Primera Tarea Programada - Recuperación de Información Textual

Grupo: 01

Estudiante: Austin Edward Hakanson Hidalgo

Profesor: José Enrique Araya Monge

Fecha de Entrega: 12/03/20

Introducción

El objetivo de la tarea programada es generar un modelo de lenguaje unigrama a partir de una colección de páginas web en español sobre países y dependencias del mundo. Además se desean obtener todas las referencias de un archivo hum o html dado.

Funcionalidad

Para ello se desarrollaron las siguientes herramientas en el lenguaje de programación Python:

Entre las librerías utilizadas se tienen:

- **bs4:** Permite la manipulación sencilla de archivos html o htm.
- **re:** Permite el uso de expresiones regulares para validar texto y extraer información.
- **unicodedata:** Permite obtener información acerca de la composición de caracteres en cadenas en formato Unicode (el estándar de Python 3), útil para el procesamiento de términos (extraer tildes, etc).
- **pathlib:** Permite recorrer directorios y realizar operaciones sobre ellos. Útil para procesar un colección de archivos separada en carpetas.

genere_LM(*Ruta, ArchivoLM*): Esta herramienta tiene por objetivo generar un modelo de lenguaje de tipo unigrama a partir de un texto html o htm. Para lograrlo, se extrae todo el texto del archivo htm o html que recibe por parámetro y luego lo separa en palabras que pertenezcan exclusivamente al conjunto [a-zñ]. Una vez hecho esto se debe contar el número de apariciones de cada término procesado y dividirlo entre el total de palabras en el texto para determinar el peso de cada término. Finalmente, los resultados se escriben en un archivo de texto con el nombre recibido por el parámetro *ArchivoLM.*

opr(LM_específico LM_general Escalafón): Esta herramienta se encarga de **O**btener **P**alabras **R**elacionadas entre los dos archivos de texto que contienen un modelo de lenguaje unigrama y generar un escalafón donde se reflejen los términos con mayor relevancia para el documento específico. Dicho escalafón se escriben en un archivo de texto con el nombre pasado por parámetro.

extraer_refs(RutaArchivo, ArchivoRefs): Esta herramienta recibe la ruta de un archivo hum o html, extrae las referencias contenidas en las etiquetas <a> href="referencia" y las escribe en un archivo con el nombre pasado por parámetro.

Resultados

Primeras 20 líneas del archivo lm_geo.txt:

a: 0.014588310012987897
aa: 6.043438034981617e-06
aaadonta: 5.494034577256016e-07
aaaz: 5.4940345772560155e-06
aabz: 5.494034577256016e-07

aacz: 5.494034577256016e-07
aadu: 5.494034577256016e-07
aaiun: 3.845824204079211e-06
aaiunbojadorsaguia: 5.494034577256016e-07
aaj: 5.494034577256016e-07
aalborg: 1.6482103731768046e-06
aali: 2.1976138309024064e-06
aalto: 1.0988069154512032e-06
aana: 5.494034577256016e-07
aap: 5.494034577256016e-07
aar: 5.494034577256016e-07
aarafa: 1.0988069154512032e-06
aarau: 5.494034577256016e-07
aarbecht: 5.494034577256016e-07
aardgas: 5.494034577256016e-07

Primeras 20 líneas del archivo lm_cr.txt:

a: 0.014698533086397977
abajo: 5.879413234559191e-05
abandonaran: 2.9397066172795955e-05
abandonaron: 2.9397066172795955e-05
abandono: 5.879413234559191e-05
abangares: 5.879413234559191e-05
abanico: 2.9397066172795955e-05
abatido: 2.9397066172795955e-05
abel: 0.00011758826469118382
abelardo: 2.9397066172795955e-05
abiertamente: 2.9397066172795955e-05
abogado: 2.9397066172795955e-05
abogaron: 2.9397066172795955e-05
abolicion: 0.00011758826469118382
abolido: 2.9397066172795955e-05
abolio: 2.9397066172795955e-05
abolir: 5.879413234559191e-05
aborigen: 8.819119851838787e-05
aborigenes: 0.00011758826469118382
abre: 8.819119851838787e-05

Primeras 20 líneas del archivo opr_cr.txt:

1- acostumbrado: 53.5072463768116
2- boruca: 53.5072463768116
3- boyero: 53.5072463768116
4- braulio: 53.5072463768116
5- cartin: 53.5072463768116
6- cañasjerez: 53.5072463768116
7- ccss: 53.5072463768116
8- chirripo: 53.5072463768116
9- cleto: 53.5072463768116
10- deredia: 53.5072463768116
11- diquis: 53.5072463768116
12- euned: 53.5072463768116

13- garabito: 53.5072463768116
14- garcimuñoz: 53.5072463768116
15- huetar: 53.5072463768116
16- huetares: 53.5072463768116
17- irazu: 53.5072463768116
18- jamaiquinos: 53.5072463768116
19- libertario: 53.5072463768116
20- maleku: 53.5072463768116

Comentarios Finales

El programa funciona correctamente. Se pueden correr varias consultas al mismo tiempo escribiéndolas todas al final del archivo TP1.py (el archivo TP1.py adjunto tiene incluidas las consultas para obtener lm_geo.txt, lm_cr.txt, opr_cr.txt, refs_cr.txt). También se pueden correr las funciones escribiéndolas directamente en el Shell de Python con los parámetros correspondientes. Una **nota importante** es que la carpeta de Geografía/ debe estar en el mismo directorio que el programa TP1.py

En cuanto a los resultados mostrados en el archivo por_cr.txt, se evidencia en palabras como “boruca”, “irazu”, “maleku”,”ccss” que sí se logra encontrar palabras que son muy relevantes y específicas a Costa Rica.

Sin embargo, de los primeros 20 términos también se puede evidenciar que todos tienen el mismo peso (53.5072463768116). En efecto hay 918 con el mismo, de lo que se puede concluir que la técnica de usar modelos de lenguaje de tipo unigrama, por una parte sí logra destacar a las palabras más relevantes, pero por otra, no genera suficiente diferencia en los términos tratando a grandes porciones como igualmente relevantes.

Nótese que todo el código fue escrito en inglés (después de todo el lenguaje de programación Python está basado en ese idioma)