PRÁCTICA III:

...

Implementación de un Sistema de Recuperación de Información utilizando Lucene

...

Problema

18 de octubre de 2023

Índice

1. Objetivo

El objetivo final es que el alumno comprenda todos los procesos que intervienen en el diseño de un Sistema de Recuperación de Información y cómo puede ser implementado utilizando la biblioteca Lucene.

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). El resto de prácticas se irán entregando conforme avance la asignatura.

2. Conjunto de Datos

Para ello, en este curso utilizaremos la base de datos "**Los Simpsons**" que nos podremos descargar de Prado. Esta base de datos contiene contenido de unos 564 capítulos de la serie (los datos originales han sido descargados de Kaggle, `https://www.kaggle.com/datasets/prashant111/the-simpsons-dataset`, aunque se les ha tratado para ajustarlos a las necesidades de esta práctica). El formato en el que se encuentran los datos es CSV.

Los Simpson es un programa de televisión animado estadounidense. El conjunto de datos contiene los personajes, ubicaciones y episodios de la mayoría, si no de todos, los programas de Los Simpson. Este conjunto de datos tiene muchas características que lo hacen atractivo para este curso:

- es lo suficientemente grande como para proporcionar algunos datos interesantes, pero lo suficientemente pequeño como para poder ejecutarlo en su estación de trabajo local.
- los datos tienen relaciones que se pueden modelar en nuestro sistema
- no es demasiado complejo, lo que distrae la atención del curso
- También hay datos basura allí, lo cual es extremadamente valioso ya que será necesario no indexarlos y agregar carga innecesaria al índice.

En esta curso trabajaremos con dos versiones distintas del conjunto de datos: La primera incluye información sobre el guión detallado de cada uno de los capítulos,

que llamaremos **Guiones** y la segunda tiene información adicional sobre cada capítulo, que llamaremos **Allcapítulos**.

Los grupos deben trabajar con las dos versiones del conjunto de datos, mientras que quién trabaje de forma individual debe trabajar sólo con AllCapítulos.

Pasamos a describir los dos conjunto de datos:

2.1. Guiones

Tenemos un fichero por capítulo, donde en cada uno aparece en una línea del fichero CSV los siguientes datos

- ID: Primer campo que no tiene utilidad
- episode_id: El número de episodio
- number: Entero que representa la posiciónn (orden) del diálogo en el episodio
- timestamp_in_ms: tiempo desde el inicio del capítulo
- raw_character_text: Personaje que habla
- raw_location_text: Ubicación del discurso
- spoken_words: El diálogo en si.

Por ejemplo, mostramos varias líneas del fichero

```
148763,1,2,8000,Marge Simpson,Car,"Ooo, careful, Homer."
148764,1,3,10000,Homer Simpson,Car,There's no time to be careful.
148765,1,4,10000,Homer Simpson,Car,We're late.
...
148770,1,9,34000,Homer Simpson,Auditorium,Pardon my galoshes.
148771,1,10,44000,Seymour Skinner,Auditorium,"Wasn't that wonderful?
            And now, ""Santas of Many Lands,"" as presented
            by the entire second grade class."
```

Como hemos comentado, es un fichero CSV donde los datos se encuentran separados por comas, en el caso de que exista una coma en el texto se usa dobles comillas como delimitador, y si existen comillas como en `"Santas of Many Lands,"` se utilizan dos dobles comillas. En algunos casos podemos encontrar campos vacíos que se deben tratar de forma apropiada.

2.2. AllCapitulos

En este caso, cada fichero CSV contiene información extra asociada a cada capítulo. Cada fichero cuenta de dos líneas, la cabecera y el contenido en si. En concreto, los datos que nos encontramos son:

- ID (código que no es de utilidad para nuestro propósito)
- episode_id número del episodio
- spoken_words: Todos los diálogos de los que constan el episodio
- raw_character_text: Listado de todos los personajes que intervienen en el capítulo, en orden alfabético.
- imdb_rating: La valoración media del capítulo en IMDB (Internet Movie Data Base)
- imdb_votes: Número de votos recibidos
- number_in_season: número del episodio en la temporada (igual que episode_id)
- original_air_date: Fecha de emisión original
- original_air_year: Año de emisión
- season: Temporada del capítulo
- title: Títuo del capítulo
- us_viewers_in_millions: Número de espectadores en US (en millones)

- views: Espectadores totales

Pasamos a mostrar un extracto de uno de los ficheros de datosm

```
0,1,"Ooo, careful, Homer. There's no time to be careful.
We're late. .... Like Attila the Hun!",
"Dog's Owner, Elf #1, ..., Students",
8.2, 3734.0, 1,1989-12-17, 1989, 1,
Simpsons Roasting on an Open Fire,
26.7, 171408.0
```

Nuestro objetivo será construir un software de búsqueda que nos permite hacer consultas sobre la serie por contenido (descripción de la misma) y que nos devuelva la información que pueda ser más relevantes a nuestros intereses, posiblemente categorizadas. El sistema en si lo realizaremos a lo largo de las tres siguientes prácticas (una dedicada a la indexación, otra a la recuperación y otra final al uso de facetas), sin embargo el propósito de esta práctica es poder leer los datos y decidir qué tipo de tratamiento (y por tanto analizador) vamos a considerar para cada uno de los campos en nuestros ficheros de datos.

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

1. Análisis de requisitos y adquisición de datos.
2. Pre-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.

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. Por ejemplo, si estamos interesados en capítulos de Los Simpsons que estén relacionados con la navidad podríamos pensar en una consulta con los términos `Christmas Carol  Santa`.

En este caso, tenemos dos alternativas posibles para poder realizar nuestras búsquedas, dependiendo del conjunto de datos sobre el que se realice.

En la primera, si utilizamos el conjunto de datos por **Guiones** podemos considerar que cada línea/diálogo es un documento en sí, y por tanto, nuestro criterio de búsqueda estará basado en la búsqueda dentro de los diálogos

Así, para la consulta anterior, una posible salida del sistema será la siguiente:

1. Cap 309 línea 21: Hey Hommer, writing a new Christmas Carol? (Ned Flanders)

2. Cap 320 línea 93: And now back to "Mr. McGrew's Christmas Carol"(Cartoon Announcer)

3. Cap 472 línea 204: Just as soon as we sing one Christmas Carol (Montgomery Burns)

4. Cap 1 línea 328: THEN ONE FOGGY CHRISTMAST EVE / SANTA CAME TO .. (Lisa Simpson)

5. ...

7. Cap 399 línea 72: Lookin' good, Carol. (DOLPH)

9. Cap 414 línea 100: Got any Joyce Carol Oates? (Prisoner Lisa)

De igual forma, si consideramos el conjunto de datos **AllCapitulos**, donde todos los diálogos de un capítulo forman un único documento nos encontramos que la salida del sistema puede ser del tipo

1. Cap 320: Tis the Fifteenth Season (Temporada 15, IMDB rating 7.2)

2. Cap 472 The Fight Before Christmas (Temporada 22, IMDB rating 6.9)

3. Cap 1: Simpsons Roasting on an Open Fire (Temporada 1, IMDB rating 8.2)

4. Cap 309: Dude, where is my ranch (Temporada 14,IMDF rating 7)
5. Cap 365: Simpsons Christmas Stories (Temporada 17, IMDB rating 6.9)
6. Cap 341: Midnight Rx (Temporada 16, IMDB rating 7.3)
7. Cap 188: Miracle on Evergreen Terrace (Temporada 9, IMDB rating 7.6)
8. Cap 387: Kill Gil, Volumes I & II (Temporada 18, IMDB rating 6.4)
9. Cap 275: She of little Faith (Temporada 13, IMDB rating 7.1
10. ...

Como podemos ver, no todos los documentos recuperados (líneas de guión o capítulos) tienen que estar relacionados con la temática de la consulta. Esto es normal cuando consideramos un sistema de recuperación de información.

En PRADO se puede encontrar los ficheros .csv (con los campos separados por comas). Podéis descargarlos para empezar a familiarizaros con los mismos y también consultar la página de Kaggle para haceros una idea sobre la distribución de los datos.

Para poder trabajar con los mismos deberemos ser capaz de extraer los atributos del fichero .csv. Con este fin se recomienda el uso de una biblioteca específica para la lectura de este tipo de información como puede ser `openCSV` (o cualquier otra que se conozca) `https://mkyong.com/java/how-to-read-and-parse-csv-file-`
Si usamos openCSV deberemos considerar las bibliotecas opencsv-x.x.x.jar y commons-lang3-y.y.y.jar.

4. Diseñando nuestro sistema

Una vez que conocemos los datos de nuestra práctica, 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. En los ejemplos anteriores hemos visto que ante una consulta el sistema puede devuelvor el capítulo, línea del fichero, personaje, temporada, IMDB rating, etc.

Lucene permite hacer consultas por campos, por lo que será necesario identificar los mismos. No todos los campos en un conjunto de datos son importantes para las tareas de búsqueda y recuperación. A su vez, un mismo campo podrá utilizarse para dos funciones distintas, por ejemplo como texto sin tokenizar y texto tokenizado. Los campos concretos dependerá de la aplicación concreta sobre la que estemos trabajando, En cualquier caso, 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, por ejemplo ID, tags, direcciones webs, nombres de fichero, etc.

- TextField: Secuencia de términos que es procesada para la indexación, esto es, convertida a minúscula, tokenizada, estemizada, etc. Como podría ser el título, resumen, etc de un artículo científico o de la consulta de Stack Overflow.

- Numérico, datos que se expresan mediante este tipo de información, bien sean enteros o reales.

- Facetas (Categorías) que permiten una agrupación lógica de los documentos con la misma faceta, como por ejemplo la revista donde se publicó el trabajo, la fecha de publicación, el género de una película e incluso los actores en la misma.

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, como podría ser la valoración de los usuarios).

4.1. Búsqueda por Facetas

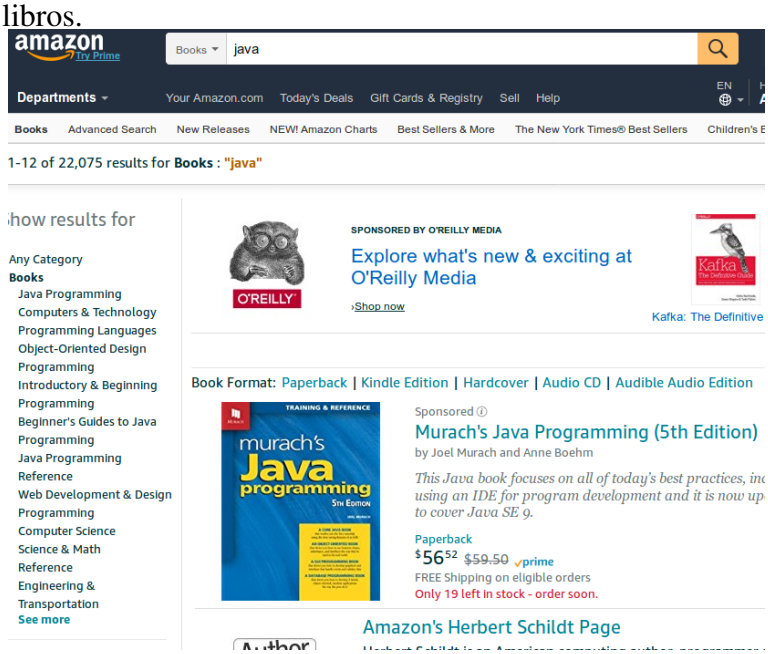
Una búsqueda por facetas nos permite acceder a la información refinando la búsqueda de acuerdo a una clasificación por categorías, filtrando los datos teniendo en cuenta las categorías a las que pertenecen. Para ello, es necesario que cada documento pueda ser clasificado a lo largo de múltiples dimensiones (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 acceder a la base de datos científica Scopus `https://www.scopus.com/search/form.uri?display=basic#basic` y 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.

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.

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



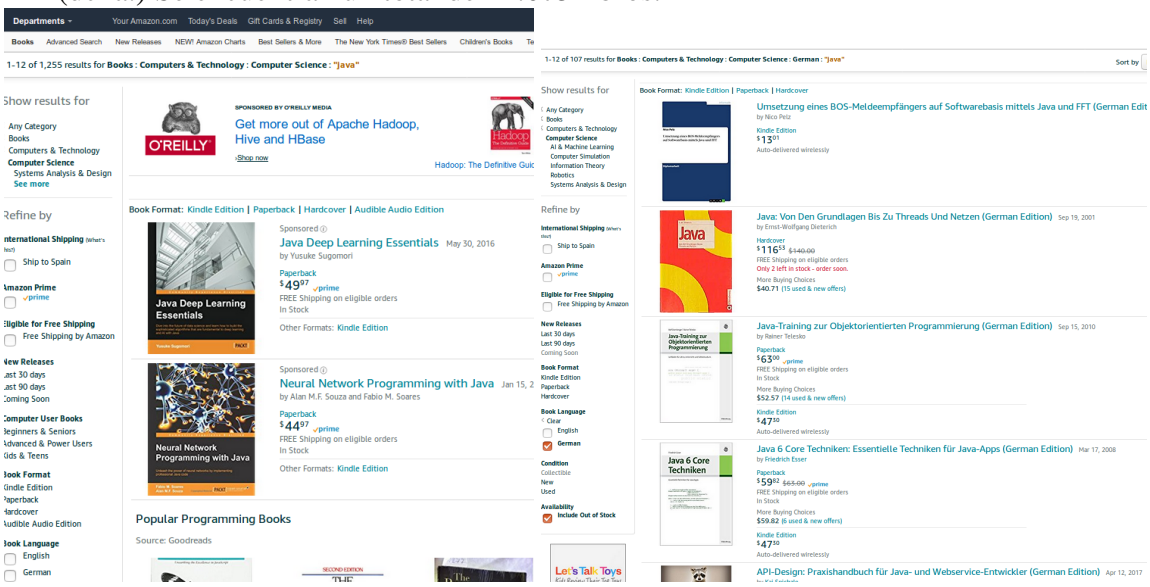
4.2. Uso de Facetas

Nuestro sistema final deberá realizar la búsqueda por facetas. Para ello en esta etapa se 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.

5. Ejercicios

1. Diseñar un pequeño programa que lea el fichero de datos y calcule la valoración media, el número de votos medios considerando la colección **AllCapitulos)** así como el número de personajes, en media, que tiene los datos en la colección **Guiones**. Se recomienda utilizar openCSV (o cualquier otra librería para el acceso correcto a los datos). Además los datos deben pasar por un proceso de limpieza y mecanismo para tratar las excepciones. El

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.



siguiente fragmento de código nos muestra como leer los datos utilzando openCSV

```
1  Reader reader = new FileReader("/home/jhg/Docencia/RI/
       Colecciones/Netflix/n_movies.csv");
2      if (!reader.ready()) {
3          System.out.println("Error en el fichero de datos");
4      }
5      CSVReader csvReader = new CSVReader(reader);
6      String[] nextRecord;
7      String[] firstLine;
8      int i = 1;
9          // ================================================
10
11         // Read documents
12     firstLine = csvReader.readNext();
13     while ((nextRecord = csvReader.readNext()) != null) {
14       int p;
15       System.out.println(i + "=====");
16       for (p = 0; p < firstLine.length; p++) {
17           System.out.println(firstLine[p] + "-> " +
```

```
                nextRecord[p]);
18          }
19          i++;
20      }
```

Es importante notar que esto es un caso de ejemplo con pocos datos. Pero en general, el fichero de entrada se procesará línea a línea (no utilizar readAll).

2. Identificar para cada campo de los distintos ficheros qué analizador utilizarías, y en caso de necesitar uno específico implementarlo. Recordad que el tipo de analizador dependerá de cómo vayáis a realizar las búsquedas por lo que es necesario plantearos antes esta cuestión.

3. Diseñar una consulta, para ello debemos de proporcionar los términos de la misma (con cuatro o mas términos) así como qué es lo que debe tener un documento para ser considerado relevante. Para encontrar los documentos (diálogos/capítulos) relevantes en la colección se debe lanzar la consulta sobre dos índices (uno para cada colección). Los índices están en el siguiente enlace `https://drive.google.com/drive/folders/17CJ5Dd9n4BTbFoL5zT_my0GlR8PouBxB?usp=sharing`

   Para ello, se recomienda utilizar `luke` para abrir cada índice por separado, generar la consulta (search) y ejecutarla contra el campo spoken_words. Analizar los 20 primeros documentos en el ranking y determinar para cada uno de ellos si el documento es relevante o no para vuestra consulta.

   La consulta será considerada válida si en la salida se obtienen tanto documentos relevantes como no relevantes.

   Se generarán dos documentos con el siguiente formato, uno por colección. Cada documento tendrá como nombre el id del grupo con el sufijo de la coleccion y la extensión `qrels`:

   ```
   Query: Grupo
   t1 t2 t3 t4
   Description:
   una linea que describa mas detalladamente la consulta
   ```

```
Index:
D o C (datos en Diálogos o Capítulos)
Relevant documents:
n
cap x1 l y1
cap x2 l y2
...
cap xn l yn
```

Por ejemplo, en nuestro caso el documento se llamaría `jhg_D.qrels`

```
Query: jhg
Christmas Carol Santa
Description:
Se es relevante si está esencialmente
relacionado con la temática navideña
Index:
D
Relevant documents:
8
cap 309 l 21
cap 320 l 93
...
```

6. Fecha de Entrega

Los resultados de esta práctica se deberán entregar el miércoles 25 de Octubre, antes de las 23:50 horas