PRÁCTICA III:

...

Implementación de un Sistema de Recuperación de Información utilizando Lucene

...

Diseño

16 de octubre de 2019

Índice

1. Objetivo

El objetivo final de esta práctica es que el alumno realice un diseño preliminar de la futura aplicación de búsqueda. Esta práctica representa el primer paso hacia la práctica final de la asignatura (que en su totalidad es un 80 % de la nota final de prácticas). La práctica final se dividirá en 4 bloques en total, que se irán entregando conforme avance la asignatura.

En cualquier aplicación de búsqueda podemos distinguir los siguientes pasos, que detallaremos a continuación:

1. Análisis de requisitos y adquisición de datos.
2. Procesamiento de los datos, que ya hemos considerado en parte en prácticas anteriores.
3. Indexación y almacenamiento de los mismos.
4. Búsqueda sobre el índice y presentación de los resultados.

1.1. Fecha de entrega y defensa de la práctica

La fecha de entrega de la práctica completa (diseño, indexación, búsqueda y facetas), junto con la documentación asociada será el Viernes 29 de Noviembre de 2019. La defensa se realizará los siguientes días de clase.

2. Se pide

Para esta práctica concreta se pide lo siguiente:

- Diseño preliminar de la interfaz de búsqueda. Nos debe indicar qué tipos de búsqueda son los que queremos hacer, los campos que queremos indexar y el tratamiento (procesamiento) que le queremos dar.
- Diseñar una clase que sea capaz de extraer todas las películas del fichero csv y para cada una de ellas extraer todos y cada uno de los campos anteriormente mencionados.

3. Diseñando el Sistema de Recuperación de Información

A la hora de diseñar cualquier aplicación de búsqueda, el primer paso es analizar qué tipo de información se va a buscar y cómo se realizan las búsquedas por parte de un usuario. En esta práctica el objetivo es crear un buscador sobre el conjunto de datos *Wikipedia Movie Plots* que en total contiene 34886 descripciones sobre películas extraídas de Wikipedia. Los datos vienen todos en un único fichero en formato csv (con los campos separados por comas y en caso de necesidad, si campo contiene el separador coma, sus elementos aparecen entrecomillados) que podeis encontrar en la web de la asignatura así como en Kaggle (`https://www.kaggle.com/jrobischon/wikipedia-movie-plots`)

El la base de datos podemos encontrar los siguientes campos

- Release Year: Año de lanzamiento: año en que se lanzó la película
- Title: Título de la película
- Origin: Origen de la película (es decir, estadounidense, Bollywood, tamil, etc.)
- Director: Director(es)
- Cast: Actor principal y actrices
- Genre: Género(s) de la película
- Wiki Page: URL de la página de Wikipedia desde la cual se raspó la descripción de la trama
- Plot: descripción en texto de la trama de la película

3.1. Análisis de requisitos y adquisición de datos.

En esta fase de análisis de requisitos se considerarán las necesidades de los hipotéticos usuarios finales de la aplicación para determinar qué objetivos se deben cubrir. En este sentido se debe proporcionar la documentación de la especificación

completa de lo que debe hacer el sistema sin entrar en detalles internos. Para ello, será necesario ubicarnos sobre el problema concreto, en este caso el acceso a la información en Wikipedia Movie Plots.

Como vemos, en una línea de cada fichero CSV se tiene toda la información que necesitamos sobre una película.

Una vez que conocemos los datos, podremos imaginar qué tipos de consultas se podrían realizar por un usuario así como la/las respuesta que daría el sistema, en la cual se devuelven los elementos ordenados por relevancia, facilitando las labores del usuario.

Lucene permite hacer consultas por campos, por lo que será necesario identificar los mismos. Los campos concretos dependerá de la aplicación concreta sobre la que estemos trabajando, pero como hemos visto un mismo campo podrá utilizarse para dos funciones distintas, por ejemplo como texto sin tokenizar y texto tokenizado.

3.2. Diseño de la interfaz de búsqueda

El primer paso, debe ser la identificación de los campos que serán recuperables. Obviamente, para este proceso es conveniente el diseñar previamente la propia interfaz de búsqueda donde se especifique qué y cómo se desea buscar.

En nuestra aplicación deberemos identificar al menos los siguientes tipos de campos sobre los documentos de entrada:

- StringField: Texto simple que se considera literalmente (no se tokeniza), útil para la búsqueda por facetas, filtrado de consultas y también para la presentación de resultados en la interfaz de búsqueda.
- TextField: Secuencia de términos que es procesada para la indexación, esto es, convertida a minúscula, tokenizada, stemizada, etc.
- Numérico, como por ejemplo fecha
- Facetas (Categorías) (como trabajar con facetas se verá posteriormente)

Además de una consulta por texto libre, en la aplicación se deberá poder realizar como mínimo una consulta booleana que involucre a los operadores lógicos

OR, AND o NOT en la misma. Además deberemos proporcionar algún tipo de consulta avanzada como por ejemplo las consultas por proximidad, así como permitir presentar la información utilizando distintos criterios de ordenación (esto es, además de presentar los elementos ordenados por relevancia, debemos de poder presentarlo utilizando un orden distinto).

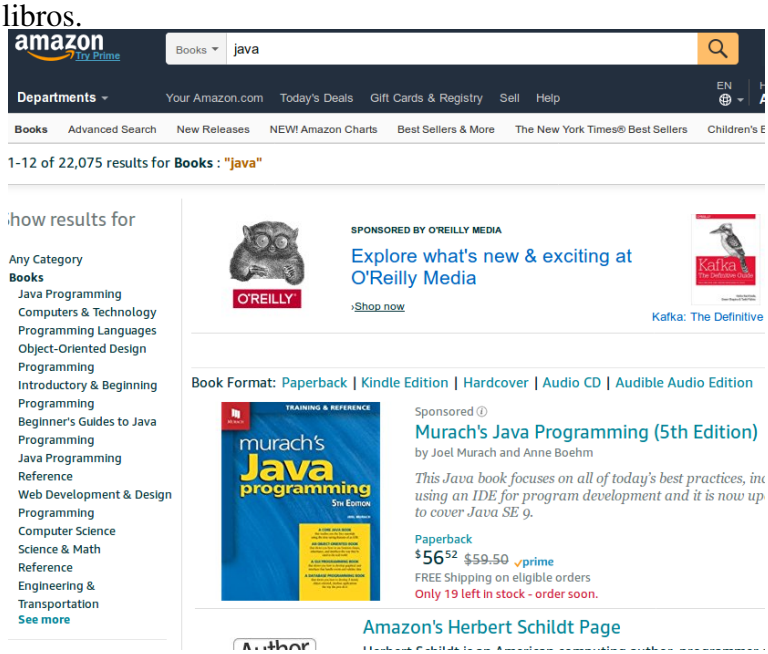
3.3. Búsqueda por Facetas

Una búsqueda por facetas nos permite acceder a la información refinando la búsqueda de acuerdo a una clasificación por categorías, filtrando los datos teniendo en cuenta las categorías a las que pertenecen. Para ello, es necesario que cada documento pueda ser clasificado a lo largo de múltiples dimensiones (llamadas facetas) como por ejemplo el autor, el idioma o en un sitio de comercio electrónica cada una de las posibles categorías bajo las que podemos clasificar un producto (marca, modelo, características, etc.).

Un atractivo de la mezcla de la búsqueda con el uso de la navegación por facetas es que, ante una consulta, podemos mostrar el número de elementos recuperados en cada una de las categorías. Así, por ejemplo, podemos saber cuantos trabajos, de entre los relevantes a la consulta, han sido publicados en el año `2015` o el `2016`, o cuántos de ellos han sido escritos por un determinado autor. Además, cuando el usuario selecciona una de ellas podemos restringir la búsqueda (drill down) entre los documentos que pertenecen a dicha categoría. Esta información hace fácil la búsqueda de los elementos de interés, ya que el usuario puede navegar fácilmente por los resultados, facilitando las siguientes interacciones con el sistema para refinar la búsqueda.

Esta peculiaridad ha hecho que la búsqueda por facetas sea muy común en sitios de comercio electrónico, como por ejemplo Amazon. En la Figura 1 podemos ver cómo ante la consulta “Java” encontramos un total de 22075 libros en el portal de ventas Amazon. A la izquierda de la misma encontramos un frame en el que se permite mostrar los resultados por categorías (aunque Amazon, por motivos internos, ha decidido no mostrar cuántos libros hay en cada una de las categorías). Entre las categorías que considera Amazon encontramos el tipo de libro, lenguaje, autores, formato, etc.

Figura 1: Búsqueda de libros en Amazon, consulta: “Java”. Se encuentran un total de 22.075 libros.



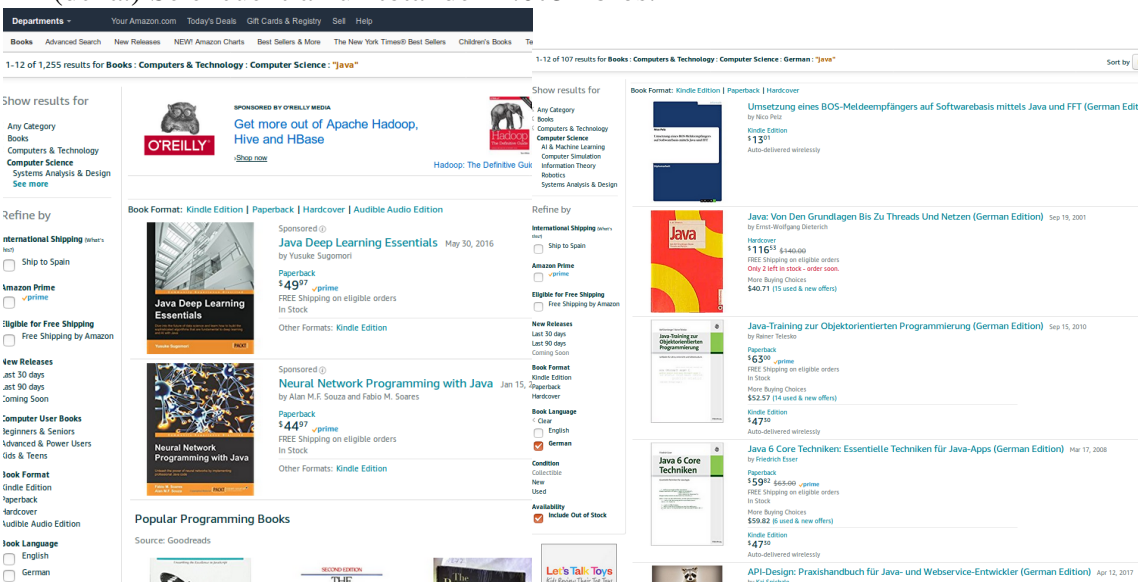
Así, podemos centrar la búsqueda dentro de la categoría Computer-Science (imagen a la izquierda de la Figura 2), encontrando un total de 1255 libros y dentro de ella, podemos de nuevo restringirnos a los libros que han sido editados en Alemán (imagen a la derecha de la Figura 2), encontrando un total de 107 libros.

3.4. Selección del analizador

El proceso de análisis nos permite identificar qué elementos (términos) serán utilizados en la búsqueda y cuales no (por ejemplo mediante el uso de palabras vacías)

Las operaciones que ejecuta un analizador incluyen: extracción de tokens, supresión de signos de puntuación, acentos o palabras comunes, conversión a minúsculas (normalización), stemización. Para ello, debemos de seleccionar de entre los tipos que tiene Lucene implementados, el que se considere adecuado para nuestra aplicación, justificando nuestra decisión. Este es un criterio importante para poder alcanzar los resultados óptimos en la búsqueda. En esta práctica será necesario

Figura 2: Consulta: “Java”, restringimos la búsqueda a los libros que se encuadran dentro de la categoría computer-science (izq.) o computer-science ->Alemán (dcha.) Se encuentran un total de 22.075 libros.



utilizar un analizador distinto al por defecto para algunos de los posibles campos a indexar, para ello podemos utilizar un `PerFieldAnalyzerWrapper` como indica el siguiente ejemplo.

```
1 // map field-name to analyzer
2 Map<String, Analyzer> analyzerPerField = new HashMap<String,
      Analyzer>();
3 analyzerPerField.put("uncampo", new StandardAnalyzer());
4 analyzerPerField.put("otrocampo", new EnglishAnalyzer());
5
6 // create a per-field analyzer wrapper using the Whitespace as
      .. default analyzer ;)
7 PerFieldAnalyzerWrapper analyzer = new PerFieldAnalyzerWrapper(
      new WhitespaceAnalyzer(), analyzerPerField);
```

3.5. Uso de Facetas

Se deberá realizar la búsqueda por facetas. Para ello deberá identificar los campos por los que podrá clasificar los documentos. Así, como resultado de la

búsqueda, podremos tener los resultados agrupados por categorías, permitiendo al usuario bucear por ellas en busca de la información de su interés. La documentación más extensa sobre esta parte la veremos posteriormente.