2017/2018

Diseño del Sistema de Recuperación

RECUPERACIÓN DE INFORMACIÓN

DANIEL MARTÍN – 702858

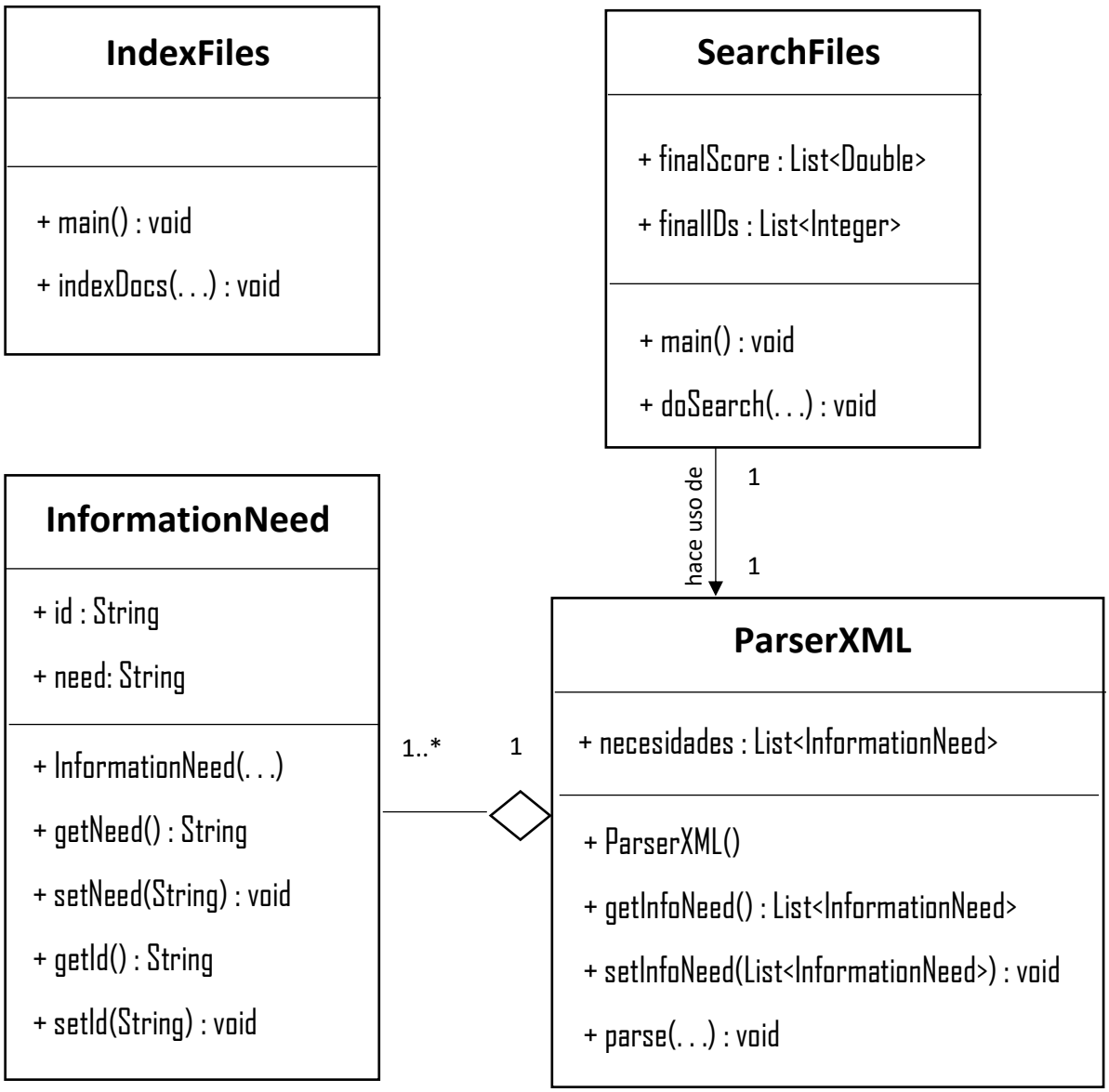
PABLO LUESIA - 698387

1.- Introducción

Este trabajo práctico de la asignatura de Recuperación de Información consiste en el diseño e implementación de un sistema de recuperación de información para posibles necesidades sobre la colección de documentos de Zaguán, repositorio de los trabajos finales de alumnos de la Universidad de Zaragoza.

En este primer informe se comentarán las decisiones de diseño primeras sobre este sistema.

2.- Diagrama de clases (Estructura básica del proyecto)



Respecto a este diagrama de clases, se incluye a continuación algunos detalles y anotaciones que se consideran relevantes para su comprensión.

- El proceso de *indexación* y de *búsqueda* son autónomos. Esto quiere decir, ambos se programan y ejecutan individualmente, pero existe una dependencia en su funcionamiento, que requiere que la indexación se haga previamente a la búsqueda.
- La clase ***InformationNeed*** representa únicamente una forma de instanciar de forma cómoda y manejable cada necesidad de información en Java. Por tanto, si dichas necesidades sufriesen modificaciones, bastaría con modificar la clase en sí.
- La clase ***ParserXML*** representa un intermediario entre el código manejable por Java y los datos XML recibidos en el proyecto. La instancia de esta clase es la que manejará la lista de necesidades, ya que podrá transformarla (*parse*) de la entrada que reciba, en los objetos Java correspondientes.

3.- Proceso de indexación. (IndexFiles)

A continuación, se indican la secuencia de acciones que realiza el proceso de indexación, así como cualquier detalle escogido que se considere deba ser mencionado en este trabajo. [1]

1) **Creación del directorio de los índices**. La primera decisión ha radicado sobre dónde crear dicho directorio, ante la posibilidad de hacerlo en disco o en RAM. Se ha escogido la opción de File System, disco, debido a su gran potencial en escalabilidad, evitando así cualquier posible límite de memoria.
2) **Elección del analizador**. En primera instancia, se parte del analizador básico de español que proporciona Lucene. Se razona esta decisión porque se cree difícil mejorar un analizador que se encuentra en una de las bibliotecas más importantes del ámbito. Por este motivo, se decide hacer uso de este analizador (en lugar de modificarlo o crear uno propio según algunas pautas del Anexo de la Práctica 1) y estudiar su funcionamiento para saber cómo llega a los resultados que obtiene.
    a. **Descripción**: Resumiendo brevemente, el analizador de Lucene cuenta con tres elementos clave en su funcionamiento:
        i. **Stopwords**: Se trata de una lista que, tanto en indexación como en búsqueda, son omitidas o ignoradas. En este listado, se podrían incluir preposiciones, conjunciones, artículos, pronombres, determinantes y otros elementos que no tienen significado/valor por sí mismos.
        ii. **Stemming**: Se trata del método que permite obtener la raíz semántica de una palabra. Se emplea para que una palabra pueda ser relacionada con cualquier otra de su familia, ampliando así el ratio de búsqueda. [2]
        iii. **Modelo de espacio vectorial**: El analizador emplea este modelo algebraico para filtrar, indexar y calcular relevancia de la diferente información que se va obteniendo, es un espacio multidimensional (haciendo operaciones sobre ángulos en dicho espacio).

---

[1] Se ha decidido explicar paso a paso el proceso en lugar de emplear diagramas de secuencia para facilitar así la adición de notas, comentarios o explicaciones respecto a las decisiones tomadas.

[2] Por ejemplo, tras aplicar la eliminación de stopwords, y stemming a “`Estamos interesados en recopilar todo tipo de información acerca de las características de la frontera entre Francia y España.`” se obtendría “`interesad recopilar tipo informacion acerc caracteristic fronter franci españ “.`

3) **Creación de los índices**. Tras haber hecho los ajustes necesarios y haber creado los elementos indispensables en el proceso, se pasa a la creación de los índices. Para ello, se hace uso de las mismas estructuras y técnicas de las que se dispuso en las prácticas de la asignatura, creando documentos (clases ***Document***, ***DocumentBuilder***…) y haciendo uso de las estructuras de *nodos* y *listas de nodos* para transformar el XML recibido en datos manejables por Java.
4) **Elección de los campos a indexar**. Para elegir que campos son indexados y guardados en los documentos, se ha estudiado la colección de documentos que se proporcionaba, y tomado la siguiente decisión.
    a. Se guardará la ruta del fichero (para tener siempre la referencia origen de los resultados), aunque esta no será analizada, por lo que se guardará como un campo de tipo ***StringField***.
    b. Se guardarán tres campos relativos al trabajo, que serán analizados, y a los cuales se les asignará un peso para su posterior valoración. (Ver Proceso de búsqueda). Como han de ser analizados, serán campos ***TextField***.
        i. <dc:title> : Contiene el título del trabajo.
        ii. <dc:subject> : Contiene las palabras clave del trabajo.
        iii. <dc:description> : Contiene la descripción extensa del trabajo.
5) **Escritura de los índices**. Finalmente, se guardarán estos índices en el directorio indicado por el usuario.

4.- Proceso de búsqueda y valoración (ranking)

Tras el proceso de indexación, y la obtención de los índices mediante Lucene, se da paso al proceso de búsqueda, para el cual se leen las necesidades de información obtenidas en clase y se ejecuta una búsqueda con valoración sobre los resultados obtenidos. Por los mismos motivos que previamente, se explica a continuación una secuencia de pasos relevantes en el proceso.

1) **Lectura del fichero de necesidades**: El primer paso parte de la lectura del fichero de necesidades y su transformación en una lista de objetos manejables por Java, a través de una clase que tratase el XML creada personalmente. Esta transformación facilita enormemente la iteración, ordenación, extracción de información, etc. [3]
2) **Obtención de resultados para cada necesidad**: Tras la obtención de esta lista, se itera sobre ella para obtener los resultados esperados del sistema. Los siguientes pasos se realizarán individualmente para cada necesidad.
3) **[ITER-1] Generación de "*queries*" para la consulta**: Siguiendo la filosofía citada previamente, se llevarán a cabo tres consultas sobre cada necesidad, una por cada uno de los tres campos indexados previamente (a saber, <dc:title>, <dc:subject> y <dc:description>), de los cuales se obtendrán tres tablas de resultados diferentes, según cada búsqueda en esos campos. El número de resultados NO está limitado.
4) **[ITER-2] Cálculo de las puntuaciones obtenidas**: Teniendo en cuenta los resultados de las 3 consultas, se pasará a generar una puntación global de los resultados, siguiendo un algoritmo que responde de esta manera:

---

[3] Se ha empleado una implementación propia del tratador de XML debido a que los miembros del grupo, implementadores del sistema, ya contaban con una clase similar, y ha agilizado y facilitado el proceso, así como afianzado el resultado.

$$score_{final}(d) = score_{titulo}(d) + 0.5 \times score_{temas}(d) + 0.25 \times score_{descripcion}(d)$$

Esta puntuación se debe a una decisión de equipo entorno a la siguiente filosofía:

- El usuario leerá primero los títulos, por lo que es lo más relevante.
- Si el título parece conformarle, mirará los tópicos o temas del trabajo, dándoles una relevancia media.
- Si estos parecen interesarle, pasará a leer la descripción. Por ello, la relevancia es la menor.

5) **[ITER-3] Ordenación de los documentos**: Tras obtener una puntuación global de cada documento, se ordena el resultado, para así poder mostrarlo, guardarlo y devolver aquella información que sea necesaria.

5.- Resultados obtenidos

Tras esto, se prueba el funcionamiento del sistema, que parece responder adecuadamente. Tomando como ejemplo la primera búsqueda, cabe destacar que los resultados que obtiene contienen en el título términos como “visión”, “computador”, “robótica”, o similares. En general, esto ocurre con todas las necesidades.

La primera conclusión que se observa es que el método de ranking elegido se plasma en los resultados, y se ve su precisión: Valora muy positivamente que el fichero contenga un título apropiado y conciso, y da valor a que los tópicos elegidos sean precisos.

Es posible que haya documentos con una descripción bastante buena, pero es arriesgado darle el mismo peso que a otros campos, puesto que en una búsqueda real en un repositorio, un usuario no va a emplear tiempo en leer algo cuyo resumen, título o enlace no ha parecido cubrir sus necesidades.