**Proyecto Final: Recuperación de la Información**

Reyes, Alfonso.

Benemérita Universidad Autónoma de Puebla

Recuperación de la información

Fecha: 29 de agosto de 2022

**Resumen:** Se realizó la representación según el modelo vectorial y los esquemas de pesado tf, tf-idf del corpus contenido en el archivo: “blog-gender-dataset.xlsx”, el cual se utilizó para generar una clasificación de género hombre-mujer de las entradas de blog.

**Palabras clave:** Modelo vectorial, Python, clasificación de textos, tf-idf.

**I. INTRODUCCIÓN**

Para este pequeño proyecto se cuenta con un corpus compuesto por entradas de usuarios en un blog cualquiera en internet, las cuales están etiquetadas originalmente por hombre-mujer. El objetivo final de este sistema es el de poder clasificar las entradas del blog correspondiendo a si fueron escritas por un hombre o por una mujer; para esto, recurriremos a la representación de cada documento (entrada del blog) por medio del modelo vectorial, conforme a los esquemas tf y tf-idf. A su vez, se realizarán comparaciones sobre la exactitud de estos cálculos conforme una exploración con el vocabulario completo y otra con el vocabulario reducido, para determinar si estas diferencias hacen que nuestro sistema sea más o menos preciso en la clasificación de textos.

**II. OBJETIVO Y PLANTEAMIENTO DEL PROBLEMA**

Clasificar las entradas del blog del corpus *“blog\_gender-dataset.xlsx”* considerando si estas fueron escritas por hombres o mujeres utilizando el modelo vectorial para la representación de textos.

**III. DESARROLLO EXPERIMENTAL**

**Materiales:**

Instalar Python y NLTK y el corpus *“blog\_gender-dataset.xlsx”.*

**Descripción:**

1. Elegir alguno de los corpus localizados en Teams en la carpeta de Proyecto. Encontrarán ahí documentos para entrenamiento y pruebas o en su defecto tome el 80 por ciento de datos para entrenar y el 20 para probar.

2. Preprocesen los documentos para obtener la representación vectorial de los mismos.

3. Utilicen Weka o Python para clasificar los textos, experimenten clasificando con tf con tf-idf y reduciendo el vocabulario

4. Generen su reporte en el mismo formato de las prácticas reportando la exactitud, precisión, recuerdo y medida F obtenidas.

**Desarrollo:**

1. Leer el archivo:

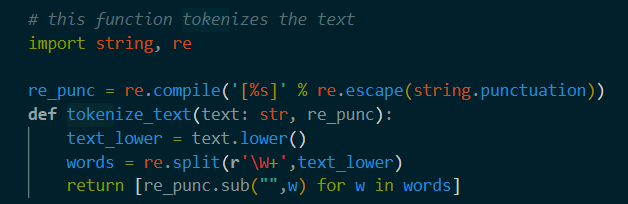
Para esto, se usó la librería *pandas* de Python para recuperar el dataset y mostrarlo brevemente.

Observamos que contamos con 3232 documentos ya clasificados.

1. Preprocesamiento

Ahora, debemos realizar el preprocesamiento propio de un sistema de recuperación de la información, comenzando por la obtención del vocabulario total, tokenizando el texto excluyendo signos de puntuación, números y palabras vacías.

El truncamiento será realizado con porter-stemmer, haciendo uso de la la función *nltk.stem.porter.stem().*

Para esto, también se usó el apoyo de una función llamada tokenize\_text (text, re\_punc), que nos ayudará a realizar dicho preprocesamiento.



Esta es la primera oración del primer documento originalmente.



Después de realizar dicho preprocesamiento, vemos que ahora todas estas palabras se encuentran tal y como describimos.

Notamos que todas las palabras están en minúsculas, no aparece ya el punto y seguido y que tampoco aparecen ya palabras como “it” o “of”, además de que el truncamiento provoca que las palabras plurales o conjugadas sean reducidas a una versión simplificada de las mismas.

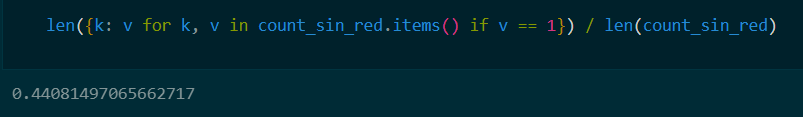
1. Reducción del vocabulario

Lo que podemos realizar a continuación es testear acerca de la distribución de las palabras, **para buscar reducir el vocabulario**.

Para este análisis, podemos revisar los quantiles (n=20) de la frecuencia de aparición de las palabras:

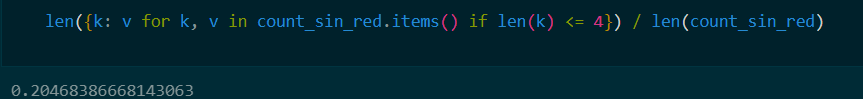


Se nota que la mayoría de las palabras tienen una aparición de entre 1 y 2 veces en el vocabulario, por lo que eliminar solamente las de 1 aparición quitaría aproximadamente el 40% del mismo lo que es demasiado.

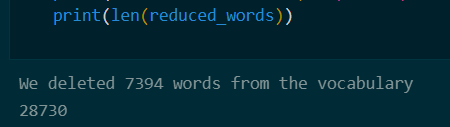


Para ser exactos, el 44%.

Evitando reducir demasiado el vocabulario, se analizará el vocabulario conforme el largo de cada palabra.



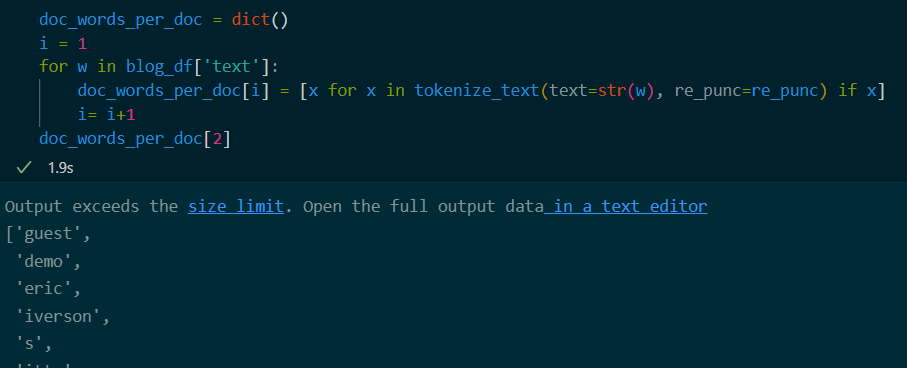
Un análisis rápido nos arroja que si reducimos el vocabulario con las palabras solo de largo 4 o más, lo acortaremos en solo un 20.46%, lo cual es ideal para nuestra aplicación.



Ahora, ya tenemos un vocabulario reducido y uno sin reducir los cuales usaremos en las tareas de clasificación. No sin antes realizar el modelado vectorial con los esquemas tf y tf-idf de ambos vocabularios.

1. Modelo vectorial

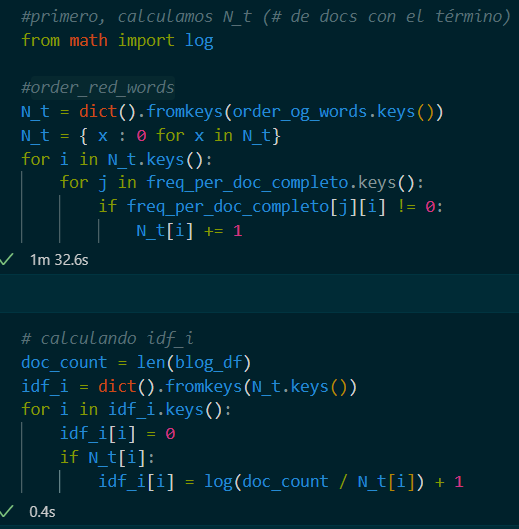
Para generar la representación con el modelo vectorial del corpus, primero necesitamos tokenizar cada documento de forma individual, para esto volvemos a utilizar la función antes usada tokenize\_text(text, re\_punc) para hacer esto documento a documento.



Ahora, lo que necesitamos es hacer el conteo de ocurrencias de cada término por documento; esto lo debemos hacer tanto con el conjunto completo como reducido.

Esto nos genera los esquemas tf del corpus, lo cual procedemos a guardar en los archivos “tf\_completo.json” y “tf\_reducido.json”.

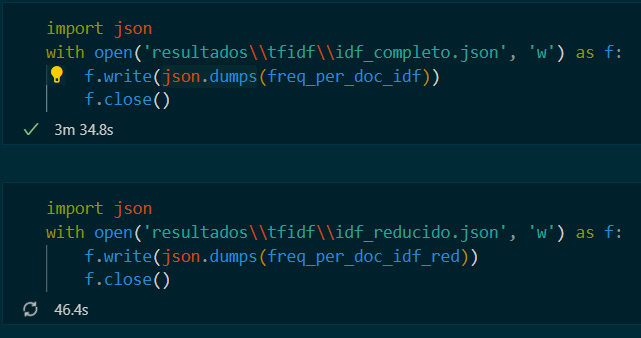
1. Tf-idf

Buscando generar los esquemas con tf-idf, primero debemos obtener ciertas matrices como la matriz N\_t y la matriz idf\_i, que requirere de calcular el idf con la fórmula:

Lo cual podemos hacer utilizando las palabras ordenadas en orden alfabético y las listas de frecuencias de los documentos de la colección.

Luego, por cada documento se realiza la multiplicación del arreglo idf\_i por cada término de cada documento, formando así la representación con tf-idf tanto del vocabulario completo como el reducido.

Finalmente, estas representaciones también son guardadas en archivos para ser finalmente clasificadas.



**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/>