Sistemas de Gestión de Datos y de la Información
Máster en Ingeniería Informática, 2021-22
Práctica Recuperación Información

Fecha de entrega: domingo 5 de diciembre de 2021, 23:55h

Entrega de la práctica
La práctica se entregará en un único fichero mediante el Campus Virtual de la asignatura. El fichero contendrá una carpeta por cada uno de los apartados.

Lenguaje de programación
Python **3.8** o superior.

Ficheros
Colección de mensajes `20news-18828.tar.gz` de un grupo de noticias, obtenido de `https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups`.

Calificación
Además del correcto funcionamiento del código se valorará también su claridad, su documentación y su organización en funciones auxiliares reutilizables.

Declaración de autoría e integridad
Todos los ficheros entregados contendrán una cabecera en la que se indique la asignatura, la práctica, el grupo y los autores. Esta cabecera también contendrá la siguiente declaración de integridad:

> *Declaramos que esta solución es fruto exclusivamente de nuestro trabajo personal. No hemos sido ayudados por ninguna otra persona ni hemos obtenido la solución de fuentes externas, y tampoco hemos compartido nuestra solución con otras personas. Declaramos además que no hemos realizado de manera deshonesta ninguna otra actividad que pueda mejorar nuestros resultados ni perjudicar los resultados de los demás.*

No se corregirá ningún fichero que no venga acompañado de dicha cabecera.

---

1. Recuperación vectorial [5pt]

Completar el esqueleto VectorialIndex.py para implementar la clase VectorialIndex que procesa una colección de documentos, crea un índice invertido con pesos TF-IDF y lo usa para acelerar consultas vectoriales y booleanas (únicamente conjunción) sobre la colección. La clase debe contener al menos los siguientes métodos:

- __init__(self, directorio)
  Recorre **recursivamente todos los ficheros** que hay en directorio y crea un índice invertido para relacionar las palabras con una lista de parejas (documento, peso). Los pesos se deben calcular usando la técnica TF-IDF, y los documentos dentro del índice deben representarse como números enteros en lugar de rutas completas para minimizar el espacio que ocupan en memoria. Para que todos consideremos la misma *definición* de palabra, se debe usar la función extrae_palabras que aparece en el esqueleto para extraer las palabras de una línea de texto. Es importante que el índice invertido almacene la **mínima información necesaria** para optimizar su tamaño.

- consulta_vectorial(self, consulta, n=3)
  Dada una consulta representada como una cadena de palabras no repetidas separadas por espacios, devuelve una lista de parejas (ruta_fichero, relevancia) con los n resultados más relevantes usando el modelo de recuperación vectorial ordenadas de mayor a menor relevancia.

- consulta_conjuncion(self, consulta)
  Dada una consulta representada como una cadena de palabras no repetidas separadas por espacios que se entiende como una conjunción, devuelve una lista de rutas de fichero con todos los resultados en los que aparecen **todas las palabras** de la consulta.

En los métodos consulta_vectorial() y consulta_conjuncion() el índice invertido se debe utilizar de la manera más eficiente posible para resolver la consulta.

La clase VectorialIndex debe funcionar correctamente al ser importada y usada de la siguiente manera:

```
>>> from VectorialIndex import VectorialIndex
>>> i = VectorialIndex('20news-18828')
>>> i.consulta_vectorial('DES Diffie-Hellman', n=5)
[('20news-18828/sci.crypt/15991', 0.5096209025568682),
 ('20news-18828/sci.crypt/15826', 0.40780449792065276),
 ('20news-18828/sci.crypt/16141', 0.3327934131638604),
 ('20news-18828/sci.crypt/15894', 0.31276008256792764),
 ('20news-18828/sci.crypt/15592', 0.29572512521719024)]
>>> i.consulta_conjuncion('DES Diffie-Hellman')
['20news-18828/sci.crypt/15670',
 '20news-18828/sci.crypt/14831',
 '20news-18828/sci.crypt/15746',
 '20news-18828/sci.crypt/15493',
 '20news-18828/sci.crypt/15991',
 '20news-18828/sci.crypt/15644',
 '20news-18828/sci.crypt/15301',
 '20news-18828/sci.crypt/15241']
```

2. Índice invertido completo [5pt]

Completar el esqueleto CompleteIndex.py para implementar la clase CompleteIndex que procesa una colección de documentos, crea un índice invertido completo y lo usa para acelerar consultas de frase exacta sobre la colección. La clase tiene un interfaz similar al índice vectorial y debe contener al menos los siguientes métodos:

- __init__(self, directorio)
  Recorre **recursivamente todos los ficheros** que hay en directorio y crea un índice invertido completo para relacionar cada palabra con una lista de tuplas (numero_doc, l_pos), donde numero_doc es el número de documento y l_pos contiene una lista de posiciones. Como granularidad usaremos el número de palabra dentro del documento empezando en 1. Al igual que en el apartado anterior, se debe usar la función extrae_palabras que aparece en el esqueleto para extraer las palabras de una línea de texto.

- consulta_frase(self, frase).
  Dada una consulta representada como una cadena de palabras separadas por espacios, devuelve una lista con las rutas de los ficheros en los que aparece **exactamente** esa frase. **Este método debe utilizar la técnica INTERSECT_PHRASE que hemos visto en clase.**

La clase CompleteIndex debe funcionar correctamente al ser importada y usada de la siguiente manera:

```
>>> from CompleteIndex import CompleteIndex
>>> i = CompleteIndex('20news-18828')
>>> i.consulta_frase('either terrestrial or alien')
['20news-18828/alt.atheism/53209',
 '20news-18828/alt.atheism/53222',
 '20news-18828/alt.atheism/53218']
>>> i.consulta_frase('is more complicated')
['20news-18828/comp.os.ms-windows.misc/9966',
 '20news-18828/comp.os.ms-windows.misc/10010',
 '20news-18828/alt.atheism/53564',
 '20news-18828/soc.religion.christian/20725']
```

Importante

- Al recorrer la colección deberéis ir abriendo los ficheros para lectura con la función **open** (`https://docs.python.org/3/library/functions.html#open`). Es muy importante que indiquéis explícitamente la codificación utilizada para abrir los ficheros, porque en caso contrario será dependiente del sistema operativo y produciréis resultados diferentes para la misma colección dependiendo del sistema operativo que uséis. En esta práctica podéis abrir los ficheros con iso−8859−1 o utf−8.

- Una vez forzada una codificación, es importante no fallar al abrir los ficheros debido a que su codificación sea diferente y haya *algunos caracteres* que no sepa interpretar. Revisad la documentación de **open** en `https://docs.python.org/3/library/functions.html#open` para descubrir qué valores del parámetro errors os permiten leer ficheros de manera segura sin lanzar excepciones en cada error (quizá cambiando los caracteres ilegibles por algún otro símbolo).

- Recorrer todos los ficheros de un directorio es sencillo en Python usando funciones de la biblioteca os (`https://docs.python.org/3/library/os.html`). Mi recomendación es que uséis

os.walk() (`https://docs.python.org/3/library/os.html#os.walk`) porque ya os devuelve toda la información de todos los niveles, pero hay más funciones que os pueden gustar más y os pueden servir igual.

- La construcción de los índices debe funcionar correctamente independientemente del directorio que se pase como parámetro, ya sea una ruta absoluta o relativa. No podéis hacer ninguna suposición acerca de la organización de ficheros dentro del directorio ni del nombre de los ficheros o las carpetas.
- No tratéis de procesar la colección completa 20news−18828 desde el primer momento, ya que puede tardar mucho. Cread una colección mínima para hacer pruebas y probad la colección completa únicamente cuando estéis seguros de vuestro código.