UNIVERSIDAD DE GRANADA
E.T.S.I. INFORMÁTICA Y TELECOMUNICACIÓN

Departamento de Ciencias de la Computación e Inteligencia Artificial

Gestión de Información en la Web

Guión de Prácticas

Práctica 3:
Desarrollo de un Sistema de Recuperación de Información con la biblioteca Lucene

Curso 2019-2020

Máster en Ingeniería Informática

Práctica 3

Desarrollo de un Sistema de Recuperación de Información con la biblioteca Lucene

1. Objetivos

1. Conocer las partes principales que tiene un sistema de recuperación de información y qué funcionalidad tiene cada una.
2. Implementar un sistema de recuperación de información.
3. Emplear la biblioteca Lucene para facilitar dicha implementación.

2. Trabajo a Realizar

En algún momento nos puede surgir la necesidad de desarrollar un sistema de recuperación de información. Aunque podemos partir de cero, existe un gran número de bibliotecas en diferentes lenguajes de programación que nos pueden permitir montar el software de búsqueda de una forma rápida y sencilla.

Este es el caso de Lucene, el cual es un conjunto de bibliotecas, escritas en Java (nativo), C++, C#, PHP, Python, Ruby o Perl, que nos facilitará la confección de aplicaciones para realizar la indexación de grandes volúmenes de documentos y recuperación a partir de consultas suministradas por los usuarios del sistema.

En esta práctica se construirá un sistema de recuperación de información empleando la biblioteca Lucene **(en su última versión)**, compuesto de dos programas:

1. Un **indexador**, el cual recibirá como argumentos la ruta de la colección documental a indexar, el fichero de palabras vacías a emplear y la ruta donde alojar los índices, y llevará a cabo la indexación, creando los índices oportunos y ficheros auxiliares necesarios para la recuperación. Esta aplicación se ejecutará en la línea de mandatos y no tendrá ningún componente gráfico. Este software realizará las tareas de tokenización, eliminación de palabras vacías y extracción de raíces antes de crear el índice.
2. Un **motor de búsqueda**, que al ejecutarse recibirá como argumento la ruta donde está alojado el índice de la colección y permitirá que un usuario realice una consulta de texto y obtenga el conjunto de documentos relevantes a dicha consulta. En este caso, el programa sí será gráfico. Sobre la consulta se realizarán los mismos procesos que sobre los documentos en el indexador.

3. Documentación y Ficheros a Entregar

La práctica se subirá a la plataforma docente Prado en forma de fichero comprimido (en formato zip o rar), con el nombre "practica3_apellidos_nombre.zip". Este fichero contendrá a su vez todo el código fuente, dos ficheros jar, uno para el indexador y otro para el recuperador, la colección usada y por último, un archivo pdf con la documentación.

La **documentación** de la práctica será un fichero *pdf* que deberá incluir, al menos, el siguiente contenido:

a) Portada con el número y título de la práctica, el curso académico y el nombre, DNI y dirección e-mail del alumno.

b) La documentación de la práctica contendrá una breve descripción de la misma, explicando cómo se ha hecho, los elementos innovadores, una descripción de la colección (tipo, estructura y contenido) y un pequeño manual de usuario.

c) Referencias bibliográficas u otro tipo de material distinto del proporcionado en la asignatura que se haya consultado para realizar la práctica (en caso de haberlo hecho).

La fecha de entrega será el 10 de marzo, como máximo justo antes de clase (15:30h). En esa clase de prácticas procederéis a mostrar vuestras aplicaciones al resto de compañeros y a explicar cómo la habéis desarrollado.

Aunque lo esencial es el contenido, también debe cuidarse la presentación y la redacción.

4. Evaluación

La evaluación de esta práctica tendrá en cuenta la calidad de los siguientes aspectos:

- Adecuación a los requerimientos de las aplicaciones propuestas.
- Correcto uso de las bibliotecas de Lucene.
- Inclusión de aspectos innovadores no vistos en clase.
- Documentación de la práctica.

5. Logística

Colección documental

La colección documental que emplearemos será libre, en inglés o español, pero al menos deberá contener 100 documentos.

Listas de palabras vacías para el inglés y el español que se pueden emplear podréis encontrarlas en la sección de la *Practica 3* de Prado.

Material de referencia sobre Lucene

- https://lucene.apache.org
- Lucene in Action.
- http://en.wikipedia.org/wiki/Lucene

Apuntes sobre Lucene

Al descargarnos del sitio web de Lucene la aplicación, nos encontramos tres ficheros que tenemos que incluirlos en el classpath de las aplicaciones que los usen (XXX es la versión de Lucene):

- lucene-core-XXX.jar
- lucene-analyzer-XXX.jar
- lucene-queryparser-XXX.jar
  Para usar Lucene, una aplicación de indexación debería:
- Crear Documents añadiéndole Fields.
- Crear IndexWriter y añadir documentos con addDocument(); (eliminando palabras vacías y haciendo stemming, en su caso).

  Un código simplificado de la indexación sería el siguiente:

```
Analyzer analyzer = new   StandardAnalyzer(Version.LUCENE_43);

// Store the index in memory:

Directory directory = new RAMDirectory();

//  en disco ... //Directory directory = FSDirectory.open("/tmp/testindex");

IndexWriterConfig config = new  IndexWriterConfig(Version.LUCENE_43, analyzer);

IndexWriter iwriter = new IndexWriter(directory, config);

Document doc = new Document();

String text = "This is the TEXT to be indexed (and searched)!! .";

doc.add(new Field("name", text, TextField.TYPE_STORED));

iwriter.addDocument(doc);

iwriter.close();
```

Lucene ofrece una amplia variedad de analizadores, que son los encargados de realizar toda la gestión de tokens del texto original. En el caso de esta práctica, tendréis que emplear el SpanishAnalyzer, que realiza stemming del español (https://lucene.apache.org/core/4_3_0/analyzers-common/org/apache/lucene/analysis/es/SpanishAnalyzer.html).

Una vez tengamos los índices creados, podemos emplear la herramienta Luke (https://code.google.com/p/luke) para acceder a un índice existente, poder visualizarlo y comprobar que el proceso se ha realizado correctamente.

Y en el caso de una aplicación de consulta:

- Llamar a QueryParser.parse() para construir una consulta desde una cadena de caracteres.
- Crear un IndexSearcher y pasarle la consulta con el método search();

El código simplificado para la consulta y la recuperación sería el siguiente:

```
DirectoryReader ireader = DirectoryReader.open(directory);

IndexSearcher isearcher = new IndexSearcher(ireader);

// Parse a simple query that searches for "text":

 QueryParser parser = new QueryParser(Version.LUCENE_43, "fieldToSearch", analyzer);

 Query query = parser.parse("text");

 ScoreDoc[] hits = isearcher.search(query, null, 1000).scoreDocs;

 // Iterate through the results:

 for (int i = 0; i < hits.length; i++) {

     Document hitDoc = isearcher.doc(hits[i].doc);

      System.out.println("salida "+hitDoc.get("name").toString());

      System.out.println("salida "+hitDoc.toString());

     }

ireader.close();

directory.close();
```

Las clases que se han empleado en este ejemplo son:

```
import org.apache.lucene.analysis.Analyzer;

import org.apache.lucene.analysis.standard.StandardAnalyzer;

import org.apache.lucene.store.Directory;

import org.apache.lucene.store.RAMDirectory;

import org.apache.lucene.index.DirectoryReader;

import org.apache.lucene.search.IndexSearcher;

import org.apache.lucene.queryparser.classic.QueryParser;

import org.apache.lucene.queryparser.classic.ParseException;
```

```
import org.apache.lucene.search.Query;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.document.TextField;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.util.Version;
```