**Laboratorio 1: Recuperación de la Información**

Agis, Jaime., Reyes, Alfonso., y Torres, Nancy.

Benemérita Universidad Autónoma de Puebla

Recuperación de la información

Fecha: 23 de agosto de 2022

**Resumen:** En esta práctica se instaló Python con la librería NLTK el cual es una herramienta para el procesamiento del lenguaje natural, con el fin de aprender a preparar un texto, en este caso se usó el e-book: “Around the World in Eighty Days, by Jules Verne”. Para llevar a cabo el trabajo de alistar el texto, se leyó el archivo y después se separó el texto por palabras, se eliminaron los signos de puntuación usando “re”, se cambió las mayúsculas a minúsculas y finalmente se eliminaron las palabras vacías. Esta práctica aportó un entendimiento más claro en el uso de Python con NLTK para preparar textos, ya que se pudo analizar con cada paso como cambiaba el texto.

**Palabras clave:** Tokens, Python, NLTK, texto, palabras vacías.

**I. INTRODUCCIÓN**

En esta práctica usamos el texto “Around the World in Eighty Days, by Jules Verne” para comprender y analizar el uso de Python con NLTK para preparar un texto, llevando a cabo la separación de tokens, la eliminación de puntuación, las mayúsculas y las palabras vacías. Es importante limpiar el texto (preprocesamiento) que puede contener palabras no deseadas o sin importancia, si no se hace este proceso, puede que tengamos resultados que no brinden una precisión eficiente y dificulten su comprensión y análisis.

El trabajo está conformado por un resumen, la introducción, el objetivo del problema, el desarrollo del problema, los resultados, las conclusiones y la bibliografía.

**II. OBJETIVO Y PLANTEAMIENTO DEL PROBLEMA**

Aprender a preparar los textos para que sean de utilidad en el proceso de recuperación de información. Para ello deberás separar el texto en tokens, eliminar los tokens inútiles (signos de puntuación, números, palabras vacías y convertir a minúsculas.

**III. DESARROLLO EXPERIMENTAL**

**Materiales:**

Instalar Python y NLTK

**Descripción:**

*1. Descarga el e-book en texto plano Around the World in Eighty Days, by Jules Verne*

*https://www.gutenberg.org/files/103/103-0.txt*

*2. Lee el archivo que descargaste*

filename = "julioverne.txt"

file = open(filename, 'rt',encoding='utf-8-sig')

text = file.read()

file.close()

Se tuvo que añadir en la función open()el encoding del documento para que pudiera ser leído correctamente.

*3. Ahora, para separar las 100 primeras palabras*



*¿Qué observas en estas palabras?*

Vemos en la ejecución de esta sección de código que las palabras están en todo tipo de formatos, con mayúsculas y minúsculas mezcladas, se toman en cuenta signos de puntuación que amplían el vocabulario de la colección.

*4. Utiliza* ***re*** *de python para eliminar signos de puntuación. Lee en la web cómo funciona re en python, escribe un breve resumen de lo que entendiste. Ahora prueba las siguientes líneas, asegúrate de entenderlas y hacer cambios si es necesario.*

La librería re de python permite hacer interpretaciones del texto con base en expresiones regulares **regex,** que son patrones compilados para filtrar datos de forma eficiente de un texto.

En el caso de la práctica, se hace uso de la función re.split(), para dividir el texto con palabras buscando obviar los signos de puntuación.

Aquí usamos el regex: **r**'\W+':

**r.** prefijo que permite usar secuencias de escape (\) como secuencias en ‘crudo’. Si no se incluye, el carácter de escape será tomado como una secuencia inválida.

\W busca hacer match a cualquier no-palabra carácter (!.?¡!|[]{}\*´^~, etc)

+ hace match al token anterior un número ilimitado de veces

En este caso, cualquier carácter no-palabra consecutivo.

Entonces, esta expresión busca hacer match y separar el texto en base a los caracteres no-palabra encontrados en el texto.



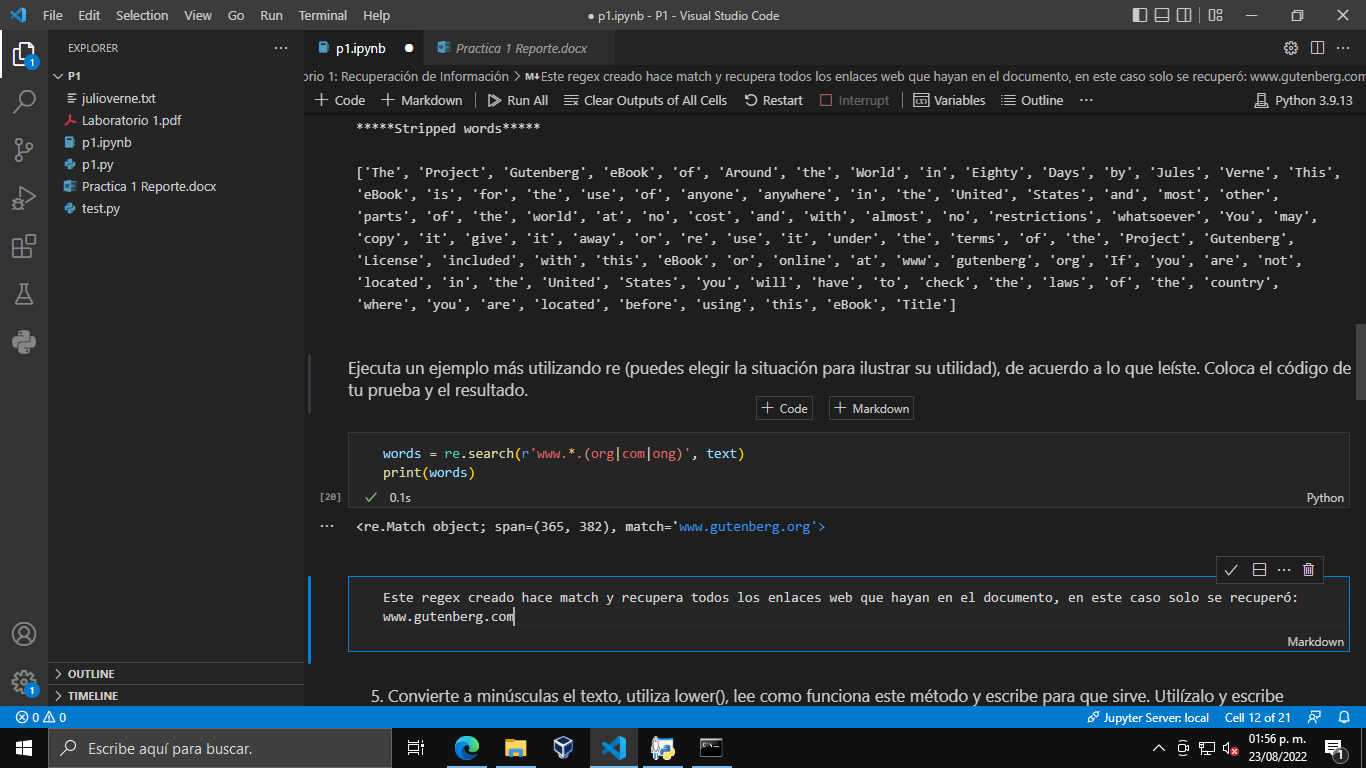
*Busca una mejor manera de eliminar los signos de puntuación. ¿Qué observas? ¿Hay cambios?*

**

*Escribe los cambios que ves en la separación de las palabras, asegúrate de entender el código.*

Al separar los datos de la nueva forma usando string.punctuation y re, a primera vista no pareciera haber cambios significativos en los resultados, pero por la forma en que se compila la expresión regular, esta toma en cuenta más casos que son los que contiene la librería de python string.

*Ejecuta un ejemplo más utilizando re (puedes elegir la situación para ilustrar su utilidad), de acuerdo a lo que leíste. Coloca el código de tu prueba y el resultado.*



Este regex creado hace match y recupera todos los enlaces web que hayan en el documento, en este caso solo se recuperó: www.gutenberg.com

*5. Convierte a minúsculas el texto, utiliza lower(), lee cómo funciona este método y escribe para qué sirve. Utilízalo y escribe nuevamente las 100 primeras palabras.*

*Escribe tus observaciones.*

Método string.lower() -> convertir a minúsculas una cadena

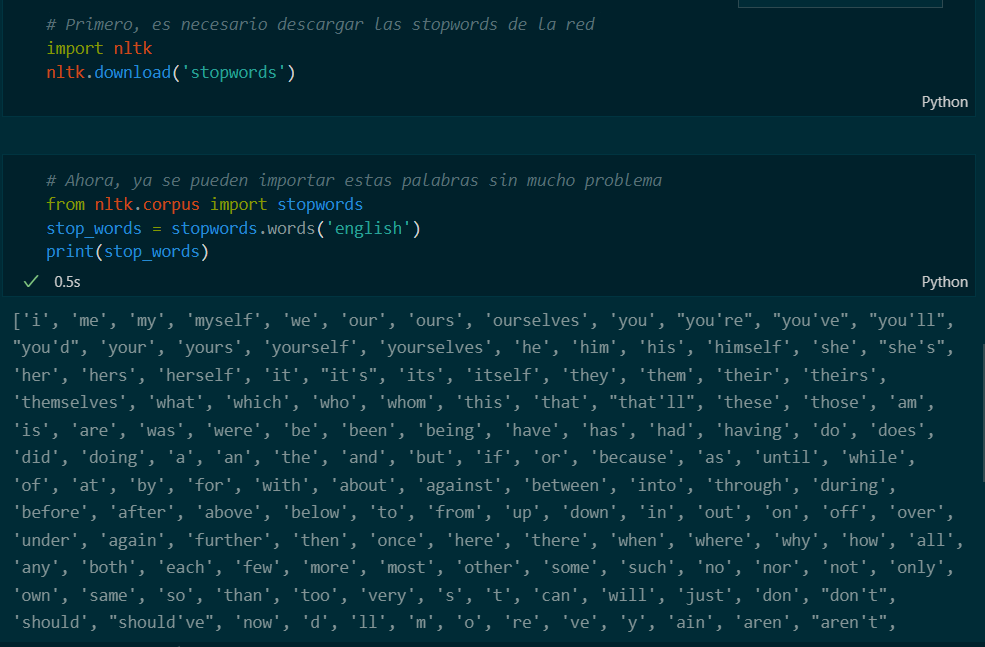
Se le aplica a cadenas de texto (se tiene que aplicar antes de que se separen las palabras).



Al pasar las palabras a minúsculas, la extensión del vocabulario disminuye, haciendo más fácil el procesado de la colección.

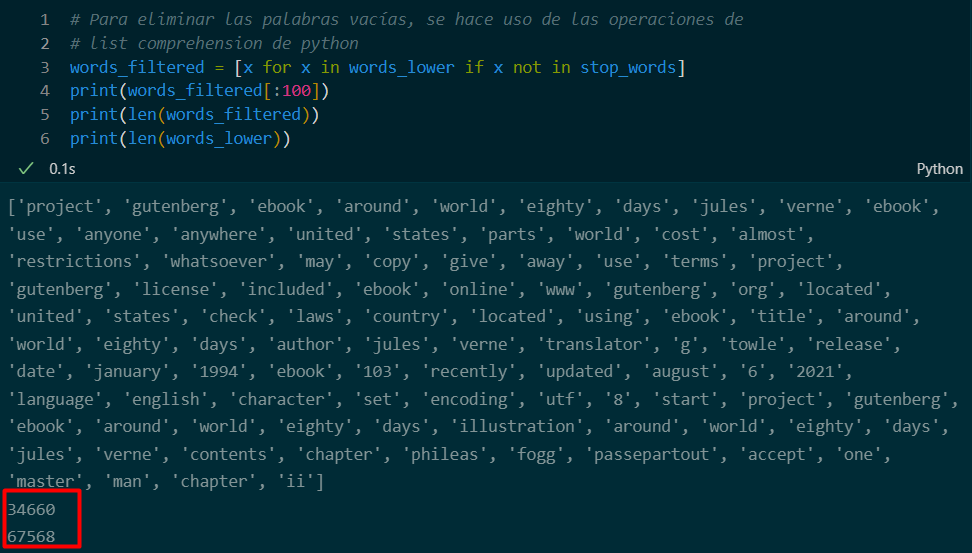
6. Ahora elimina palabras vacías

* Importa las palabras vacías en idioma inglés
* “from nltk.corpus import stopwords
* stop\_words = stopwords.words('english')”
* Imprime todas las palabras vacías, revísalas, y escribe en tu reporte las primeras 5.
* Elimina palabras vacías y escribe nuevamente las 100 primeras palabras.



Estas palabras son, en su mayoría, conectores, prefijos, sufijos, entre otros recursos literarios que, en un proceso de recuperación de la información, solamente retrasarían el procesamiento de los datos.

Ahora, se eliminarán de la colección de palabras vacías.



Aquí se ve como casi la mitad de las palabras del vocabulario eran palabras vacías.

**IV. DISCUSIÓN Y RESULTADOS**

Si comparamos la cantidad de palabras que existían antes de quitar las stop words y después, podemos observar que antes de el procesamiento había 67568 y después nos quedan 34908 palabras, en conclusión se eliminaron 32908 palabras vacías del texto. Es así que podemos observar lo importante que es preprocesar los texto para que solo nos quedan las palabras importantes.

**V. CONCLUSIONES**

La utilización de paqueterías como lo es NLTK facilita el procesamiento de texto, simplifica el código y ayuda a la automatización, sin embargo, presenta limitaciones para proyectos más complejos donde se tiene que recurrir a la ambigüedad para lograr resolverlo, pero puede afectar el rendimiento.

**VI. BIBLIOGRAFÍA**

1. Python Software Foundation. (2022). *re — Regular expression operations — Python 3.10.6 documentation*. Python Documentation. Recuperado 23 de agosto de 2022, de https://docs.python.org/3/library/re.html
2. *Regex Cheat Sheet/Basics -*. (2021, 31 marzo). Rackspace. Recuperado 23 de agosto de 2022, de <https://docs.rackspace.com/support/how-to/regex-cheat-sheet-basics/>