Laboratorio 2

Objetivo

Con el desarrollo de este laboratorio usted:

1. Construirá un motor de búsqueda de cursos.
2. Construirá un rastreador web que rastrea una copia de sombra del catálogo universitario para construir un índice simple.

El propósito de esta tarea es brindarle más experiencia en programación de Python y hacer que trabaje con documentos HTML extraídos de la web.

Fecha de entrega : 5 de marzo 23:59

Requerimientos / Preguntas del laboratorio

- Dada la consulta de una palabra en los cursos de la Universidad, ¿cómo mostrar las URLs por orden de relevancia?
- Defina una métrica de similitud para comparar dos cursos.
- Dado un listado de intereses, por ejemplo ['musica', 'composicion', 'instrumento'], el buscador debe retornar los cursos que más se relacionan a los intereses del usuario. ¿cómo se define una medida de similitud entre los cursos, y entre los cursos e intereses? ¿son dos métricas diferentes? ¿puede usar la misma métrica del punto anterior? ¿cuáles son las desventajas de usar la misma métrica, y cómo se mide el rendimiento?

Trabajando con URLs

URL se encuentra en el localizador de recursos uniforme. Una URL, por ejemplo:

```
https://educacionvirtual.javeriana.edu.co/reposteria-j-colombia
```

Tiene el siguiente formato:

```
protocolo://dirección del sitio/ruta/nombre de archivo
```

El protocolo campo (http, en el ejemplo) especifica qué protocolo se debe utilizar para interactuar con el recurso. Los protocolos comunes incluyen http, https y ftp. Trabajaremos con http y https, los protocolos de transporte de hipertexto. La dirección del sitio especifica el nombre del host (www), el nombre de dominio (educacionvirtual.javeriana) y el dominio de nivel superior (edu.co). La ruta especifica parte de la ubicación jerárquica (educacionvirtual) del archivo deseado (reposteria-j-colombia) en el sistema de archivos del host. La ruta al directorio de archivo se suministra mediante la configuración del programa del servidor web que se ejecuta en la máquina host.

La ruta completa en el sistema de archivos de la universidad para el ejemplo anterior es: https://educacionvirtual.javeriana.edu.co/reposteria-j-colombia. Las URL también pueden incluir un número de puerto, que no utilizaremos.

Dentro de una página web, un enlace puede referirse a una URL absoluta (es decir, comienza con http:// o https://) o una URL relativa, que se puede convertir en una URL absoluta. Una URL relativa se refiere a una página con una ubicación que es relativa a la página actual (muy similar a los nombres de ruta relativos en Linux).
URLs pueden usar un fragmento (denotado por un #) para referirse a una ubicación específica (conocida como un ancla) en una página. Para nuestros propósitos, solo necesitamos la parte de la URL que viene antes del #.

Algunas funciones útiles para trabajar con URLs:

- util.is_absolute_url(url): Toma una URL y devuelve verdadero si es una URL absoluta y falso en caso contrario.
- util.convert_if_relative_url(url1, url2): Toma la URL de una página (url1) y una URL encontrada en esa página (url2). Si url2 es una URL absoluta, entonces la función devolverá. Si url2 es una URL relativa, la función devolverá una URL absoluta equivalente. La función devolverá Ninguno si no puede convertir la URL.

- util.remove_fragment(url): Toma una URL y elimina el # y cualquier texto que lo siga en la URL. Por ejemplo, dado el URL relativo index.html#minorprogramincomputerscience, la función devolvería index.html. Si la URL no contiene un fragmento, la función simplemente devuelve la URL original.

Extrae la página web

Usamos **`requests`**, un paquete de Python, para hacer conexiones http y recuperar documentos. Las siguientes funciones wrapper:

- util.get_request(url): Toma una URL y devuelve un objeto **request**. Esta función devuleve None si la solicitud falla.
- **util.read_request(request):** Toma un objeto **request** y devuelve una cadena que contiene el documento HTML recuperado por la solicitud.

Esta función puede generar una advertencia de la forma:

```
WARNING:root:Some characters could not be decoded, and were replaced with
REPLACEMENT CHARACTER.
```

- **util.get_request_url(request):** Toma un objeto **request** y devuelve la URL asociada.

Note que la URL devuelta puede ser diferente a la URL proporcionada a la solicitud original a **get_request**. Esta aparente anomalía ocurre cuando la URL original redirige a otra URL.

Note que el crawler debe verificar los fallos, comprobar si la salida de alguna de estas funciones es None antes de continuar.

Catálogo

Estamos interesados en tres tipos de etiquetas HTML: a, div y p. Usaremos el primero para encontrar enlaces a otras páginas, el segundo para encontrar secciones de la página pertinentes a un curso en particular, y el tercero para extraer títulos y descripciones de cursos reales desde dentro de las etiquetas div.

Por ejemplo, aquí hay un enlace de una página en el catálogo de cursos:

```
<a href="https://educacionvirtual.javeriana.edu.co/propiedad-horizontal" target="_b
lank">
```

Las descripciones de los cursos están encerradas en una etiqueta div de la forma . Por ejemplo, aquí está la entrada para Propiedad Horizontal:

```
<div class="card-body">
<a href="https://educacionvirtual.javeriana.edu.co/propiedad-horizontal" target="_blank">
<b class="card-title text-primary font-weight-bold">Propiedad Horizontal</b> </a>
<p class="card-text mt-2 mb-0"><small class="text-muted">
<i class="fas fa-clock iconnn" aria-hidden="true"></i>
<b>Duración: </b>80 Horas</small></p>
<p class="card-text mb-0"><small class="text-muted">
<i class="fas fa-list-alt iconnn" aria-hidden="true"></i>
<b> Nivel: </b> Intermedio</small></p>
<p class="card-text mb-0"><small class="text-muted">
<i class="fas fa-calendar-alt iconnn" aria-hidden="true"></i>
<b> Fecha de inicio:
</b>29/02/2024</small>
</p>
<p class="card-text"><small class="text-muted">
<i class="fas fa-coins iconnn" aria-hidden="true"></i>
<b> Precio: </b> $ 2.130.000</small>
</p>
</div>
```

Los títulos y descripciones de los cursos se pueden encontrar en etiquetas p anidadas dentro de la etiqueta div.

Construya un índice que mapee palabras a listas de ids de cursos. Un id de curso es un String que identifica de forma única un código de curso específico ("propiedad-horizontal", por ejemplo).
Los ids de curso se pueden encontrar al principio de cada título de curso. Ciertos caracteres reservados en HTML deben ser reemplazados con caracteres. Para incluir un símbolo < como

texto debemos usar la entidad de caracteres <. Es necesario porque < es una palabra reservada en la sintaxis HTML.
Al utilizar el atributo .text de `beautifulsoup4`, convertirá el   en un carácter de espacio. Sólo deberá reemplazarlo con un espacio cuando construya un código de curso.

Secuencias

Si un grupo de cursos forma una secuencia, el catálogo de cursos de UJaveriana enumera el título del curso y la descripción tanto para la secuencia (class="item-programa ais-Hits-item col-12 m-0 p-0 border-0 shadow-none") como para los cursos individuales asociados con esa secuencia (`class="card-body"`).

Vea a continuación un ejemplo:

```
<div class="col">
<div class="card-body">
<a href="/errores-innatos-metabolismo" target="_blank">
<b class="card-title text-primary font-weight-bold">Errores innatos del
metabolismo de molécula pequeña</b> </a>
<p class="card-text mt-2 mb-0"><small class="text-muted"><i class="fas fa-clock
iconnn" aria-hidden="true"></i>
<b>Duración: </b>116 Horas</small></p>
<p class="card-text mb-0"><small class="text-muted">
<i class="fas fa-list-alt iconnn" aria-hidden="true"></i>
<b> Nivel: </b> Intermedio</small></p>
<p class="card-text mb-0"><small class="text-muted">
<i class="fas fa-calendar-alt iconnn" aria-hidden="true"></i>
<b> Fecha de inicio:
</b>Próximamente</small>
</p>
<p class="card-text"><small class="text-muted">
<i class="fas fa-coins iconnn" aria-hidden="true"></i>
<b> Precio: </b> $ 3.190.000</small>
</p>
</div>
```

Trabajando con HTML

Use Beautiful Soup para analizar y navegar por documentos HTML con el analizador `html5lib`. Aquí está la referencia para beautifulsoup4 que proporciona más detalles https://www.crummy.com/software/BeautifulSoup/bs4/doc/ .

Es posible que necesite actualizar beautifulsoup4 y html5lib. Puede hacerlo con los siguientes comandos en la línea de comandos de Linux:

```
pip3 install --upgrade beautifulsoup4
pip3 install --upgrade html5lib
```

Usted importará Beautiful Soup como bs4. Para crear un objeto **soup** a partir de una cadena de texto, que contiene el texto de un documento HTML, puede utilizar la declaración:

```
soup = bs4.BeautifulSoup(text, "html5lib")
```

Tenga en cuenta que, en nuestro caso, el parámetro texto será la cadena devuelta de una llamada a **read_request**.

Una vez que tenga un objeto **soup**, puede utilizar el método **find_all** para extraer una lista de los objetos de etiqueta en la **soup** que coinciden con un patrón particular. Por ejemplo, la expresión:

```
soup.find_all("div")
```

Devolvería una lista de etiquetas, una para cada bloque **div** que ocurre en el texto. Puede reducir la búsqueda incluyendo información sobre atributos y sus valores. Por ejemplo, la expresión:

```
soup.find_all('div', class_="  "col")
```

Devolvería una lista de objetos de etiqueta, uno para cada bloque **div** en el que el atributo de clase tiene el valor `"col"`. Tenga en cuenta el uso de class_ como el nombre del atributo en lugar del nombre esperado de class. El guion bajo es necesario porque **class** es una palabra reservada en Python.

Hay varias operaciones útiles que puede hacer en un objeto de etiqueta `t` y el nombre del atributo `a`: t.text
Extraer el texto asociado con la etiqueta t: `t.has_attr(a)`

Devuelve verdadero si la etiqueta contiene un atributo a: `t[a]`

Devuelve el valor asociado con el atributo a en la etiqueta. Esta expresión fallará si a no es un atributo de `t`.

También puede llamar a **find_all** en objetos de etiqueta para extraer etiquetas anidadas.

Ejercicio

Un motor de búsqueda se basa en un índice que mapea palabras a las páginas en las que aparecen. Este índice se construye rastreando la web e indexando la información en las páginas

que el rastreador visita. El rastreador comienza en una página específica y sigue los enlaces hacia otras páginas, y desde esas páginas hacia las demás, y así sucesivamente hasta que ha visitado todas las páginas alcanzables desde la página inicial.

Su labor es construir una función llamada go en un archivo crawler.py. Esta función recibe como argumentos 1) el número de páginas para rastrear, 2) el nombre de un archivo JSON que contiene el diccionario de ids de curso, y 3) el nombre del archivo de salida como argumentos. Su implementación debe rastrear el catálogo de cursos para crear un índice que mapea palabras a listas de ids de cursos, y luego generar un archivo CSV a partir del índice.

```
def go(n:int, dictionary:str, output:str):
```

Rastreador

Su rastreador debe:
1. Comenzar en la URL de inicio especificada.
2. Visitar páginas que tengan URLs verificadas para seguir.
3. Detenerse después de haber visitado el número máximo de páginas especificado.
4. Visitar las páginas en orden de ocurrencia (primero en entrar, primero en salir).
5. Visitar una página sólo una vez para evitar ciclos (por ejemplo, A->B->C->A).
6. Evite llamar a **get_request** en la misma URL varias veces.

La función, `util.is_url_ok_to_follow(url, domain)`, que toma una URL y un dominio y devuelve `True`, si la URL:
1. Es una URL absoluta,
2. Cae dentro de un dominio especificado,
3. No contiene un símbolo "@" o "mailto:",
5. Tiene un nombre de archivo que termina sin extensión o la extensión "html".

La URL de inicio es : https://educacionvirtual.javeriana.edu.co/nuestros-programas-nuevo y el dominio limitante es: `educacionvirtual.javeriana.edu.co`

Usted debe visitar los enlaces en orden "primero en entrar, primero en salir", lo que significa que una cola es la estructura de datos apropiada para gestionar los enlaces. Una cola típicamente tiene operaciones para añadir valores al final, para quitar valores del frente o cabeza, y para determinar si la cola está vacía. Puede implementar una cola con unas pocas operaciones de lista simples o puede utilizar la biblioteca Python Queue.

Al rastrear las páginas visitadas, la URL devuelta por **get_request_URL** puede ser distinta de la URL utilizada en la llamada a **get_request**.

No construya un rastreador recursivo. En su lugar, utilice una cola para realizar un seguimiento de las páginas que se han visitado.

Indexador

Escriba el código que construye un índice que mapea una palabra a una lista de identificadores de cursos. Un identificador de curso debe incluirse en la entrada del índice para una palabra si la palabra aparece en el título o la descripción del curso en el catálogo de cursos. Su implementación debe construir el índice a medida que su rastreador visita las páginas.

Su indexador debe ser insensible a las mayúsculas y minúsculas. Por ejemplo, su indexador no debe distinguir entre "Foo" y "foo". Recuerde que puede convertir una cadena, s, a minúsculas utilizando la expresión s.lower().

Una palabra es una cadena de longitud mayora a 1, que comienza con una letra (A-Z o a-z) y contiene sólo letras, dígitos (0-9) y/o un guion bajo (_). Elimine los signos de puntuación: !, . o : al final de una palabra. Puede utilizar una expresión regular.

Algunas palabras ocurren con mucha frecuencia en el texto normal en español ("un", "y", "o") y algunas ocurren con mucha frecuencia en el catálogo ("profesionales", "estudiantes", "curso", etc). Identifique un listado de las palabras más frecuentes, que no aportan información, y que su indexador no debe incluir en el índice.

Sí existen listas para un bloque de curso, su indexador debe mapear palabras en el título principal y la descripción a todos los cursos en la lista. Además, debe mapear palabras que aparecen sólo en la descripción de una lista al identificador de curso para esa lista, no todos los identificadores para todos los cursos en la secuencia.

Una función útil, llamada util.find_sequence(tag), para trabajar con subsecuencias. Esta función toma una etiqueta **bs4** y comprueba las subsecuencias asociadas. Si existen subsecuencias, la función devuelve una lista de los objetos de etiqueta **div** para la subsecuencia; de lo contrario, devuelve una lista vacía.

Salida

Cargue el índice generado en una base de datos relacional. Para facilitar este proceso, debe crear un archivo CSV a partir de su índice. Cada línea del archivo debe contener un identificador de curso y una palabra, en ese orden, y separados por una barra (|). Aquí están las primeras líneas del archivo ejemplo:

```
Curso_0001|aba
Curso_0002|aba
Curso_0003|aba
Curso_0001|abaá
Curso_0002|abaá
Curso_0003|abaá
Curso_0001|ababol
```

```
Curso_0002|ababol
Curso_0001|ábaco
Curso_0002|ábaco
```

Tenga en cuenta que la misma palabra puede aparecer en varias líneas del archivo (correspondiente a la palabra que aparece en las descripciones de varios cursos). Cualquier par de identificador de curso/palabra dada debe ocurrir como máximo una vez.

Aunque la salida anterior está ordenada por palabra (primaria) e identificador de curso (secundaria), no es necesario que produzca la salida en orden ordenado.

Comparar y buscar cursos

A partir del índice, construya un comparador de cursos y un buscador.

Comparador de cursos

El objetivo es determinar una función con una medida de similitud entre objetos como lo estudiamos en clase. El propósito es ayudar a los profesores que proponen nuevos cursos y saber si ya existen cursos similares.

Escriba un método para comprar cursos e identificar los cursos similares. La función `compare(curso1, curso2)` retorna un valor float [0, 1] con la medida de similitud. Esta función le permitirá pre-calcular una matriz de similitud entre todos los cursos. Describa el funcionamiento de su función de similitud en el informe del laboratorio y documente hallazgos que considere relevantes.

Buscador de cursos

Diseñe e implemente una función que le permita a un usuario, buscar cursos relevantes a sus intereses. Por ejemplo, si un usuario está interesado en fotografía puede indicar sus intereses “luminosidad”, “enfoque” , “composición” y el buscador retornará un listado de cursos (ids y URLs) por orden de relevancia.

- Determine una métrica de similitud entre el cursos y los intereses.
- Implemente una medida de relevancia para listar primero los cursos más relevantes.

Implemente un método `search(keywords)` que retorna un listado de URLs.

Organización del Código

Defina la descomposición apropiada del trabajo para esta tarea. No debe hacer todo en una función.

Calificación

Las tareas de programación se calificarán de acuerdo con una rúbrica general. Específicamente, asignaremos puntos por completitud, corrección, diseño y estilo. Los pesos serán:

1. Completitud: 65%
2. Exactitud: 10%
3. Diseño: 15%
4. Estilo: 10%

Limpieza

Antes de enviar su trabajo final, debe, eliminar:

1. Cualquier declaración de impresión que haya agregado con fines de depuración y
2. Todos los comentarios en línea.

Compruebe el código con la guía de estilo de Google styleguide/pyguide.md at gh-pages · google/styleguide · GitHub.

1. ¿Utilizó buenos nombres de variables?
2. ¿Tiene alguna línea que sea demasiado larga, etc.?
3. No elimine los comentarios de encabezado, es decir, las cadenas de triple comillas que describen el propósito, las entradas y los valores de retorno de cada función.

Entrega

El trabajo se envía por la plataforma de la Universidad en **Laboratorio 2** o en el repositorio. Verifique que los commit y push son correctos antes de enviar la versión final.

La entrega se compone de lo siguiente:

- Archivo README.md con la descripción del lab y la documentación de las conclusiones.
- Archivo crawler.py con el código del rastreador.
- Archivo SQL con la tabla con la salida del rastreador.
- Archivo SQL con consultas para identificar en cuál URL se encuentra una palabra dada.
- Archivo search.py con la función de búsqueda de cursos dado un listado de intereses.
- Archivo compare.py con la función de similitud entre cursos, retorna [0,1]