UNIVERSIDAD DE GRANADA

Máster Universitario en Ingeniería Informática

Gestión de Información en la Web

Desarrollo de un Sistema de Recuperación de Información con Lucene

Alumno
Marvin Matías Agüero Torales
DNI
4423998
E-mail
maguero@correo.ugr.es

2016-2017

Contenido

Introducción

En este trabajo se construirá un sistema de recuperación de información, se utilizará Lucene, el cual es un conjunto de bibliotecas, escritas en Java, que nos pueden permitir montar el software de búsqueda de una forma rápida y sencilla: nos facilitará la confección de aplicaciones para realizar la indexación de grandes volúmenes de documentos y la recuperación a partir de consultas suministradas por los usuarios del sistema.

Objetivos

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

Metodología y herramientas

Para llevar acabo el desarrollo del este trabajo, se ha empleado Java, que por ser el lenguaje de programación nativo de Lucene, cuenta con mucha más documentación que sus pares (C#, PHP, Python u otros). El IDE (Integrated Development Environment) utilizado es Eclipse Neon.2, y como la interfaz es Web, se utilizará además el servidor Apache Tomcat 9 para desplegar las consultas realizadas, a su vez se hizo uso de Maven 3.3.9.

Se utilizará Lucene en su versión 4.3[1].

Desarrollo

Para este trabajo se desarrolló un indexador sobre la colección documental de noticias en español de la agencia EFE, publicadas en los años 1994[2], y un motor de búsqueda simple (búsquedas de palabras) sobre una plataforma Web (que nos recuerda mucho al de Google). En ambos casos se utilizaron los códigos base, proveídos en el guión del trabajo, solo que utilizando el *SpanishAnalyzer* en vez del estándar, y los añadidos para manejar los documentos en formato SGML.

Indexador

La aplicación del indexador, emplea una clase Java, que recibe como argumentos la ruta de la colección documental a indexar primeramente, además del archivo de palabras vacías a emplear y la ruta donde alojar los índices a crear. Para llevar a cabo la indexación, se podrá ejecutarla desde la línea de mandatos, creando los índices oportunos y archivos auxiliares necesarios para la recuperación posterior.

Se utiliza además otra clase Java para la manipulación de los campos a indexar, en este caso los correspondientes al título y el texto de las noticias.

Motor de búsqueda

El motor de búsqueda, al ejecutarse recibe como argumento la ruta donde está alojado el índice de la colección y a su vez permite al usuario realizar una consulta de texto simple, para obtener el

1 http://archive.apache.org/dist/lucene/java/4.3.0/
2 https://consigna.ugr.es/g/NVUThaH1Q69zSopM/efe94.tgz

conjunto de documentos relevantes a dicha consulta, aplicando los mismos procesos que sobre los documentos en el indexador.

Para este trabajo, se optó por utilizar las tecnologías provistas por Java EE (Servlet[3] y Java Server Pages o JSP[4]) para montar el buscador en una plataforma Web *minimalista* utilizando BootStrap 4[5]. Básicamente se invoca una clase Java (origen) con la lógica en Lucene, desde un Servlet que actúa de *middleware*, intercambiando objetos con páginas HTML5 (para la entrada de datos) y JSP (para la interacción: entrada de datos y salida de coincidencias).

Consideraciones

Como todo desarrollo en donde se entrelazan varios elementos, se encontraron algunos problemas, por ejemplo para leer los archivos SGML, no tenían el formato estándar, por ello el contenido de cada archivo se añadió bajo la etiqueta <SGML></SGML> y se tuvieron que remediar algunos errores en la lectura.

Otro inconveniente se dio al mostrar los datos en la interfaz (codificación), en mi caso sobre la plataforma Web, para remediarlo se tomaron los datos en ISO-8859-1, se indexaron en UTF-8 y se mostraron en ISO-8859-1.

Al emplear una plataforma Web tuve algunos percances a la hora de tomar la carpeta de indices para la búsqueda, primeramente cree un proyecto del tipo *Web Dynamic Content*, lo convertí a un proyecto del tipo Maven (para utilizarlo lo debemos tener instalado en el equipo e integrado a Eclipse). Seguidamente asigné las rutas de creación y lectura de indices dentro de la carpeta *WebContent,* a manera de tenerla disponibles para ambas aplicaciones: indexador, se accede sin inconvenientes, y motor de búsqueda, en este caso, debemos tomar el contexto de la aplicación Web y añadir la ruta a la misma.

Manual de usuario

Indexador

Para llevar a cabo la indexación de documentos, se debe ejecutar la clase Index desde el IDE o un .jar generado a partir del mismo. Los argumentos de ejecución se encuentran dentro del código, las rutas relativas a directorios y archivos básicamente, es por ello que para cambiar algún dato habría que recompilarlo. Las rutas son las siguientes:

- WebContent/data/indexes/ → directorio de índices
- WebContent/data/efe → colección documental
- WebContent/data/palabras_vacias_utf8.txt → palabras vacías en español

Para ejecutar el .jar se debe ejecutar:

```
java -jar nombreIndiceJar.jar
```

No se incluyó la escritura de logs del proceso (por los costos de ejecución que esto acarrea), pero si emite en pantalla un estado del proceso mínimo, a manera de no ralentizar el mismo, con el tiempo total empleado para el proceso.

3 http://www.oracle.com/technetwork/java/javaee/servlet/index.html
4 http://www.oracle.com/technetwork/java/javaee/jsp/index.html
5 https://v4-alpha.getbootstrap.com

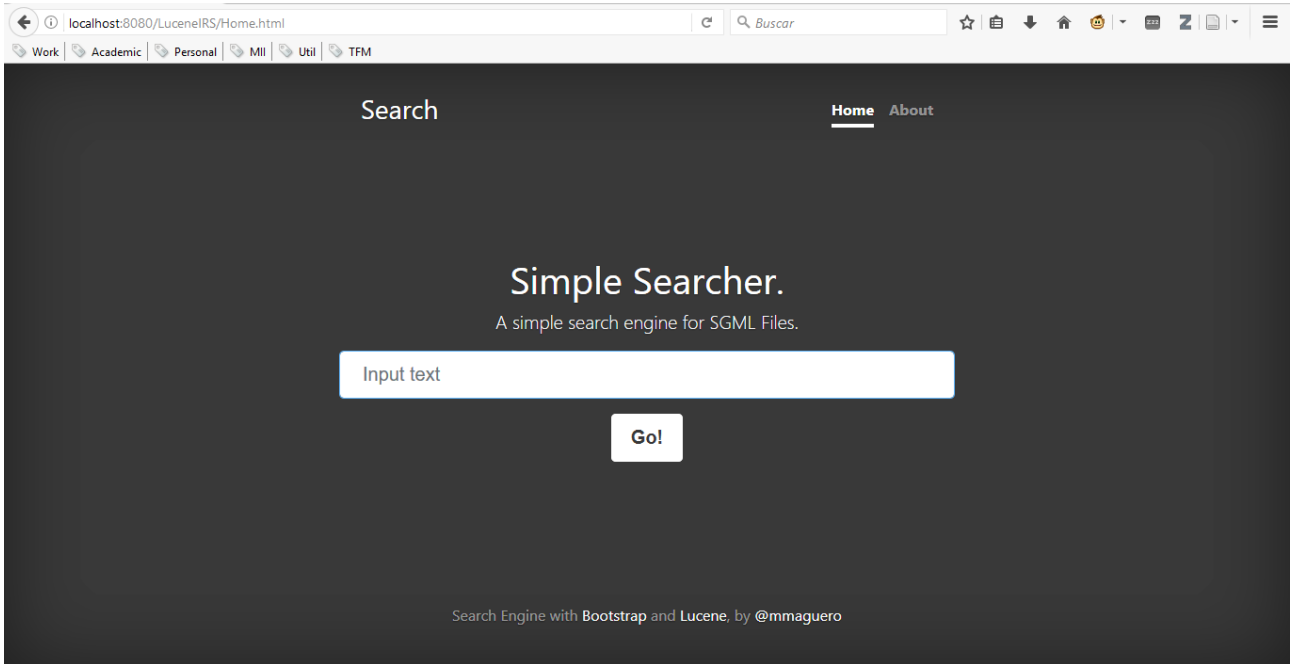
```
C:\Users\adm\Desktop\Demo>java -jar index.jar
Create the analyzer
Read news from a directory
Create the index writery
Delete the lastest index
Iterate each document
Total min. 4.326333333333333
```

Motor de Búsqueda

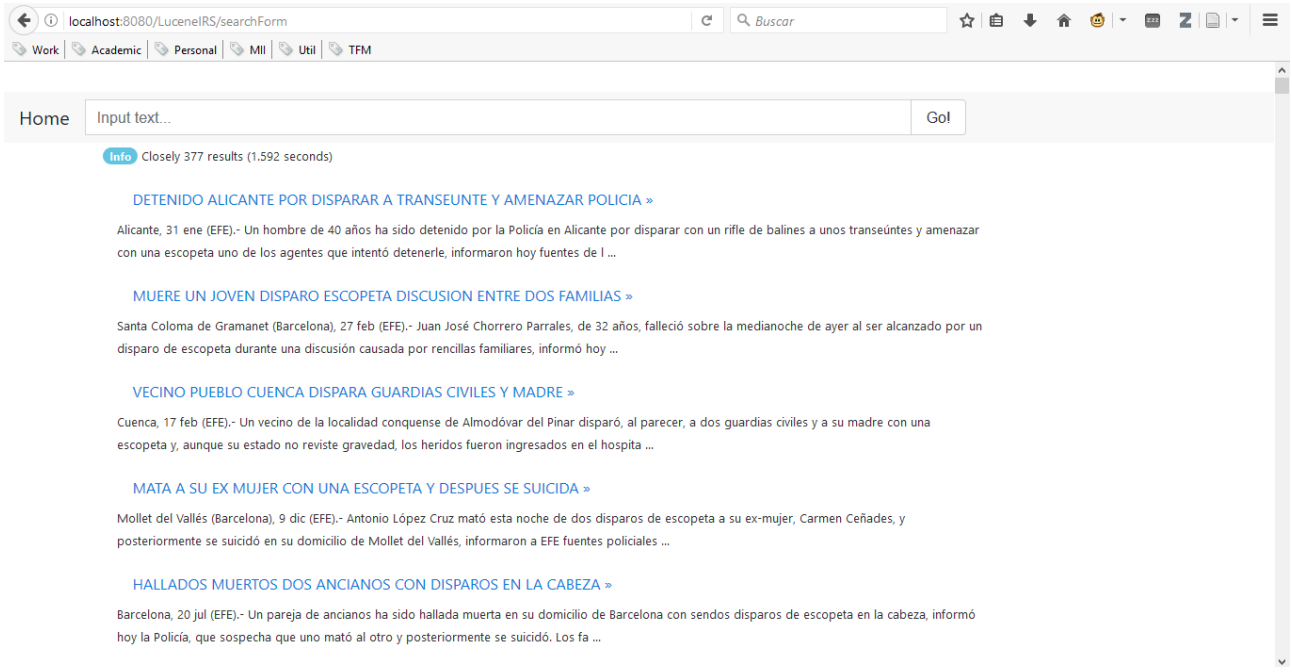
Para lanzar el motor de búsqueda hay que desplegar la aplicación Web que ejecutará la interfaz del usuario con el motor de búsqueda en un servidor Apache Tomcat (ideal, versión 9 o superior), o a través del proyecto con un IDE como Eclipse, previamente configurado dicho servidor. Cabe destacar, que para desplegar un .war se debe copiarlo en la carpeta *webapps* del directorio de instalación de Tomcat.

El motor usa las mismas rutas que el indexador, que para cambiar la ruta habría que recompilarlo también. Es decir, que los indices creados en la carpeta WebContent (data/indexes), junto con el archivo de palabras vacías (data/palabras_vacias_utf8.txt) en el paso anterior, debe ser copiado enteramente a la misma altura que los archivos HTML o JSP en la carpeta LuceneIRS creada al desplegar el .war.

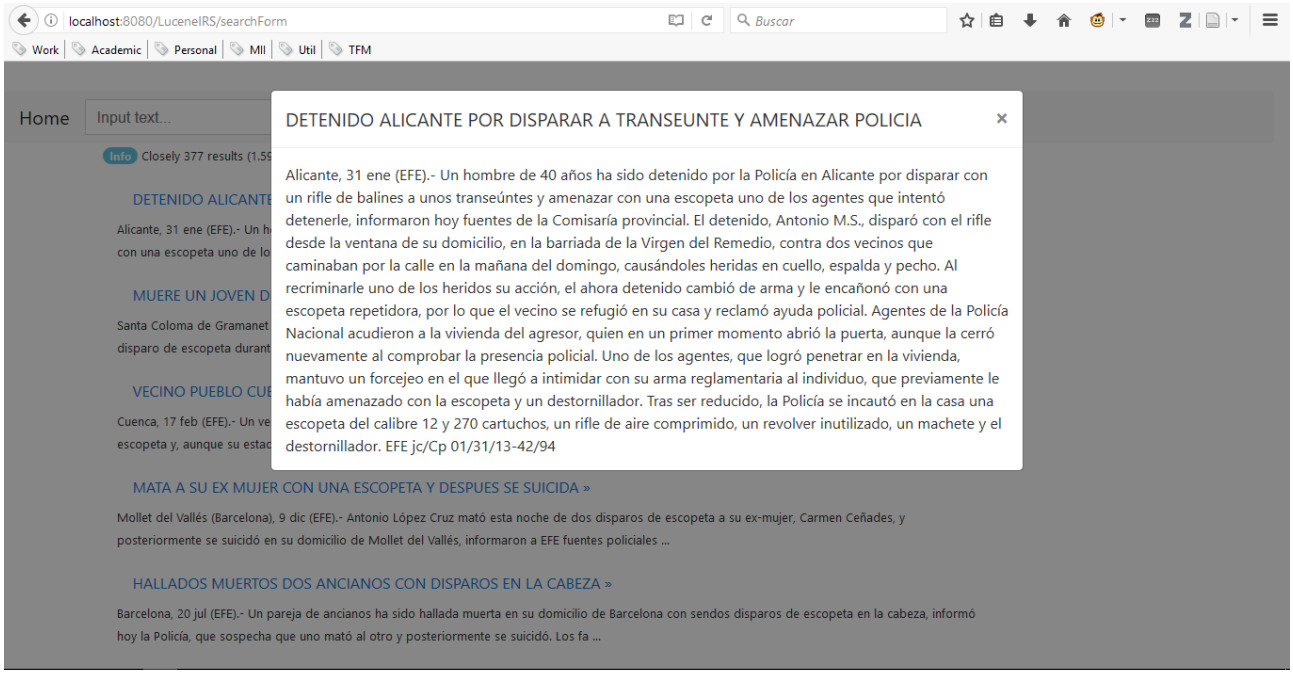
Una vez desplegado, se puede acceder desde el navegador a la siguiente dirección: http://localhost:8080/LuceneIRS/Home.html.



Cada consulta se toma desde los índices creados, para ello hay que ingresar el texto a buscar y los resultados se desplegarán en una pantalla nueva, con el número de resultados y el tiempo aproximado que ha tardado en encontrarlos.



Ya dentro de la pantalla de resultados, cuando se haga clic sobre el título de cualquiera de los resultados, se desplegará una pantalla modal con el texto completo del elemento.



Conclusiones

Las imágenes que vimos, son las relacionadas a la búsqueda de la palabra “escopeta”. Como había mencionado la interfaz de este buscador, nos recuerda mucho a Google, por lo que muy intuitiva para el usuario, aunque se hecho de menos la paginación de resultados (o las búsquedas avanzadas), opciones que podrían ser implementadas en un trabajo futuro, puesto que implican un trabajo mayor de backend.

De esta manera, se lograr montar un Sistema de Recuperación de Información a partir de Lucene, una herramienta potente, que para aplicaciones no muy complejas, resulta útil y práctico. No obstante, si se quisiera, por ejemplo montar una solución de búsquedas de manera distribuida y de

gran envergadura, podríamos usar Solr[6], incluso podríamos reutilizar como base este trabajo, puesto que esta herramienta está basada en Lucene.

Bibliografía consultada

Apache Lucene - Getting Started Guide (2013). Recuperado el 29 de Abril de 2017 de https://lucene.apache.org/core/2_9_4/gettingstarted.html
Example – Bootstrap (2017). Recuperado el 01 de Mayo de 2017 de https://v4-alpha.getbootstrap.com/examples/
GitHub - apache/lucene-solr: Mirror of Apache Lucene + Solr (2017). Recuperado el 30 de Abril de 2017 de https://github.com/apache/lucene-solr
How do I get the location of my web application context in the file system? (2014). Recuperado el 01 de Mayo de 2017 de http://www.avajava.com/tutorials/lessons/how-do-i-get-the-location-of-my-web-application-context-in-the-file-system.html
How do I use Lucene to index and search text files? (2014). Recuperado el 28 de Abril de 2017 de http://www.avajava.com/tutorials/lessons/how-do-i-use-lucene-to-index-and-search-text-files.html
Pasar datos entre JSPs y Servlets. Page, Request, Session y Application scope (2016). Recuperado el 01 de Mayo de 2017 de http://chuwiki.chuidiang.org/index.php?title=Pasar_datos_entre_JSPs_y_Servlets._Page,_Request,_Session_y_Application_scope

Anexos

Se adjunta el proyecto Maven creado en Eclipse con los directorios de datos y de índices vacíos, debido a su gran tamaño. Un .jar, para el indexador y un .war para el recuperador.

Cabe resaltar que para que funcione todo el Sistema de Recuperación de Información, se debe respetar las rutas (o modificarlas y volver a compilar).

La estructura es la siguiente:

- index.jar → ejecutable, indexador
- LuceneIRS.war → desplegable, motor de búsqueda
- WebContent/data/indexes/ → directorio de índices
- WebContent/data/efe → colección documental
- WebContent/data/palabras_vacias_utf8.txt → palabras vacías en español

Disponible en https://github.com/mmaguero/Lucene-IRS

6 http://lucene.apache.org/solr/