UNIVERSIDAD DE GRANADA

RI

Recuperación de Información

Práctica 3: Indexación

Autor: Patricia Villalba Crucelaegui

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2022 - 2023

Índice

1. Ejercicio 1

Enunciado: Se pide implementar un pequeño programa que nos permite añadir varios documentos a un índice Lucene.

El ejecutable que debemos entregar deberá permitir o bien crear un índice desde el principio o añadir posibles nuevas películas a una colección ya creada. Yo he optado por crear un índice desde el principio.

Partiendo del código proporcionado en el guion, he creado mi IndiceSimple.java. Para compilarlo y ejecutarlo debemos poner el siguiente comando:

java -cp ./lucene-9.3.0/modules/lucene-core-9.3.0.jar:./lucene-9.3.0/modules/lucene-analysis-common-9.3.0.jar:./lucene-9.3.0/modules/lucene-queryparser-9.3.0.jar:./commons-csv-1.0.jar:./opencsv-4.1.jar:./commons-lang3-3.1.jar IndiceSimple.java

A continuación voy a explicar el código nuevo que se ha añadido al proporcionado por el profesor:

- Primero, he creado los dos constructores por defecto (el constructor sin parámetros y el constructor de copia), ya que los necesitaba en el main.
- Luego, para algunos campos he creado funciones auxiliares que me serán de ayuda a la hora de obtener los datos correctamente. Las funciones son las siguientes:
  - **getVotes()**: proceso los datos para que se quiten las comas, así en vez de tener el número 117,330 y tomarlo como un String, lo paso a 117330 y lo tomo como un campo numérico. Esto es por si más adelante es necesario hacer alguna operación especial en la que debo tomar los valores del campo Votes como numéricos.

```
public String getVotes(String cadena){
    //quito la coma
    if(!cadena.isEmpty()){
        cadena = cadena.replace(",", "");
    }

    return cadena;
}
```

  - **getGenre()**: al igual que antes, mientras la cadena no esté vacía, quito las comas que encuentre. El resultado sería: [comedia, drama, terror] → [comedia drama terror]. En vez de tomar el campo como un StringField, lo puedo usar como un TextField

```
public String getGenre(String cadena){
    if(!cadena.isEmpty()){
        cadena = cadena.replace(",", "");
    }

    return cadena;
}
```

  - **getDuration()**: quito todo el texto que encuentre. Así me quedo solo con el número entero de la duración. Igual que antes, hago esto por si más adelante tengo que hacer alguna operación en la que debo tomar los valores como numéricos.

```
public String getDuration(String cadena){
    //quito min y lo dejo como un entero
    if(!cadena.isEmpty()){
        cadena = cadena.replaceAll("[^0-9]", "");
    }

    return cadena;
}
```

  - **getStars()**: reemplazo todo aquello que no sean letras por la cadena vacía. El resultado sería: ['Claire Foy, ', 'Olivia Colman, '] → [Claire Foy Olivia Colman]

```
public String getStars(String cadena){
    if(!cadena.isEmpty()){
        cadena = cadena.replaceAll("[^a-zA-Z]", " ");
    }

    return cadena;
}
```

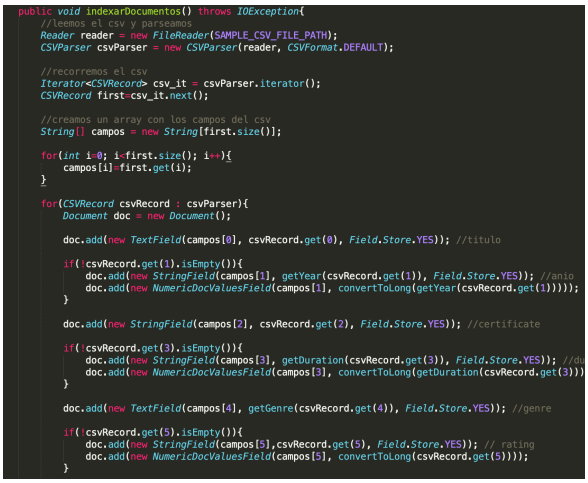
- **getYear()**: aquí lo que hago es eliminar todo aquello que no sea el número o el '–'. Luego si la cadena contiene '–', la divido en dos y me quedo con el número inicial y el final. Esto lo hago para poder ir generando los años que hay de por medio entre esos dos años.

```
public String getYear(String cadena){
    if(!cadena.isEmpty()){
        cadena = cadena.replaceAll("[^0-9–]", ""); //quito parentesis y lo que no sean numeros

        String parts[] = cadena.split("–");
        String part1 = parts[0];
        String part2 = parts[0];
        String c = "";

        if (parts.length > 1){
            part2 = parts[1];
        }

        int inicio=0, fin=0, aux=0;

        //convierto a int para poder operar con los datos
        try{
            inicio = Integer.parseInt(part1);
            fin = Integer.parseInt(part2);
        }catch (NumberFormatException ex){
            System.out.println("");
        }

        //genero anios
        if(cadena.contains("–")){
            for (aux=inicio; aux<=fin; aux++) {
                c += " " + String.valueOf(aux) ; //2005 2006 2007...
            }
            //cadena = cadena + " " + String.valueOf(aux);
            cadena = cadena.replaceAll("[–]", " ");
            cadena = c;
        }else{
            return part1;
        }
    }

    return cadena;
}
```

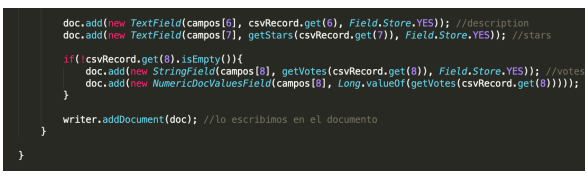
- **convertToLong()**: convierte un string a un long. (Usada para NumericDocValuesField)

```
public static long convertToLong(String cadena) {
    long valor;
    try {
        valor = Long.parseLong(cadena + "");
    } catch (NumberFormatException | NullPointerException nfe) {
        return 0; //Valor default en caso de no poder convertir  a Long
    }
    return valor;
}
```

- Luego en **indexarDocumentos** lo que hago es leer el csv por defecto que se nos da para la realización de la práctica y voy leyendo cada uno de sus campos. Recorro los valores de los campos y voy asignandoles los diferentes tipos predefinidos de Lucene. Año, duracion, rating y votes los almaceno tanto como tipo numérico como tipo String. Una vez definidos todos los tipos, los voy indexando e insertando en el documento.



```
public void indexarDocumentos() throws IOException{
    //Leemos el csv y parseamos
    Reader reader = new FileReader(SAMPLE_CSV_FILE_PATH);
    CSVParser csvParser = new CSVParser(reader, CSVFormat.DEFAULT);

    //recorremos el csv
    Iterator<CSVRecord> csv_it = csvParser.iterator();
    CSVRecord first=csv_it.next();

    //creamos un array con los campos del csv
    String[] campos = new String[first.size()];

    for(int i=0; i<first.size(); i++){
        campos[i]=first.get(i);
    }

    for(CSVRecord csvRecord : csvParser){
        Document doc = new Document();

        doc.add(new TextField(campos[0], csvRecord.get(0), Field.Store.YES)); //titulo

        if(!csvRecord.get(1).isEmpty()){
            doc.add(new StringField(campos[1], getYear(csvRecord.get(1)), Field.Store.YES)); //anio
            doc.add(new NumericDocValuesField(campos[1], convertToLong(getYear(csvRecord.get(1)))));
        }

        doc.add(new StringField(campos[2], csvRecord.get(2), Field.Store.YES)); //certificate

        if(!csvRecord.get(3).isEmpty()){
            doc.add(new StringField(campos[3], getDuration(csvRecord.get(3)), Field.Store.YES)); //du
            doc.add(new NumericDocValuesField(campos[3], convertToLong(getDuration(csvRecord.get(3)))
        }

        doc.add(new TextField(campos[4], getGenre(csvRecord.get(4)), Field.Store.YES)); //genre

        if(!csvRecord.get(5).isEmpty()){
            doc.add(new StringField(campos[5],csvRecord.get(5), Field.Store.YES)); // rating
            doc.add(new NumericDocValuesField(campos[5], convertToLong(csvRecord.get(5))));
        }
```

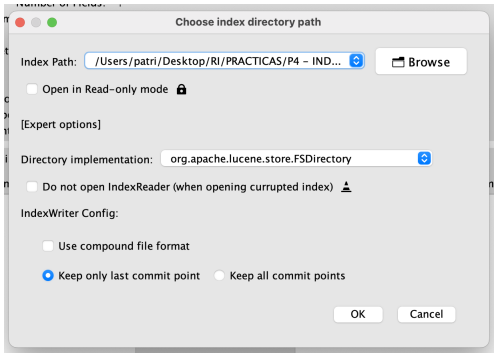


```
        doc.add(new TextField(campos[6], csvRecord.get(6), Field.Store.YES)); //description
        doc.add(new TextField(campos[7], getStars(csvRecord.get(7)), Field.Store.YES)); //stars

        if(!csvRecord.get(8).isEmpty()){
            doc.add(new StringField(campos[8], getVotes(csvRecord.get(8)), Field.Store.YES)); //votes
            doc.add(new NumericDocValuesField(campos[8], Long.valueOf(getVotes(csvRecord.get(8)))));
        }

        writer.addDocument(doc); //lo escribimos en el documento
    }

}
```

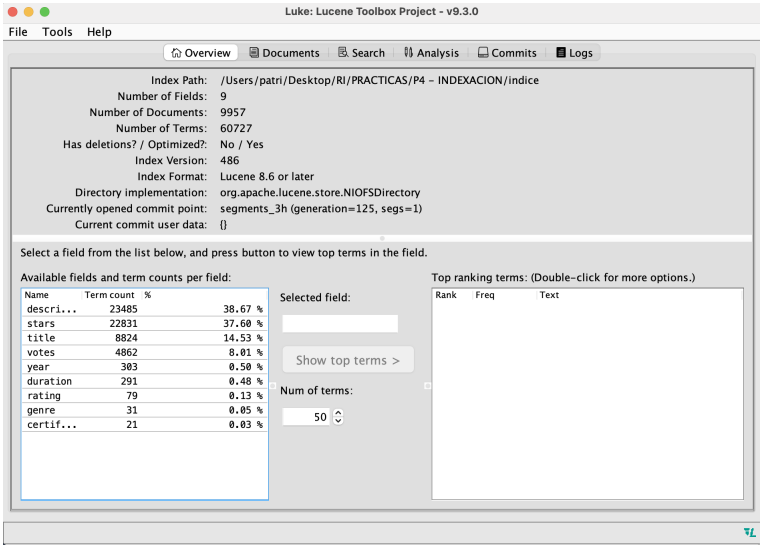
2. Ejercicio 2

Enunciado: Utilizar Luke para ver el índice y realizar distintas consultas sobre el mismo.

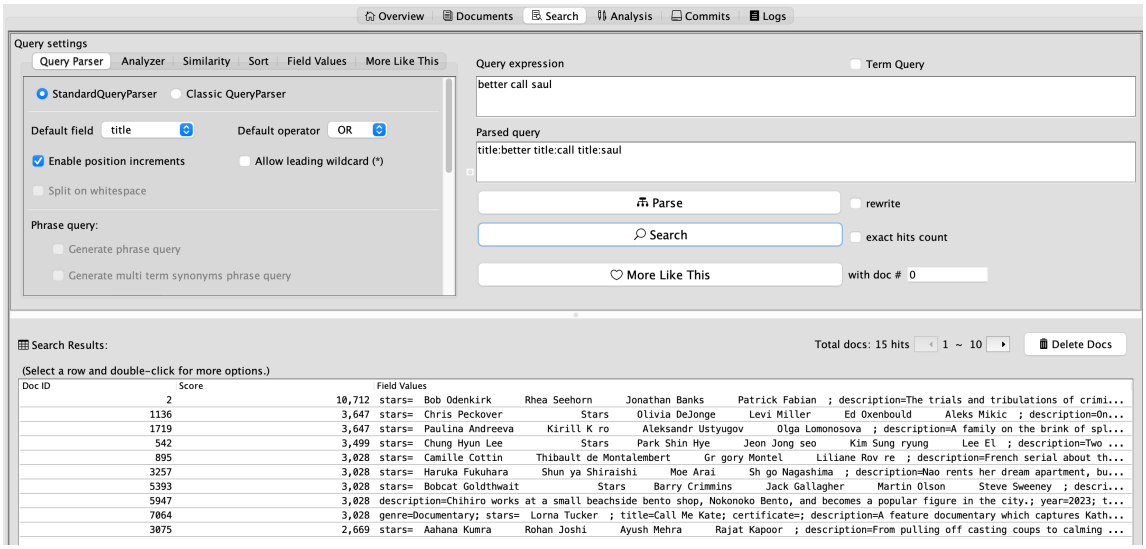
Una vez hemos compilado y ejecutado el código anterior, pasamos a utilizar luke. Luke es una herramienta con la que podemos acceder gráficamente a un índice Lucene. Procedemos a abrir el enlace que hemos creado:



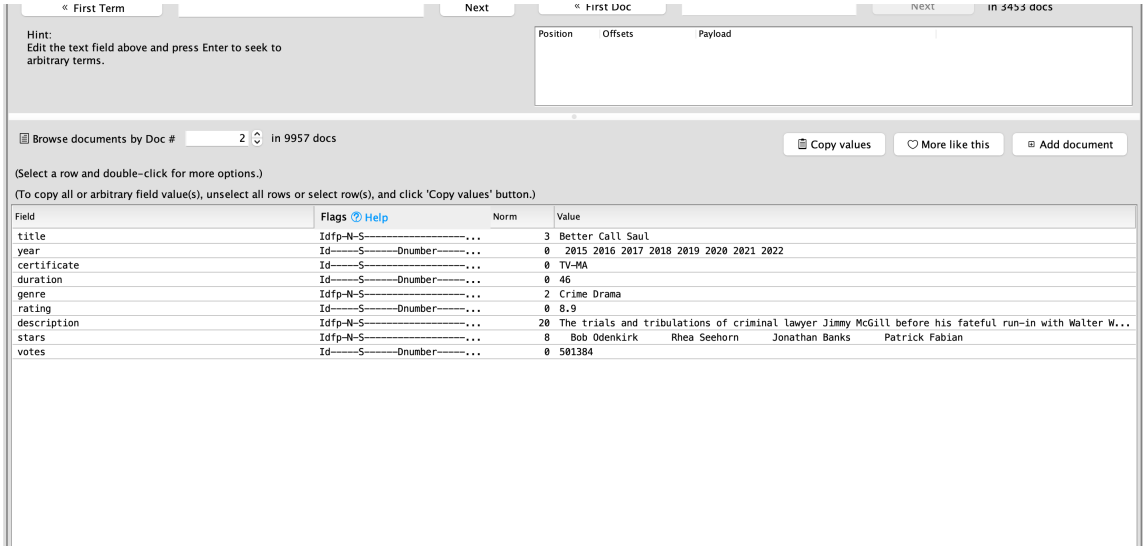
Una vez abierto, podemos ver como ha quedado nuestro índice: nombre del campo, número de términos y su porcentaje.



Ahora vamos a realizar diversas consultas sobre nuestro índice para poder observar su comportamiento. Las búsquedas las vamos a realizar por campos y de forma ordenada.
Vamos a empezar con algo sencillo; vamos a hacer una búsqueda por título y vamos a buscar por ejemplo la película 'Better call saul'. Escribimos en query expresion el titulo de la pelicula que vamos a buscar. El resultado será el siguiente:

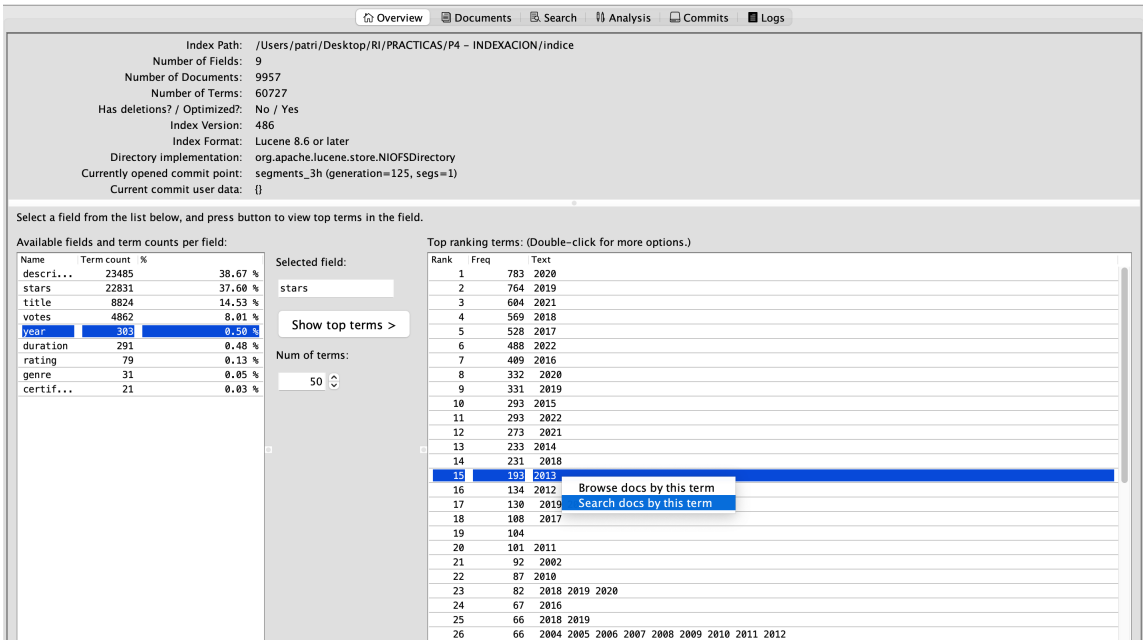


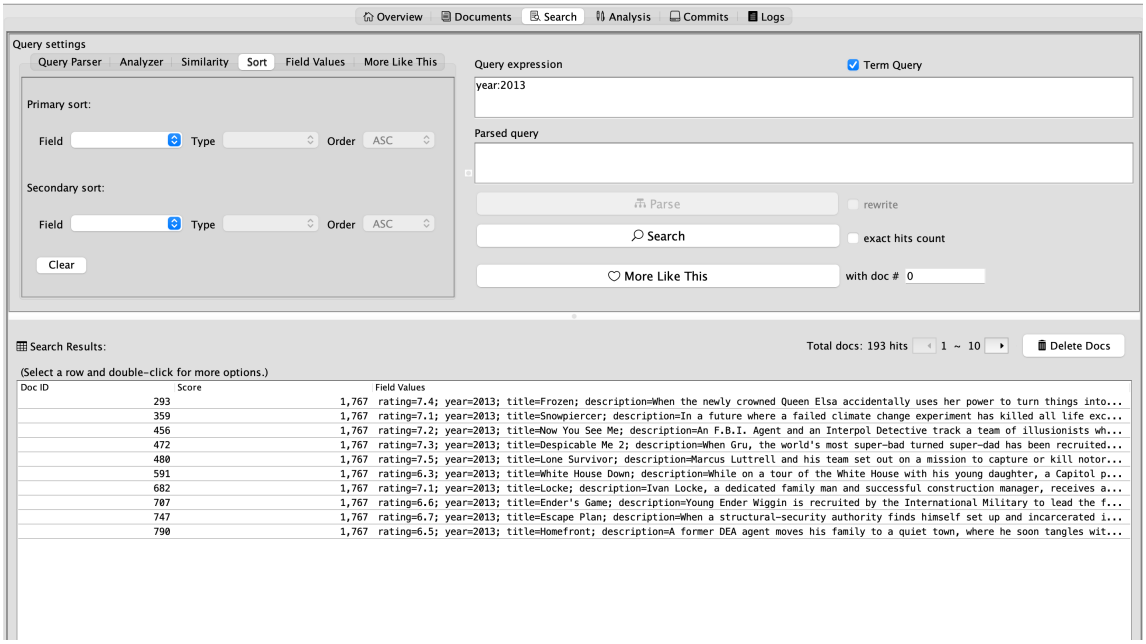
Pinchamos por ejemplo sobre el primero y le damos dos veces click y seleccionamos la opción 'show all fields'. Nos saldrá lo siguiente:



Como podemos ver, obtenemos los diferentes campos que hemos indexado con la información correspondiente a cada uno. Se puede observar que year, duration, rating y votes son de tipo string y numérico. También podemos observar como la película que iba desde el año 2015-2022 muestra todos los demás años que hay de por medio. Observamos como todos los campos tienen sus datos procesados gracias a las funciones auxiliares que he explicado antes.

Ahora vamos a hacer otro tipo de consulta. Volvemos a la pestaña de 'overview' y elegimos algún campo y seleccionamos el término que queramos. Clickamos dos veces sobre el y le damos a buscar documentos por ese término. Esto nos debería dar una búsqueda correcta.





Vemos como la consulta se realiza sin ningún tipo de problema.

Por lo tanto, luke es una herramienta muy útil para poder manejar y tratar con los datos una vez que están indexados.