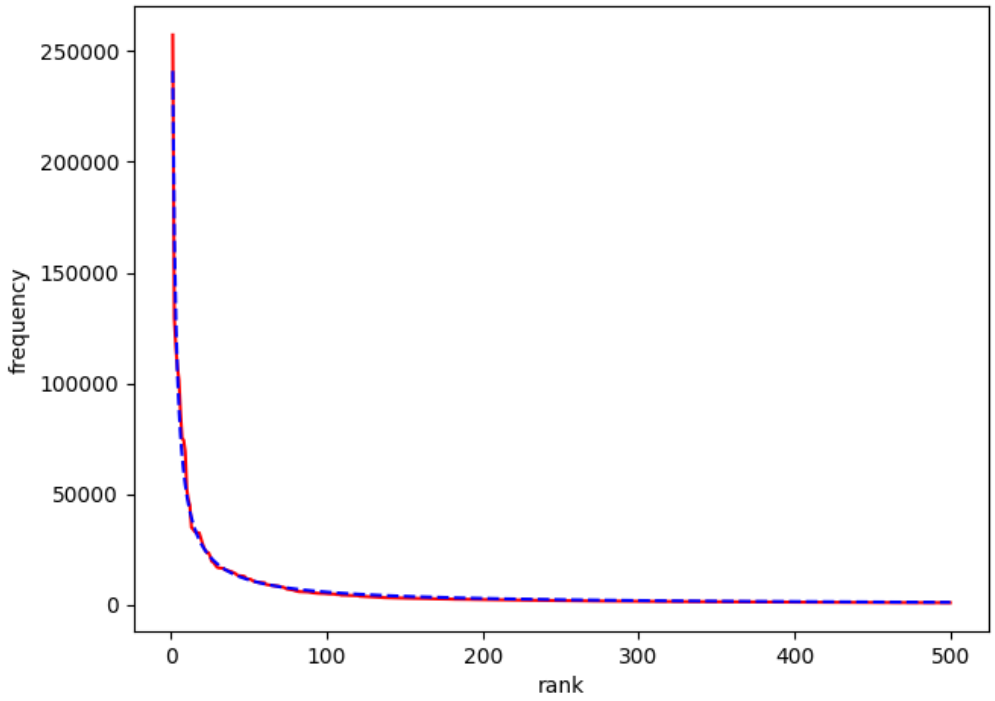

Imagen 4: Gráfica de 20Newsgroups sin escala logarítmica

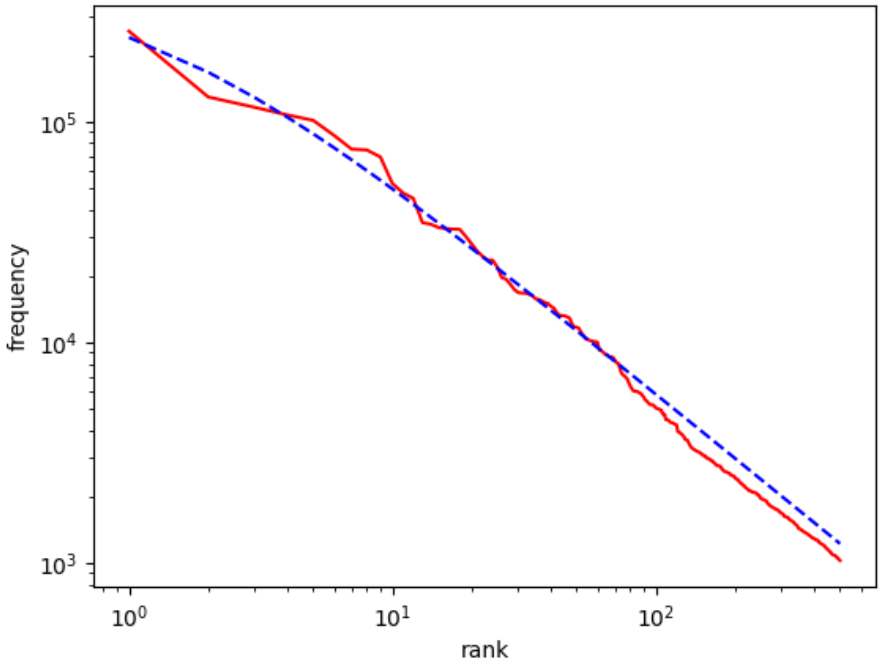

Imagen 5: Gráfica de 20Newsgroups con escala logarítmica

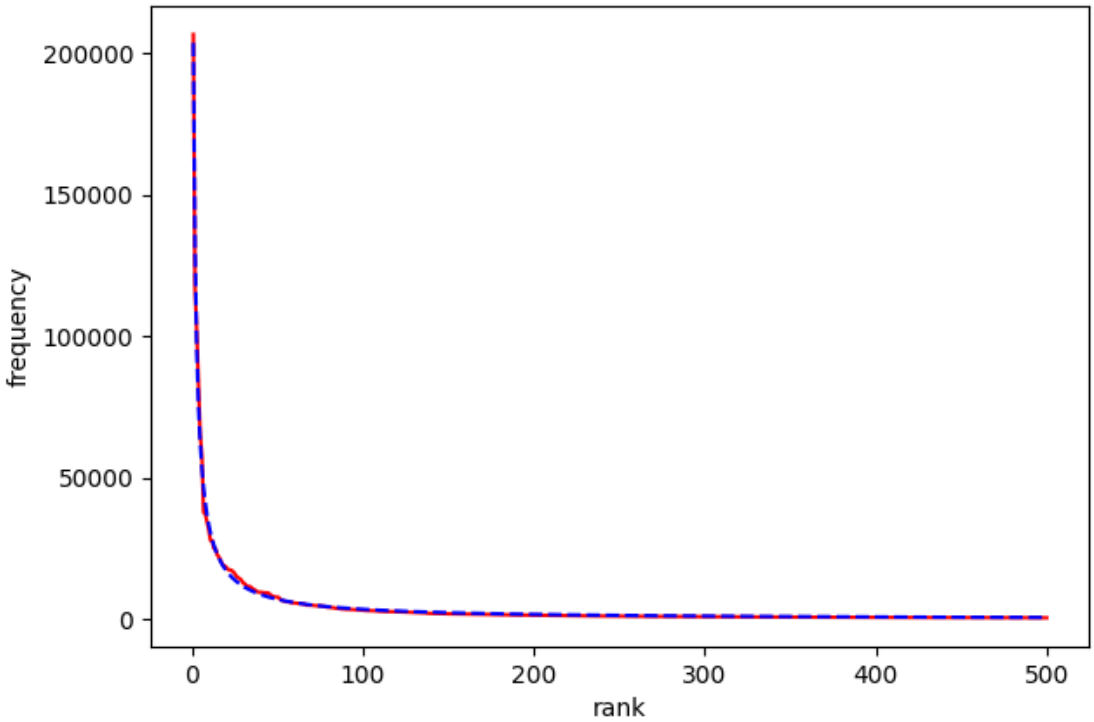

Imagen 6: Gráfica de Novels sin escala logarítmica

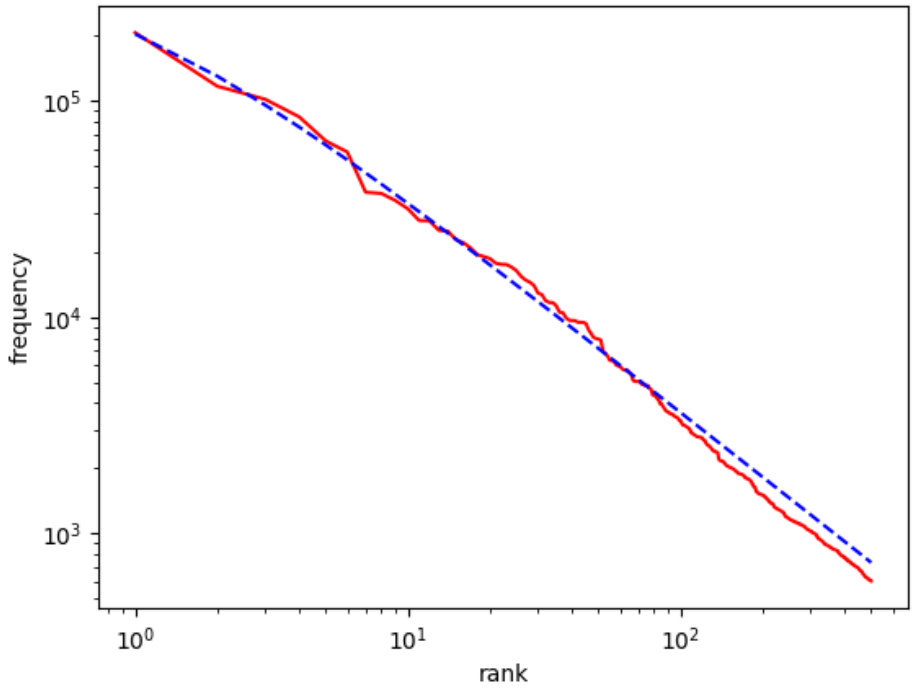

Imagen 7: Gráfica de Novels con escala logarítmica

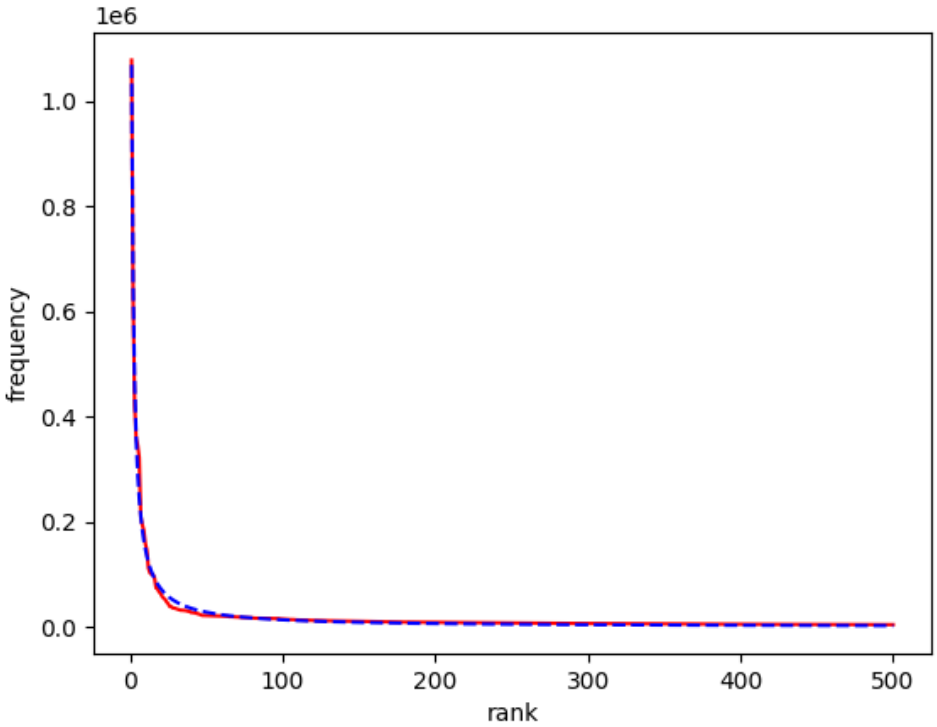

Imagen 8: Gráfica de ArXiv sin escala logarítmica

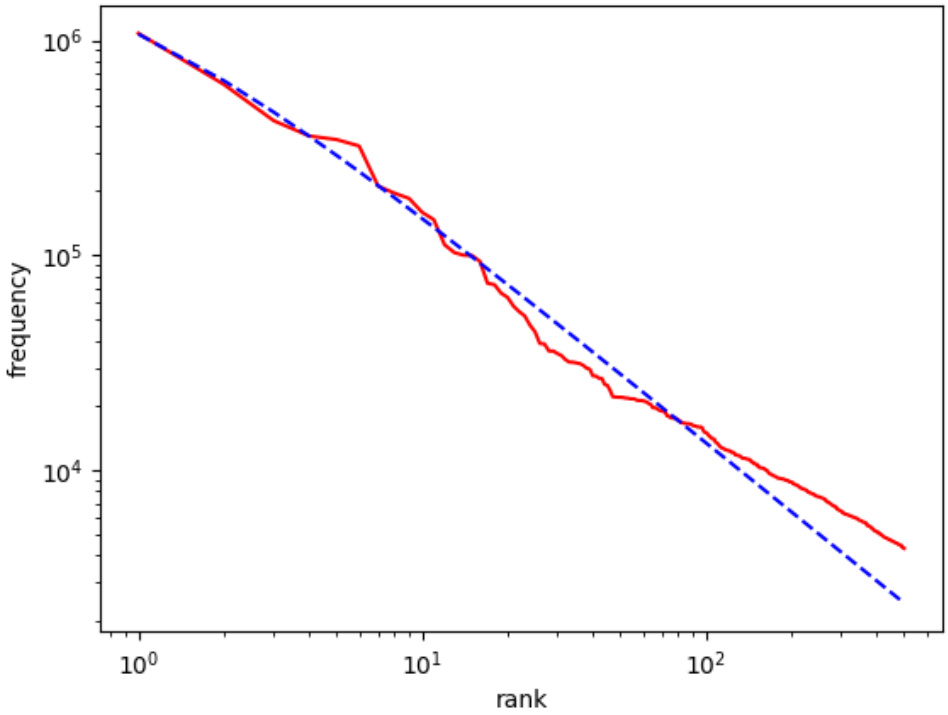

Imagen 9: Gráfica de ArXiv con escala logarítmica

Valores	**20_newsgroups**	**novels**	**ArXiv**
A	0,9741	0,9953	1,0711
B	1,2153	0,7576	0,6901
C	522973	356836	1874525

Las anteriores imágenes y la tabla muestran las gráficas obtenidas ajustadas de manera óptima y el trío de valores a, b y c con los cuales la muestra se ajusta mejor a dicha ley. Para el primer caso hemos usado el conjunto de los textos coloquiales, y vemos que el valor a es muy cercano a 1, y nos indica que se ajusta bastante bien. Para el siguiente conjunto, las novelas, vemos que tenemos el mejor ajuste, la a es casi igual a 1, y para el último conjunto, el de los textos científicos, vemos que la a excede un poco el valor de 1, por lo que tenemos un peor ajuste.

En resumen, las dos gráficas por cada conjunto de textos nos muestran la relación entre nuestros datos y la función que mejor se ajusta a ellos.

Ley de Heaps

Para la ley de Heaps hemos realizado un procedimiento similar que para demostrar la ley anterior. Hemos creado subíndices del conjunto de textos literarios de forma manual, de forma que cada subconjunto de textos tuviera potencias de 2, primero con un texto, luego con 2, luego con 4, etc. Hasta llegar al último subconjunto que contenía 32 textos.
Con el programa *Heap.py* hemos cargado cada subíndice y los hemos concatenado en las listas total (el número total de palabras de cada subíndice) y diff (las ocurrencias de cada palabra), y finalmente hemos llamado a la función *curve_fit* con unos límites para la k y la beta. Estos límites los hemos utilizado para ajustar la ecuación y no tener en cuenta valores irrelevantes.
Hemos declarado que tanto k como beta sean positivos, ya que por la definición de la fórmula no pueden ser negativos, y un límite superior de la beta de 2, ya que los valores no suelen alejarse mucho del 0,5 que es el valor estándar para la lengua inglesa.
Hemos obtenido buenos resultados, ya que los valores se ajustan muy bien a la gráfica. Sin embargo se podrían mejorar más dividiéndolos en más subíndices.

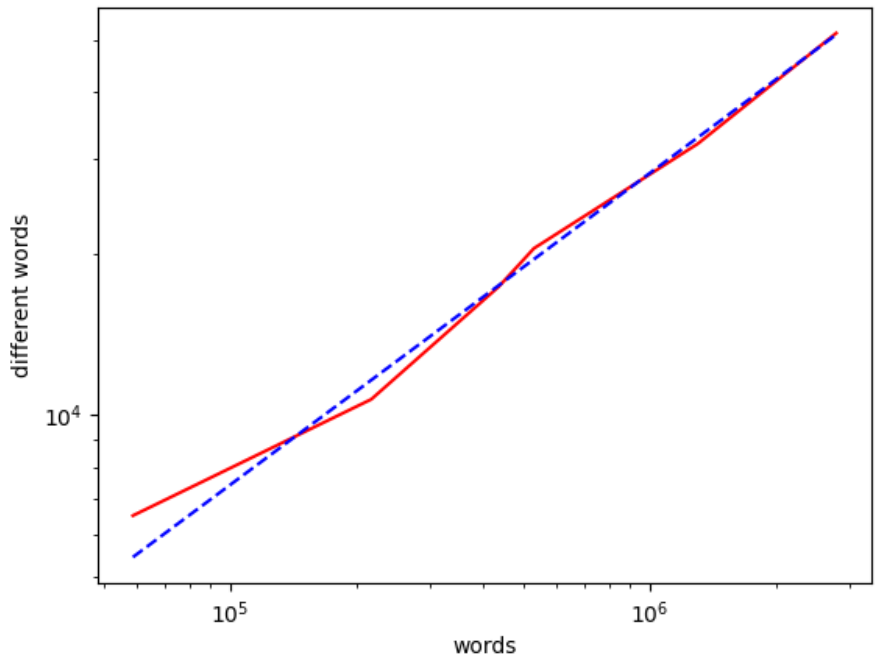

Imagen 10: Gráfica de la ley de Heaps del conjunto de las novelas

Valores	
K	9,3291
β	0,5801

Conclusiones

Para la ley de Zipf, cuanto más se acerque 'a' a 1, más se ajusta la curva. En nuestro caso, el índice de las novelas es el más cercano, debido a su limitado vocabulario. La ley de Zipf se ajusta más a textos del ámbito literario y menos a textos científicos, ya que éstos cuentan con un vocabulario más técnico y utilizan un rango más amplio de palabras, por lo que la frecuencia de uso media de cada palabra es más baja.

Para la ley de Heaps, podemos observar que que la $\beta = 0.5801$. Tal y como sabemos de teoría, el valor de este parámetro puede variar bastante según el idioma de los textos, desde ser aproximadamente 0.8 en la novela de Don Quijote hasta valores inferiores a 0.5 para textos en inglés. En nuestro caso (inglés), al haber tenido en cuenta no solo una novela, sino que 32 novelas diferentes, lo cual implica autores distintos que puedan llegar a usar un vocabulario diferente para expresarse, el valor de 0.5801 nos resulta muy acertado ya que es un poco superior en lo que a la media inglesa corresponde pero sigue ajustándose correctamente a la ley de Heaps.