TP1: Análisis de Texto

Alumno: Gallozo, Luis Angel 156308

1. Escriba un programa que realice analisis lexico sobre la coleccion RI-tknz-data. El programa debe recibir como parametros el directorio donde se encuentran los documentos y un argumento que indica si se deben eliminar las palabras vacas (y en tal caso, el nombre del archivo que las contiene). Dena, ademas, una longitud mnima y maxima para los terminos. Como salida, el programa debe generar:
a) Un archivo (terminos.txt) con la lista de terminos a indexar (ordenado), su frecuencia en la coleccion y su DF (Document Frequency). Formato de salida: < termino > [ESP] < CF > [ESP] < DF >.
Ejemplo:
casa 238 3
perro 644 6
...
zorro 12 1
b) Un segundo archivo (estadisticas.txt) con los siguientes datos (un punto por lnea y separados por espacio cuando sean mas de un valor) :
1) Cantidad de documentos procesados
2) Cantidad de tokens y terminos extrados
3) Promedio de tokens y terminos de los documentos
4) Largo promedio de un termino
5) Cantidad de tokens y terminos del documento mas corto y del mas largo1
6) Cantidad de terminos que aparecen solo 1 vez en la coleccion
c) Un tercer archivo (frecuencias.txt, con un termino y su CF por lnea) con:
1) La lista de los 10 terminos mas frecuentes y su CF (Collection Frequency)
2) La lista de los 10 terminos menos frecuentes y su CF.

2. Tomando como base el programa anterior, escriba un segundo Tokenizer que implemente los criterios del artculo de Grefenstette y Tapanainen para denir que es una \palabra" (o termino) y como tratar numeros y signos de puntuacion. Ademas, extraiga en listas separadas utilizando en cada caso una funcion especca.
a) Abreviaturas tal cual estan escritas (por ejemplo, Dr., Lic., S.A., etc.)2
b) Direcciones de correo electronico y URLs.
c) Numeros (por ejemplo, cantidades, telefonos).
d) Nombres propios (por ejemplo, Villa Carlos Paz, Manuel Belgrano, etc.) y los trate como un unico token.
Genere y almacene la misma informacion que en el caso anterior.

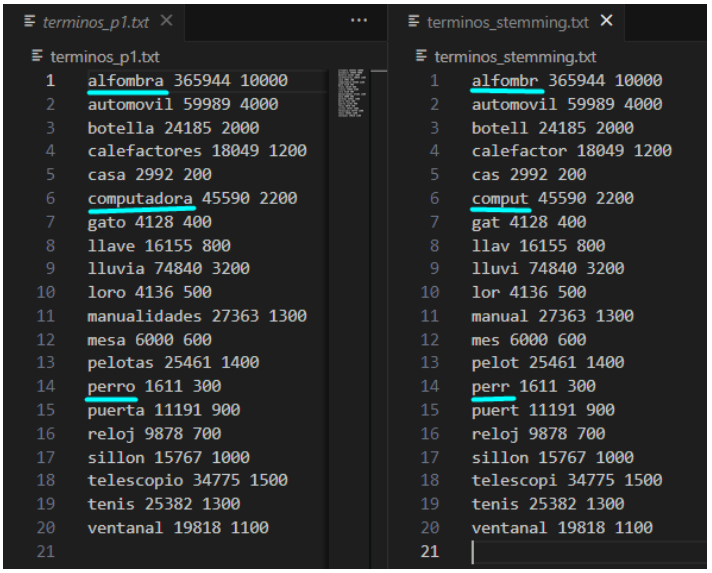
Los puntos 1 y 2 solo son código, las conclusiones de los demás puntos estarán en este documento.

Aclaraciones:

1. Se decidió un mínimo de 2 y máximo de 50 caracteres para los tokens.
2. Al extraer las URL, se decide quitar toda la url de la lista de tokens a procesar.
3. Al extraer Cantidad, se tienen en cuenta todos los números reales enteros y floats (negativos y positivos), también los números de teléfono.
4. Los top 10 no están ordenados por orden alfabético, ya que se van almacenando según son extraídos si son menores o mayores a los almacenados se adhieren a las listas.

3. A partir del programa del ejercicio 1, incluya un proceso de stemming. Luego de modificar su programa, corra nuevamente el proceso del ejercicio 1 y analice los cambios en la colección. ¿Qué implica este resultado?
Busque ejemplos de pares de términos que tienen la misma raÍz pero que el stemmer los trato diferente y términos que son diferentes y se los trato igual.

Como primera prueba se utilizaron los datos de la colección de prueba para corroborar los CF.
Se ve como los términos almacenados sufrieron el cambio a sus raíces morfológicas:



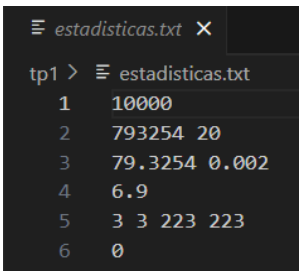
terminos_p1.txt

```
alfombra 365944 10000
automovil 59989 4000
botella 24185 2000
calefactores 18049 1200
casa 2992 200
computadora 45590 2200
gato 4128 400
llave 16155 800
lluvia 74840 3200
loro 4136 500
manualidades 27363 1300
mesa 6000 600
pelotas 25461 1400
perro 1611 300
puerta 11191 900
reloj 9878 700
sillon 15767 1000
telescopio 34775 1500
tenis 25382 1300
ventanal 19818 1100
```

terminos_stemming.txt

```
alfombr 365944 10000
automovil 59989 4000
botell 24185 2000
calefactor 18049 1200
cas 2992 200
comput 45590 2200
gat 4128 400
llav 16155 800
lluvi 74840 3200
lor 4136 500
manual 27363 1300
mes 6000 600
pelot 25461 1400
perr 1611 300
puert 11191 900
reloj 9878 700
sillon 15767 1000
telescopi 34775 1500
tenis 25382 1300
ventanal 19818 1100
```

Aún así, se sigue manteniendo la misma CF de cada uno de los términos encontrados.
Pasa lo mismo con el archivo de estadísticas y frecuencias, los valores se mantienen.

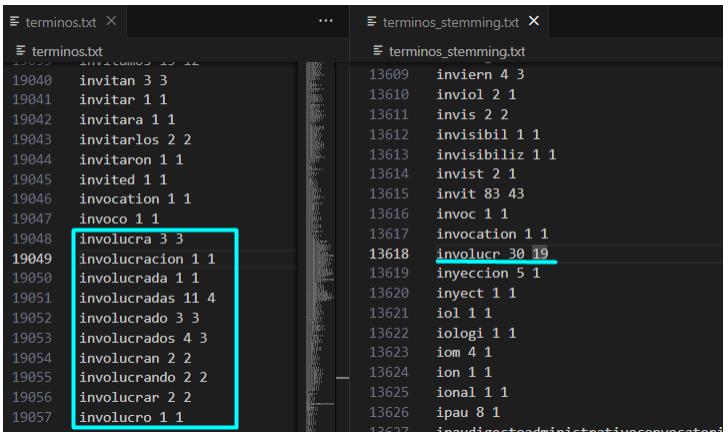


estadisticas.txt

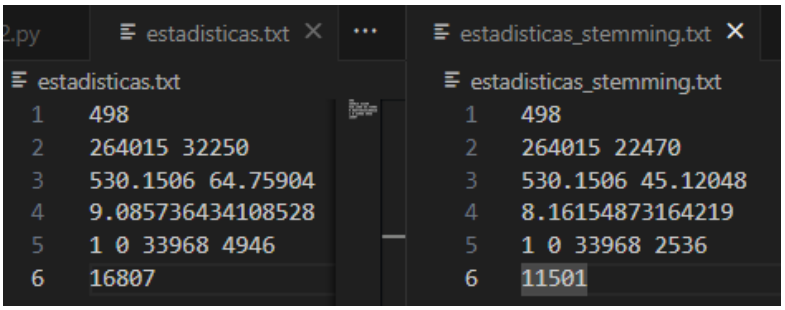
tp1 > estadisticas.txt

```
10000
793254 20
79.3254 0.002
6.9
3 3 223 223
0
```

Como segunda prueba se usó la colección **RI-tknz-data**, donde se obtuvo lo siguiente:



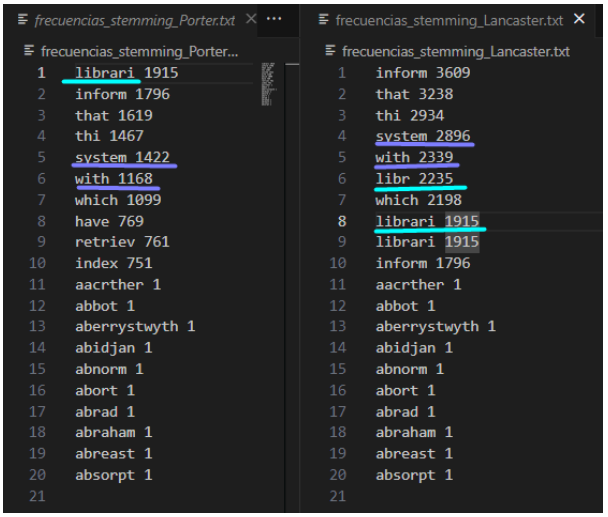
Donde se observa que gracias al algoritmo de stemming se han reducido todas las palabras relacionadas con "involucrar" a la raíz "involucr" haciendo que todas estas variantes de la palabra original se convirtieran en una sola forma base acumulando las frecuencias de cada variante.
En el archivo estadísticas tenemos:



Cambia drásticamente la cantidad de términos almacenados en el índice, se nota más cómo afecta a la cantidad de términos con frecuencia 1 (última línea.

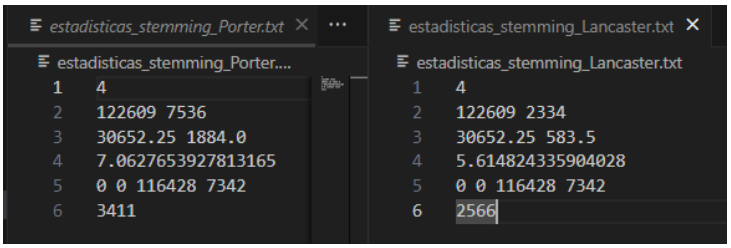
4. Sobre la colección CISI, ejecute los stemmers de Porter y Lancaster provistos en el módulo nltk.stem.
Compare: cantidad de tokens únicos resultantes, resultado 1 a 1 y tiempo de ejecución para toda la colección.
¿Qué conclusiones puede obtener de la ejecución de uno y otro?
En los resultados de los términos más frecuentes(10 primeros) y menos frecuentes (10 últimos), tenemos:

- No tienen los mismos términos en sus rankings.
- Las frecuencias en Lancaster son superiores.



En el apartado de las estadísticas, sufren grandes cambios, como por ejemplo:

- Línea 2: La cantidad de tokens y términos extraídos. **Lancaster extrajo menos términos.**
- Línea 3: Promedio de tokens y términos de los documentos. **Lancaster extrajo menos términos en promedio.**
- Línea 4: El largo promedio de un término se reduce. **Lancaster tuvo menor longitud promedio de términos.**
- Línea 5: Cantidad de tokens y términos del documento más corto y del más largo. **Similar, da 0 el más corto debido a que solo se reconocer token con letras, no tendría sentido ver como realiza stemming de números.**
- Línea 6: La cantidad de términos con frecuencia 1. **Lancaster pudo reducir la cantidad de términos.**



En tiempos de ejecución tenemos que el stemmer de Lancaster tarda más de una tercera parte que el de Porter.

```
PS C:\Users\AngelPC\Desktop\RI\Tps2024\RI_2024\tp1> python .\punto4.py cisi y stopwords.txt
Tiempo Stemming Porter: 4.255003929138184
Tiempo Stemming Lancaster: 13.085001230239868
```

Al observar ambos stemmers se denota como la rigurosidad para reducir cada término a su forma base termina afectando en gran medida el tiempo total de ejecución. Al momento de elegir cual utilizar debemos plantearnos el objetivo del SRI, además de elegir que priorizar, el uso eficiente de recursos o el tiempo de respuesta.

5. Escriba un programa que realice la identificación del lenguaje de un texto a partir de un conjunto de entrenamiento. Pruebe dos métodos sencillos:
a) Uno basado en la distribución de la frecuencia de las letras.
b) El segundo, basado en calcular la probabilidad de que una letra x preceda a una y (calcule una matriz de probabilidades con todas las combinaciones).
Compare los resultados contra el módulo Python langdetect y la solución provista.

Se diseñó un script(“verificar_resultados_lang.py”) que puede comparar las diferencias entre 2 archivos resultado.

Comparación punto a) con el archivo solution:

```
Oracion: [265] Lang_1: French Lang_2: English
Oracion: [271] Lang_1: French Lang_2: English
Oracion: [272] Lang_1: Italian Lang_2: French
Oracion: [281] Lang_1: Italian Lang_2: English
Oracion: [297] Lang_1: French Lang_2: Italian
Cantidad de Diferencias: 38
De un total de 300 casos un 12.67% tuvo clasificacion diferente.
```

Comparación punto b) con el archivo solution:
Como se ve, el punto b) fue más certero en la clasificación debido a un análisis más exhaustivo de pares de caracteres(bigramas).

```
Oracion: [22] Lang_1: Italian Lang_2: French
Oracion: [87] Lang_1: French Lang_2: English
Oracion: [113] Lang_1: French Lang_2: English
Oracion: [126] Lang_1: Italian Lang_2: French
Oracion: [201] Lang_1: Italian Lang_2: English
Cantidad de Diferencias: 5
De un total de 300 casos un 1.67% tuvo clasificacion diferente.
```

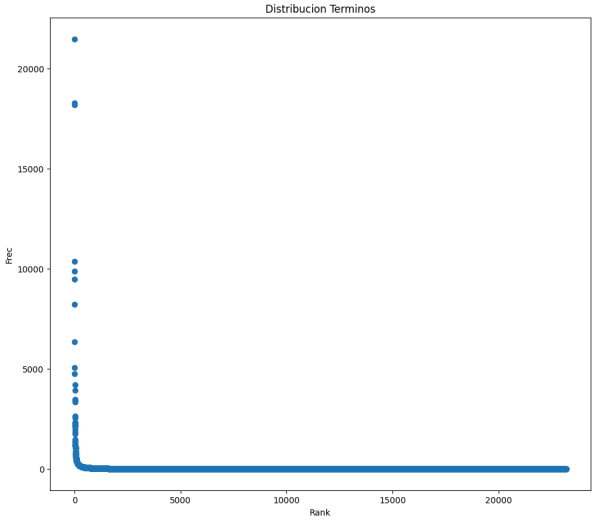
Comparación entre langdetect y archivo solution :

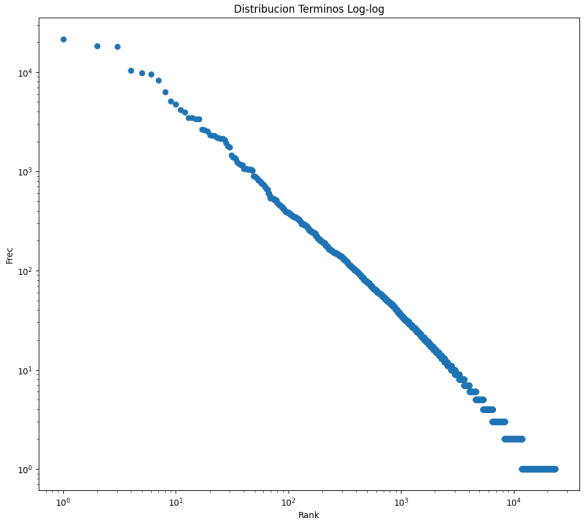
```
Oracion: [28] Lang_1: Deutsch Lang_2: English
Oracion: [87] Lang_1: Indonesian Lang_2: English
Oracion: [126] Lang_1: Italian Lang_2: French
Oracion: [201] Lang_1: Somali Lang_2: English
Cantidad de Diferencias: 4
De un total de 300 casos un 1.33% tuvo clasificacion diferente.
```

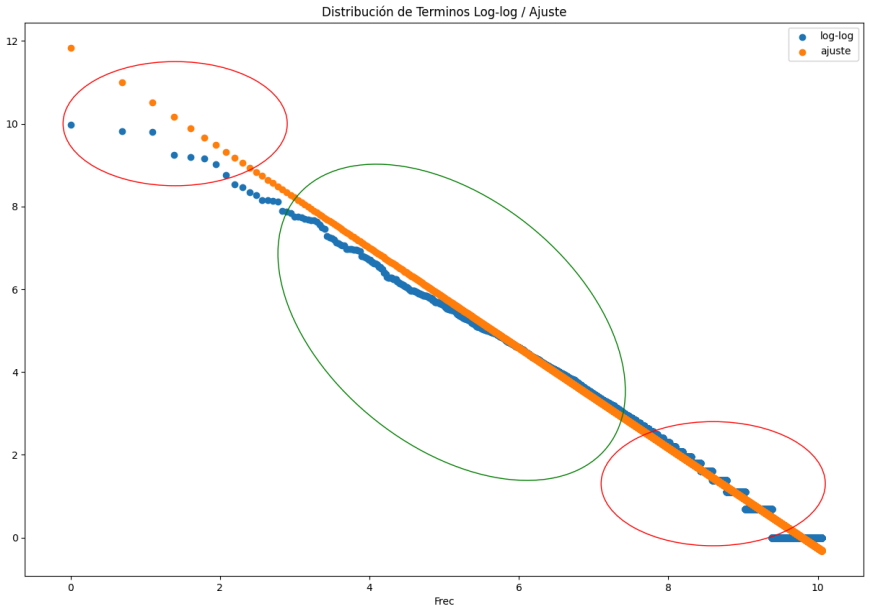
	Tipo Método		
Solution	Unigrama	Bigrama	Lang-Detect
Cant. diff.	38	5	4
% diff.	12.67%	1.67%	1.33%

Como se observa en las comparaciones la mejora del punto b al utilizar la probabilidad de que una letra preceda a otra brinda más información a favor de la predicción. Aunque no termina siendo tan bueno como el módulo langdetect, que además de ser levemente mejor sorprende al identificar nuevos idiomas, obviamente el módulo contiene mucha más robustez y mejor lógica para realizar las comparaciones.

6. En este ejercicio se propone verificar la predicción de ley de Zipf. Para ello, descargue desde Project Gutenberg el texto del Quijote de Cervantes y escriba un programa que extraiga los términos y calcule sus frecuencias (el programa debe generar la lista ordenada por frecuencia descendente). Calcule la curva de ajuste utilizando la función Polyfit del modulo NymPy9. Con los datos crudos y los estimados grafique en la notebook ambas distribuciones (haga 2 graficos, uno en escala lineal y otro en log-log). ¿Cómo se comporta la predicción? ¿Qué conclusiones puede obtener?







Como se puede ver en ambos gráficos existe correlación negativa entre la frecuencia y los términos(aproximadamente de -0.0919, valor obtenido en la notebook adjunta), lo cual quiere decir que mientras más aumenten los términos aumenta la aparición de términos poco frecuentes.

Por lo que se identifica que en el archivo se sigue la ley de Zypf. La distribución de frecuencia sigue esta relación inversa entre la frecuencia y el ranking, en la que los términos más comunes tienen un ranking alto(están en los primeros lugares) y las menos comunes tienen un ranking bajo(últimos lugares).
Esto indicaría que los términos más frecuentes en el archivo son las palabras vacías(stopwords), como se ve en la siguiente tabla:

	word	frec	rank
0	que	21475	1
1	de	18301	2
2	y	18188	3
3	la	10363	4
4	a	9880	5
5	el	9488	6
6	en	8241	7
7	no	6345	8
8	se	5078	9
9	los	4748	10

Una ayuda a esto sería determinar los términos más significativos y ponderarlos más que las stopwords.

7.Usando los datos del ejercicio anterior y de acuerdo a la ley de Zipf, calcule la cantidad de palabras que debería haber en el 10%, 20% y 30% del vocabulario. Verifique respecto de los valores reales.

	word	rank	frec	frec_aprox	frec_cum	frec_aprox_cum	frec_rel	frec_rel_cum	frec_aprox_rel_cum
0	que	1	21475	138187.450913	21475	138187.450913	0.055889	0.055889	0.208712
1	de	2	18301	59775.105772	39776	197962.556685	0.047629	0.103518	0.298993
2	y	3	18188	36612.069106	57964	234574.625790	0.047335	0.150852	0.354291
3	la	4	10363	25856.640718	68327	260431.266509	0.026970	0.177822	0.393343
4	a	5	9880	19742.717598	78207	280173.984106	0.025713	0.203535	0.423162
5	el	6	9488	15837.113203	87695	296011.097309	0.024693	0.228228	0.447081
6	en	7	8241	13144.276552	95936	309155.373861	0.021447	0.249675	0.466934
7	no	8	6345	11184.687348	102281	320340.061209	0.016513	0.266188	0.483827
8	se	9	5078	9700.183304	107359	330040.244513	0.013216	0.279404	0.498477
9	los	10	4748	8540.015934	112107	338580.260446	0.012357	0.291761	0.511376
10	con	11	4202	7610.523193	116309	346190.783640	0.010936	0.302696	0.522870

Como se ve en la tabla, existe cierta diferencia entre las frecuencias aproximadas por la ley de zipf y las reales, esto termina afectando a la ubicación de 10, 20 y 30% de la cantidad total de palabras. Más en detalle, para las frecuencias reales el 10% tiene 39776 palabras mientras la aproximada por Zipf se encuentra sólo en la primer palabra con 138187 lo cual es demasiado (obviamente puede fallar la aproximación debido a que se utilizan parámetros aproximados para la ecuación de Zipf). Vemos lo mismo para el 20% y 30% que no salen del 1° y 2° lugar del ranking según Zipf, mientras que las frecuencias reales están entre los primeros 5 lugares(con 78207 palabras) y los primeros 10 lugares(con 116309 palabras).

Utilice esta aproximación para podar el vocabulario en los mismos porcentajes e indique qué porcentaje de la poda coincide con palabras vacías11.

Resultados:

	word	rank	frec	frec_aprox	frec_cum	frec_aprox_cum	frec_rel	frec_rel_cum	frec_aprox_rel_cum
0	don	1	2652	69730.255084	2652	69730.255084	0.013940	0.013940	0.166182
1	si	2	2312	31680.776967	4964	101411.032051	0.012153	0.026093	0.241685
2	mas	3	2284	19969.757716	7248	121380.789767	0.012006	0.038099	0.289277
3	quijote	4	2180	14393.631975	9428	135774.421742	0.011459	0.049559	0.323580
4	sancho	5	2148	11165.280608	11576	146939.702350	0.011291	0.060850	0.350189
5	dijo	6	1808	9072.925942	13384	156012.628292	0.009504	0.070354	0.371812
6	tan	7	1243	7612.898981	14627	163625.527273	0.006534	0.076887	0.389955
7	asi	8	1065	6539.506328	15692	170165.033601	0.005598	0.082486	0.405541
8	senor	9	1063	5719.055850	16755	175884.089450	0.005588	0.088073	0.419170
9	respondio	10	1063	5072.758794	17818	180956.848245	0.005588	0.093661	0.431260
10	ser	11	1056	4551.263797	18874	185508.112042	0.005551	0.099212	0.442106

```
Palabras Unicas - Porcentaje de la poda coincide con palabras vacías: 0.88%
Frecuencia totales en Datos Reales - Porcentaje de la poda coincide con palabras vacías: 50.49%
Frecuencia totales en Datos Aproximdaos - Porcentaje de la poda coincide con palabras vacías: 29.3%
```

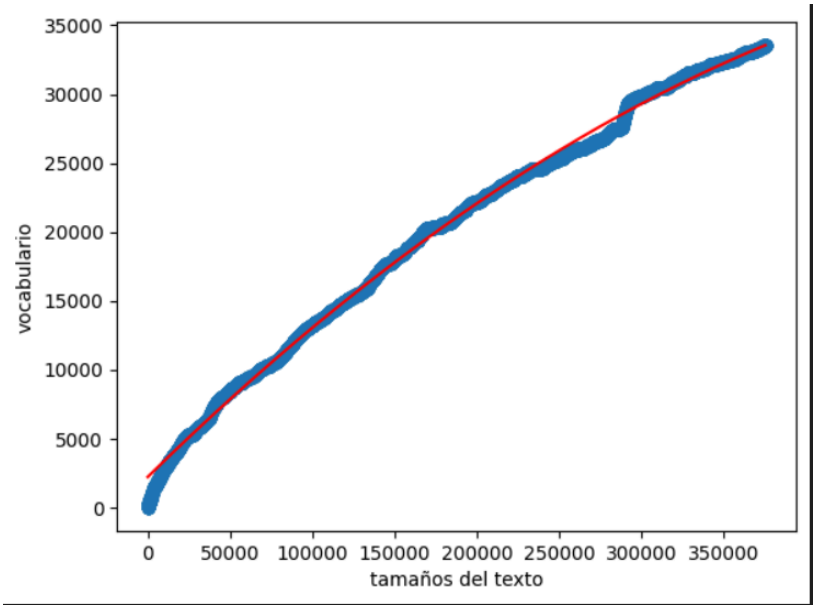
Como se observa al eliminar las stopwords, vemos un decremento las frecuencias, el porcentaje bajó drásticamente ya que estas contenían una mayor cantidad de aparición en el texto. Tal y como se sospechaba por la ley de Zipf vemos que en cantidad (cuantas palabras representaban stopwords) no llega al 1% de las palabras, mientras que en lo que respecta a frecuencias estas representaban un 50.49% de las frecuencias totales de las palabras reales y un 29.3% de las frecuencias totales de las palabras aproximadas por Zipf.

Extraiga las palabras podadas que no son stopwords y verifique si, a su criterio, pueden ser importantes para la recuperación.

Luego de la revisión de las stopwords eliminadas, a mi criterio ninguna es importante debido a que no representan entidades o sustantivos importantes, la mayoría son conectores o verbos conjugados. Cabe aclarar que si el SRI tiene como requisito las búsquedas por frases, algunos verbos que aparecen en citas muy reconocidas deberían ser tomados en cuenta como términos y no stopwords.

8. Codifique un script que reciba como parámetro el nombre de un archivo de texto, tokenize y calcule y escriba a un archivo los pares (#términos totales procesados, #términos únicos). Verifique en qué medida satisface la ley de Heaps. Grafique en la notebook los ajustes variando los parámetros de la expresión. Puede inicialmente probar con los archivos de los puntos anteriores.

Se realizó un script(“punto8.py”) que analiza todo un directorio y por cada archivo procesado escribe un registro de lo pedido. Se procesaron los archivos txt de “RI-tknz-data”, dando los siguientes resultados:



Como se aprecia en el gráfico se cumple la ley de Heaps, ya que a medida que se iban procesando documentos en el script, se descubrieron nuevos términos únicos (aumentando el vocabulario). También se ve que cada que crece la cantidad de documentos procesados, la adición de términos únicos en el vocabulario serán menos, ya que hay una alta probabilidad de que los documentos compartan términos entre sí.