PROYECTO DE PRÁCTICAS DE SAR

Trabajo en grupo (grupos de 2 personas).

Objetivo:

El proyecto consiste en la implementación en python de un sistema de indexación y recuperación de noticias. El alumno deberá desarrollar dos aplicaciones distintas: la primera extraerá las noticias de una colección de documentos alojados en un directorio, las indexará y guardará en disco los índices creados; la segunda leerá los índices y recuperará aquellas noticias relevantes para las consultas que se realicen.

La nota máxima será de 1,5 puntos. Las aplicaciones deberán contar con unas funcionalidades mínimas que se puntuarán en total con un máximo de 0,75 puntos. Opcionalmente, se podrán ampliar las funcionalidades para obtener mayor nota, hasta un máximo de 0,75 puntos adicionales.

Entrega: hasta el 19 de Mayo a través de la Tarea correspondiente en PoliformaT.

Evaluación: 2 sesiones de evaluación, 22 de Mayo y 29 de Mayo.

Funcionalidades básicas (0,75 puntos):

Indexador:

Funcionalidades:

- Aceptará dos parámetros de entrada: el primero el directorio donde está la colección de noticias y el segundo el nombre del fichero donde se guardará el índice.
- Procesará los documentos y extraerá las noticias: eliminar símbolos no alfanuméricos (comillas, sostenidos, interrogantes,...), extraer los términos (consideraremos separadores de términos los espacios, los saltos de línea y los tabuladores). No se deben distinguir mayúsculas y minúsculas.
- A cada documento se le asignará un identificador único (docid) que será un entero secuencial.
- A cada noticia se le asignará un identificador formado por dos enteros: el identificador del documento (docid) que la contiene y la posición (posid) que la noticia ocupa dentro de dicho documento.
- Se deberá crear una fichero invertido accesible por término. Cada entrada contendrá una lista con los documentos en los que aparece ese término.
- Toda la información necesaria para el recuperador de noticias se guardará en un único fichero en disco.

Recomendaciones de Implementación:

- Una versión esquemática del algoritmo del indexador podría ser:

```
mientras hay_documentos:
    doc ← leer_siguiente_documento()
    docid ← añadir_doc()
    mientras hay_noticias:
        noticia ← extraer_siguiente_noticia()
        noticia_limpia ← procesar_noticia()
        para termino en noticia_limia:
            añadir_noticia_al_postings_list_del_termino()
```

- El diccionario de documentos puede ser una tabla hash accesible por docid o una lista donde el docid indique la posición que la información del documento ocupa en la lista.

- El identificador de cada noticia puede ser una tupla (docid, posid).

- El fichero invertido puede ser una tabla hash implementada como un diccionario de python, indexado por término y que haga referencia a una lista con los postings asociados a ese término.

- La mejor forma de guardar los datos de los índices en disco es utilizar la librería ***cpickle/pickle*** que permite guardar un objeto python en disco. Si quieres guardar más de un objeto, puedes hacer una tupla con ellos, *(obj1, obj2, ..., objn)*, y guardar la tupla. Consulta la práctica del mono infinito.

Recuperador de noticias:

Funcionalidades:

- Aceptará un parámetro de entrada (el fichero que contiene los índices) y entrará en un bucle de petición de consulta y devolución de las noticias relevantes hasta que la consulta esté vacía.

- La búsqueda se hará en el cuerpo de las noticias. Las noticias relevantes para una consulta serán aquellas que contengan todos los términos de la misma (búsqueda binaria).

- La presentación de los resultados se realizará en función del número de resultados obtenidos:

    - Si sólo hay una o dos noticias relevantes. Se mostrará el titular y el cuerpo de cada noticia.

    - Si hay entre 3 y 5 noticias relevantes. Se mostrará el titular de cada noticia y un *snippet* del cuerpo de la noticia que contenga los términos buscados.

    - Si hay más de 5 documentos relevantes. Se mostrará el titular de las 10 primeras.

  En todos los casos se mostrará el nombre de los ficheros que contienen las noticias y se informará al usuario del número total de noticias recuperadas.

Recomendaciones de Implementación:

- Un *snippet* de un termino es una subcadena del documento que contiene el termino y un contexto por la izquierda y derecha. Prueba diferentes tamaños de contexto.

Funcionalidades ampliadas (hasta 0,75 puntos):

Para obtener la máxima puntuación, además de las funcionalidades básicas, se deberán implementar correctamente al menos cuatro de las siguientes funcionalidades extra:

- Permitir utilizar AND, OR y NOT en las consultas. El orden de evaluación de las conectivas (orden de prelación de las operaciones) será de izquierda a derecha.

Ejemplo: la consulta "*term1 AND NOT term2 OR term3*" deberá devolver las noticias que contienen "*term1*" pero no "*term2*" más las que contienen "*term3*".

- Permitir *opcionalmente* la eliminación de stopwords y el stemming de los documentos y las consultas. Se debe utilizar el mismo índice para hacer consultas por términos o por stems. Se necesitará añadir un parámetro adicional al recuperador de noticias

- Añadir índices adicionales para el titular de la noticia, la categoría y la fecha. En las consultas se podrán utilizar los prefijos "headline:", "text:", "category:" y "date:" junto a un término para indicar que ese término se debe buscar en el índice de los titulares, el cuerpo, la categoría o la fecha. Si no se indica nada el término se buscará en el índice del cuerpo de la noticia.

Por ejemplo: "headline:messi valencia" debería recuperar las noticias donde aparezca "messi" en el titular y "valencia" en el cuerpo de la noticia.

- Permitir la búsqueda de varios términos consecutivos utilizando las dobles comillas. Esto hace necesario el uso de postings list posicionales.

Ejemplo: buscar *"fin de semana"* encontraría sólo los documentos donde los tres términos aparecen de forma consecutiva, mientras que buscar *fin de semana* encontraría todos los documentos en los que aparecen los tres términos sin importar la posición.

- Devolver los documentos ordenados en función de su relevancia utilizando para ello una distancia entre el documento y la consulta.