TRABAJO PRACTICO

ESTRUCTURA DE DATOS

Tutora:
Valeria Backer

Alumnos
Brenda Tosini, Gonzalo Giuffre – Agustin Scazzino – Christian Maldonado

Informe del trabajo práctico de Estructura de Datos

DESARROLLO DE PROYECTO

El equipo ha trabajado de manera colaborativa en el proyecto, centrado en el desarrollo de un sistema integral de recolección, indexación y búsqueda de noticias. La fase inicial involucró la planificación y diseño del sistema, donde se discutieron las tecnologías y herramientas a utilizar.

Se comenzó con el módulo de recolección de noticias para obtener una buena base de información de las noticias de cada medio. Posteriormente se continuó con el indexador BSBI añadiendo y optimizando las funciones del código provisto en clase. Y, por último, se realizó el módulo de búsqueda booleana para poder obtener la lista de documentos de la consulta ingresada por el usuario.

La comunicación fluida y reuniones regulares han sido clave para la coordinación del equipo. La implementación del código se ha realizado de manera modular, permitiendo la integración eficiente de cada componente. El proyecto ha avanzado siguiendo un enfoque iterativo, facilitando la detección temprana de problemas y la adaptación a cambios. La colaboración activa y la combinación de habilidades individuales han sido fundamentales para el progreso exitoso del proyecto.

RECOLECCIÓN DE NOTICIAS

Método `realizar_solicitudes_guardar_xml`

Este método se encarga de realizar solicitudes a los feeds RSS de diferentes medios, recopilar las noticias y guardar la información en archivos XML.

- Entrada:

- archivo_config: Ruta del archivo de configuración (en formato INI) que contiene información sobre los medios y sus URLs.

- Pasos:

Lectura del Archivo de Configuración:

- Utiliza la biblioteca `configparser` para leer el archivo de configuración y obtener información sobre los medios y sus secciones.

Iteración sobre Medios y Secciones:

- Itera sobre cada sección del archivo de configuración, que representa un medio de noticias.

Construcción de URL y Rutas de Archivos:

- Construye la URL completa para la sección actual del medio.
- Define la ruta del archivo XML donde se almacenarán las noticias de esa sección.

Manejo del Archivo XML:

- Intenta abrir el archivo XML correspondiente.
  - Si el archivo no existe, crea un nuevo árbol XML con una raíz llamada "channel".
  - Si el archivo ya existe, obtiene un conjunto de noticias ya recopiladas para evitar duplicados.

Recopilación de Noticias del Feed RSS:

- Utiliza la biblioteca `feedparser` para analizar el feed RSS desde la URL construida.
- Recorre las entradas del feed, que representan noticias.

Generación del Árbol XML:

- Para cada noticia, verifica si ya está recopilada para evitar duplicados.
- Si la noticia no está recopilada, agrega un elemento al árbol XML con información sobre la noticia (título, fecha, descripción, enlace).

Guardado del Árbol XML:

- Guarda el árbol XML actualizado en el archivo correspondiente.
- Imprime un mensaje indicando que las noticias se han agregado correctamente o informa de cualquier error.

Método `repetir_recoleccion’

Este método se encarga de repetir la recolección de noticias a intervalos regulares utilizando la biblioteca `schedule`.

- Entrada:

- archivo_config: Ruta del archivo de configuración que contiene información sobre los intervalos de tiempo para recoger noticias.

- Pasos:

Lectura de la Configuración:

- Utiliza `configparser` para obtener el intervalo de tiempo en minutos desde el archivo de configuración.

Recolección Inicial de Noticias:

- Llama al método `realizar_solicitudes_guardar_xml` para realizar la recolección inicial de noticias.

Programación de la Recolección Periódica:

- Utiliza la biblioteca `schedule` para programar la repetición del método `realizar_solicitudes_guardar_xml` con un intervalo de diez minutos.

Ejecución Continua:

- Entra en un bucle infinito donde ejecuta tareas programadas (`schedule.run_pending()`) y espera un tiempo antes de la próxima iteración (`time.sleep(int(intervalo_minutos))`).

Bloque `if __name__ == '__main__':`

Este bloque inicia la ejecución del script cuando se ejecuta directamente (no cuando se importa como módulo). Llama al método `repetir_recoleccion` con la ruta del archivo de configuración como argumento (`"../config/config.ini"`).

INDEXADOR BSBI

Implementamos un indexador utilizando el método de "Block Sort-Based Indexing" (BSBI) para la construcción de un índice invertido a partir de documentos XML que contienen noticias.

Aquí se explica la funcionalidad de cada método:

Método `__init__`

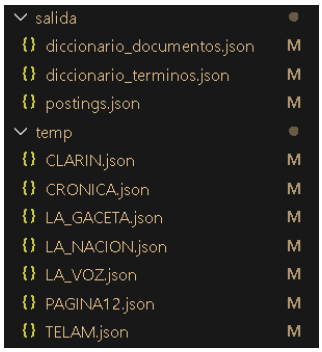
- Propósito: Inicializa la instancia del IndexadorBSBI.
- Parámetros:
    - `documentos`: Ruta del directorio que contiene los documentos XML.
    - `salida`: Ruta del directorio donde se almacenarán los resultados.
    - `temp`: Ruta del directorio temporal para almacenar archivos intermedios (por defecto, "../temp").
    - `language`: Idioma para el proceso de lematización (por defecto, 'spanish').
    - Acciones:
    - Inicializa variables de instancia.
    - Llama a los métodos `__generar_docID`, `_generar_lista`, y `__indexar` para realizar la indexación.

Método `_generar_lista`

- Propósito:Genera una lista de rutas de los documentos en el directorio de documentos.

Acciones:

- Recorre el árbol de directorios y archivos en `documentos` utilizando `os.walk`.
- Agrega las rutas de los archivos a `_lista_documentos`.



Método `__generar_docID`

- Propósito: Asigna un identificador único a cada documento.
- Acciones:
    - Crea un diccionario `doc_path_to_docID` donde las claves son las rutas de los documentos y los valores son los identificadores únicos asignados a cada documento.
    - Almacena el diccionario resultante en `_doc_path_to_docID`.

Método ‘__lematizar

- Propósito: Lematiza una palabra utilizando un stemmer de Snowball.
- Parámetros:
    - `palabra`: La palabra a lematizar.
- Retorno: La palabra lematizada.

Método `__indexar`

- Propósito: Realiza la indexación BSBI.
- Acciones:
    - Itera sobre bloques de documentos, generando el índice invertido para cada bloque.
    - Guarda cada índice invertido en archivos JSON en el directorio temporal.
    - Combina y ordena los bloques para formar el índice completo.
    - Guarda el índice completo en `postings.json`.
    - Guarda los diccionarios `term_to_termID` y `doc_path_to_docID` en archivos JSON.

Método `__invertir_bloque`

- Propósito: Invierte un bloque de pares (término, documento) para formar el índice invertido.
- Parámetros:
    - `bloque`: Lista de pares (término, documento).
- Retorno: Diccionario del índice invertido.

Método `__guardar_bloque_intermedio`

- Propósito: Guarda un índice invertido de bloque en un archivo JSON.
- Parámetros:
    - `bloque`: Índice invertido del bloque.
    - `nombre_bloque`: Nombre del bloque.
- Retorno: Ruta del archivo donde se guarda el índice invertido.

Método `__intercalar_bloques`

- Propósito: Combina bloques para formar el índice completo.
- Parámetros:
- lista_bloques`: Lista de rutas de archivos de bloques intermedios.

Método `__guardar_diccionario_terminos`

- Propósito: Guarda el diccionario `term_to_termID` en un archivo JSON.

Método `__guardar_diccionario_documentos`

- Propósito: Guarda el diccionario `doc_path_to_docID` en un archivo JSON.

Método `__parse_next_block`

- Propósito: Genera bloques de pares (término, documento) para cada medio de noticias.
- Retorno: Generador que produce bloques y nombres de bloques.

Método `strip_tags`

- Propósito: Elimina las etiquetas HTML de una línea.
- Parámetros:
    - linea: La línea con posibles etiquetas HTML.
    - Retorno:La línea sin etiquetas HTML.

Bloque `if __name__ == '__main__':`

- Propósito: Ejecuta el código si el script se ejecuta directamente (no se importa como módulo).
- Acciones:
    - Crea una instancia de `IndexadorBSBI`.
    - Imprime la longitud de la lista de documentos y un mensaje indicando que los archivos se cargaron correctamente.

BUSQUEDA BOOLEANA

Este script implementa una clase BusquedaBooleana que realiza búsquedas booleanas sobre un índice invertido construido anteriormente. A continuación, se explican las funciones principales de la clase:

Método consulta_booleana

- Propósito: Realiza una consulta booleana que se pasa por el menú.
- Parámetros:
  - parte1: Primera parte de la consulta.
  - condicional: Operador condicional (1 para AND, 2 para OR, 3 para NOT).
  - parte2: Segunda parte de la consulta.
- Acciones:
  - Verifica si parte2 sigue el patrón de una subconsulta.
  - Si es una subconsulta, realiza la subconsulta utilizando los operadores AND, OR o NOT según el valor de condicional.
  - Realiza la consulta general utilizando los operadores AND, OR o NOT según el valor de condicional.
  - Retorna el resultado de la consulta.

Método lematizar

- Propósito: Lematiza una palabra utilizando un stemmer de Snowball.
- Parámetros:
  - palabra: La palabra a lematizar.
- Retorno: La palabra lematizada.

Método buscar

- Propósito: Busca la palabra en el índice invertido y devuelve la lista de documentos que contienen la palabra.
- Parámetros:
  - palabra: La palabra a buscar.
- Retorno: Conjunto de documentos que contienen la palabra.
- Acciones:
  - Lematiza la palabra.
  - Carga el diccionario de términos desde diccionario_terminos.json.
  - Obtiene el identificador de la palabra en el diccionario.
  - Carga el índice desde postings.json.
  - Retorna la lista de documentos asociada a la palabra (si existe).

MENU DE USUARIO

Este script implementa un menú interactivo para realizar consultas booleanas utilizando la clase BusquedaBooleana. Aquí se explica cómo funciona:

Método menu_consulta

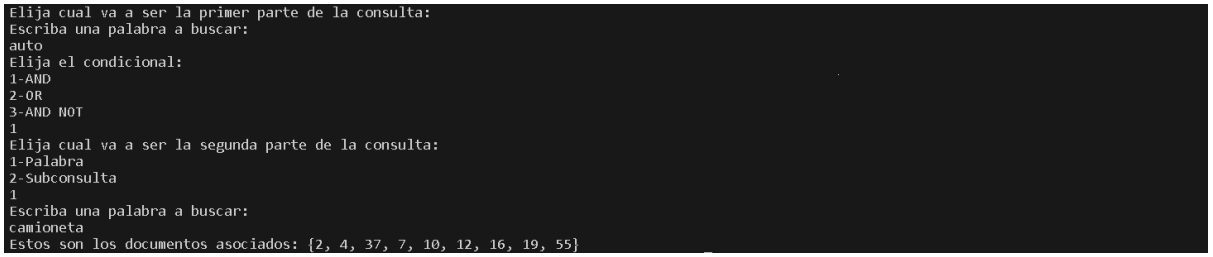
- Propósito: Presenta un menú interactivo para que el usuario ingrese los parámetros de una consulta booleana y luego muestra los documentos asociados al resultado de la consulta.
- Acciones:
    - Solicita al usuario que elija la primera parte de la consulta (una palabra a buscar).
    - Solicita al usuario que elija el condicional (AND, OR, AND NOT).
    - Solicita al usuario que elija la segunda parte de la consulta (una palabra o una subconsulta).
    - Si elige una palabra, solicita la palabra al usuario.
    - Si elige una subconsulta, solicita la primera palabra, el condicional y la segunda palabra de la subconsulta.
    - Crea una instancia de BusquedaBooleana y realiza la consulta llamando al método consulta_booleana.
    - Muestra los documentos asociados al resultado de la consulta.

Bloque if __name__ == '__main__':

- Propósito: Inicia la ejecución del script si se ejecuta directamente (no se importa como módulo).
- Acciones:
    - Crea una instancia de Menu.
    - Llama al método menu_consulta para realizar la consulta booleana.

Uso Ejemplo:

El script se ejecuta y presenta un menú interactivo. El usuario elige la primera parte de la consulta, el condicional y la segunda parte de la consulta. Luego, el programa utiliza la clase BusquedaBooleana para realizar la consulta y muestra los documentos asociados al resultado.



Este diseño permite a los usuarios interactuar con el programa y realizar consultas booleanas de manera flexible.