



Análisis de Texto

Fecha entrega: 09/04/2019

Bibliografía sugerida: MIR [1] Capítulo 7, TOL [5] Capítulo 3, MAN [4] Capítulo 6, Grefenstette et al. [2], Ha et al. [3].

1. Escriba un programa que realice análisis léxico sobre la colección T12012-gr. El programa debe recibir como parámetros el directorio donde se encuentran los documentos y un argumento que indica si se deben eliminar las palabras vacías (y en tal caso, el nombre del archivo que las contiene). Defina, además, una longitud mínima y máxima para los términos. Como salida, el programa debe generar:
 - a) Un archivo (`terminos.txt`) con la lista de términos a indexar (ordenado), su frecuencia en la colección y su DF (*Document Frequency*).
 - b) Un segundo archivo (`estadisticas.txt`) con los siguientes datos:
 - 1) Cantidad de documentos procesados
 - 2) Cantidad de tokens y términos extraídos
 - 3) Promedio de tokens y términos de los documento
 - 4) Largo promedio de un término
 - 5) Cantidad de tokens y términos del documento más corto y del más largo¹
 - 6) Cantidad de términos que aparecen sólo 1 vez en la colección
 - c) Un tercer archivo (`frecuencias.txt`) con:
 - 1) La lista de los 10 términos más frecuentes y su CF (*Collection Frequency*)
 - 2) La lista de los 10 términos menos frecuentes y su CF.

El programa debe utilizar llamadas a las siguientes funciones:

- a) `lista_tokens = tokenizar(string)`²
- b) `lista_terminos = sacar_palabras_vacias(lista_tokens, lista_vacias)`³

2. Tomando como base el programa anterior, escriba un segundo *Tokenizer* que implemente los criterios del artículo de Grefenstette y Tapanainen para definir qué es una “palabra” (o término) y cómo tratar números y signos de puntuación. Además, extraiga en listas separadas utilizando en cada caso una función específica⁴.
 - a) Abreviaturas tal cual están escritas (por ejemplo, Dr., Lic., S.A., NASA, etc.)
 - b) Direcciones de correo electrónico y URLs
 - c) Números (por ejemplo, cantidades, teléfonos)
 - d) Nombres propios (por ejemplo, Villa Carlos Paz, Manuel Belgrano, etc.) y los trate como un único token.

Genere y almacene la misma información que en caso anterior.

¹Medir la longitud del documento en cantidad de tokens.

²Realiza la normalización del texto contenido en la variable (`string`) y la divide en tokens que inserta en la lista que retorna.

³Quita de la primera lista los duplicados y las palabras que se encuentran en la segunda.

⁴Por ejemplo, para el primer caso defina la función `get_abreviaturas(string)` que devuelve una lista.



3. Repita el procesamiento del ejercicio 1 utilizando la colección T12012-qm. Verifique los resultados e indique las reglas que debería modificar para que el *tokenizer* responda al dominio del problema.
4. A partir del programa del ejercicio 1, incluya un proceso de *stemming*⁵. 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 trató diferente y términos que son diferentes y se los trató igual.
5. Sobre la colección CISI⁶, ejecute los *stemmers* de Porter y Lancaster provistos en el módulo `nltk.stem`. Compare los resultados (defina las métricas para la comparación). Qué conclusiones puede obtener de la ejecución de uno y otro?
6. Escriba un programa que realice la identificación del lenguaje de un texto a partir de un conjunto de entrenamiento⁷. 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*⁸ y la solución provista.

Propiedades del Texto⁹.

7. En este ejercicio se propone verificar la predicción de ley de Zipf. Para ello, descargue desde Project Gutenberg el texto del Quijote de Cervantes¹⁰ y escriba un programa que extraiga los términos y calcule sus frecuencias. Calcule la curva de ajuste utilizando la función *Polyfit* del módulo Numpy¹¹. Con los datos crudos y los estimados grafique en la notebook ambas distribuciones (haga 2 gráficos, uno en escala lineal y otro en log-log). ¿Cómo se comporta la predicción? ¿Qué conclusiones puede obtener? Verifique que el ajuste obtenido se comporta mejor para un rango de valores que excluye los primeros y los últimos datos.
8. Usando los datos del ejercicios anterior y de acuerdo a la ley de Zipf, calcule la proporción del total de términos para aquellos que tienen frecuencia $f = \{100, 1000, 10000\}$. Verifique respecto de los valores reales. ¿Qué conclusión puede obtener?
9. Para el texto del ejercicio 5 procese cada palabra en orden y calcule 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.

⁵Use la librería NLTK (Natural Language Toolkit). Revise qué algoritmos soporta para español e inglés.

⁶http://ir.dcs.gla.ac.uk/resources/test_collections/cisi/cisi.tar.gz

⁷Utilice los datos de entrenamiento y de evaluación provistos en: <http://www.labredes.unlu.edu.ar/sites/www.labredes.unlu.edu.ar/files/site/data/ri/languageIdentificationData.tar.gz>

⁸`sudo pip install langdetect`

⁹Los siguientes ejercicios se deben realizar en una *notebook* IPython para trabajar en un entorno interactivo que provee, además, capacidades gráficas

¹⁰<http://www.gutenberg.org/cache/epub/2000/pg2000.txt> (en UTF-8)

¹¹<https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.polyfit.html>



Referencias

- [1] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval: The Concepts and Technology Behind Search*. Addison-Wesley Publishing Company, USA, 2nd edition, 2008.
- [2] Gregory Grefenstette and Pasi Tapanainen. What is a word, what is a sentence? problems of tokenization. In *Rank Xerox Research Centre*, pages 79–87, 1994.
- [3] Le Quan Ha, Darryl Stewart, Philip Hanna, and F Smith. Zipf and type-token rules for the english, spanish, irish and latin languages. *Web Journal of Formal, Computational and Cognitive Linguistics*, 1 (8):1–12, 1 2006.
- [4] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.
- [5] Gabriel Tolosa and Fernando Bordignon. *Introducción a la Recuperación de Información. Conceptos, modelos y algoritmos básicos*. Laboratorio de Redes de Datos. UNLu, Argentina, 1st edition, 2005.