



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

ETSIIT

Escuela Técnica Superior
de Ingenierías Informática
y de Telecomunicación



Índice

1. Ejercicio 1	2
2. Ejercicio 2	4

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 todo lo que no sea 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[1];
        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 años
        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_csv_it_next().size()];

    for(int i=0; i<first_csv_it_next().size(); i++){
        campos[i] = first_csv_it_next().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)); //dur
            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

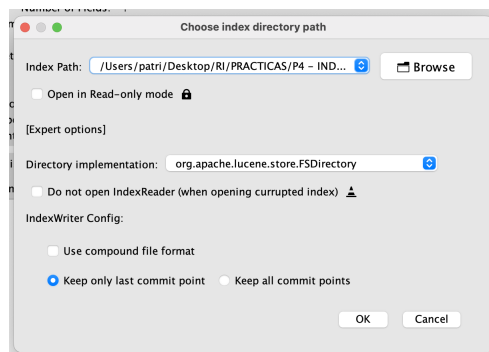
if(!csvRecord.get(8).isEmpty()){
    doc.add(new StringField(campos[8], getVotes(csvRecord.get(8)), Field.Store.YES)); //votes
    doc.add(new NumericDocValuesField(campos[8], Long.valueOf(getVotes(csvRecord.get(8)))));
}

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

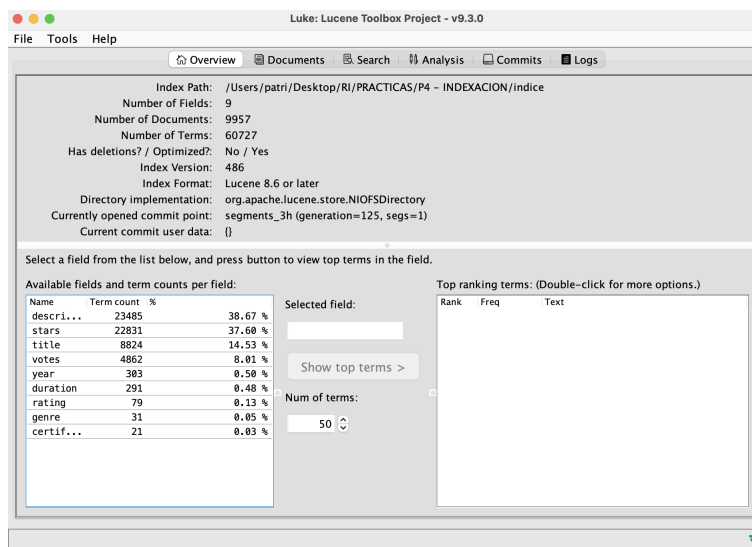
2. Ejercicio 2

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 título de la película que vamos a buscar. El resultado será el siguiente:

Query settings

Query Parser: ☒ StandardQueryParser ☐ Classic QueryParser

Default field: Default operator:

☒ Enable position increments ☐ Allow leading wildcard (*)

☐ Split on whitespace

Phrase query:

☐ Generate phrase query

☐ Generate multi term synonyms phrase query

Query expression:

☐ Term Query

Parsed query:

☐ rewrite

☐ exact hits count

with doc #

Search Results: Total docs: 15 hits 1 ~ 10 Delete Docs

(Select a row and double-click for more options.)

Doc ID	Score	Field Values
2	10,712	stars= Bob Odenkirk Rhea Seehorn Jonathan Banks Patrick Fabian ; description=The trials and tribulations of crimi...
1136	3,647	stars= Chris Peckover Stars Olivia DeJonge Levi Miller Ed Oxenbould Aleks Mikic ; description=On...
1719	3,647	stars= Paulina Andreeva Kirill K ro Aleksandr Ustyugov Olga Lomonosova ; description=A family on the brink of spl...
542	3,499	stars= Chung Hyun Lee Stars Park Shin Hye Jeon Jong seo Kim Sung ryung Lee El ; description=Two ...
895	3,028	stars= Camille Cottin Thibault de Montalembert Gr gory Montel Liliane Rov re ; description=French serial about th...
3257	3,028	stars= Haruka Fukuhara Shun ya Shiraishi Moe Arai Sh go Nagashima ; description=Nao rents her dream apartment, bu...
5393	3,028	stars= Bobcat Goldthwait Stars Barry Crimmins Jack Gallagher Martin Olson Steve Sweeney ; descri...
5947	3,028	description=Chihiro works at a small beachside bento shop, Nokonoko Bento, and becomes a popular figure in the city.; years=2023; t...
7064	3,028	genre=Documentary; stars= Lorna Tucker ; title=Call Me Kate; certificate= ; description=A feature documentary which captures Kath...
3075	2,669	stars= Aahana Kumra Rohan Joshi Ayush Mehra Rajat Kapoor ; description=From pulling off casting coups to calming ...

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

« First Term Next « First Doc NEXT In 9957 docs

Hint: Edit the text field above and press Enter to seek to arbitrary terms.

In 9957 docs

(Select a row and double-click for more options.)

(To copy all or arbitrary field value(s), unselect all rows or select row(s), and click 'Copy values' button.)

Field	Flags Help	Norm	Value
title	Idfp-N-S-----	3	Better Call Saul
year	Id-----S-----Dnumber-----	0	2015 2016 2017 2018 2019 2020 2021 2022
certificate	Id-----S-----	0	TV-MA
duration	Id-----S-----Dnumber-----	0	46
genre	Idfp-N-S-----	2	Crime Drama
rating	Id-----S-----Dnumber-----	0	8.9
description	Idfp-N-S-----	20	The trials and tribulations of criminal lawyer Jimmy McGill before his fateful run-in with Walter W...
stars	Idfp-N-S-----	8	Bob Odenkirk Rhea Seehorn Jonathan Banks Patrick Fabian
votes	Id-----S-----Dnumber-----	0	501384

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.

Overview Documents Search Analysis Commits Logs

Index Path: /Users/patri/Desktop/RI/PRACTICAS/P4 - INDEXACION/indice
 Number of Fields: 9
 Number of Documents: 9957
 Number of Terms: 60727
 Has deletions? / Optimized?: No / Yes
 Index Version: 486
 Index Format: Lucene 8.6 or later
 Directory Implementation: org.apache.lucene.store.NIOFSDirectory
 Currently opened commit point: segments_3h (generation=125, segs=1)
 Current commit user data: {}

Select a field from the list below, and press button to view top terms in the field.

Available fields and term counts per field:

Name	Term count	%
descri...	23485	38.67 %
stars	22831	37.60 %
title	8824	14.53 %
votes	4862	8.01 %
year	303	0.50 %
duration	291	0.48 %
rating	79	0.13 %
genre	31	0.05 %
certif...	21	0.03 %

Selected field: stars

Show top terms >

Num of terms: 50

Top ranking terms: (Double-click for more options.)

Rank	Freq	Text
1	783	2020
2	764	2019
3	684	2021
4	569	2018
5	528	2017
6	488	2022
7	409	2016
8	332	2020
9	331	2019
10	293	2015
11	293	2022
12	273	2021
13	233	2014
14	231	2018
15	193	2013
16	134	2012
17	130	2010
18	108	2017
19	104	
20	101	2011
21	92	2002
22	87	2010
23	82	2018 2019 2020
24	67	2016
25	66	2018 2019
26	66	2004 2005 2006 2007 2008 2009 2010 2011 2012

Browse docs by this term
Search docs by this term

Overview Documents Search Analysis Commits Logs

Query settings

Query Parser Analyzer Similarity Sort Field Values More Like This

Primary sort:

Field Type Order ASC

Secondary sort:

Field Type Order ASC

Clear

Query expression ☒ Term Query

year:2013

Parsed query

Parse

rewrite

Search

exact hits count

More Like This with doc # 0

Search Results: Total docs: 193 hits 1 ~ 10 Delete Docs

(Select a row and double-click for more options.)

Doc ID	Score	Field Values
293	1,767	rating=7.4; year=2013; title=Frozen; description=When the newly crowned Queen Elsa accidentally uses her power to turn things into...
359	1,767	rating=7.1; year=2013; title=Snooper; description=In a future where a failed climate change experiment has killed all life ext...
456	1,767	rating=7.2; year=2013; title=Now You See Me; description=An F.B.I. Agent and an Interpol Detective track a team of illusionists wh...
472	1,767	rating=7.3; year=2013; title=Despicable Me 2; description=When Gru, the world's most super-bad turned super-dad has been recruited...
480	1,767	rating=7.5; year=2013; title=Lone Survivor; description=Marcus Luttrell and his team set out on a mission to capture or kill notor...
591	1,767	rating=6.3; year=2013; title=White House Down; description=While on a tour of the White House with his young daughter, a Capitol p...
682	1,767	rating=7.1; year=2013; title=Locke; description=Ivan Locke, a dedicated family man and successful construction manager, receives a...
707	1,767	rating=6.6; year=2013; title=Ender's Game; description=Young Ender Wiggin is recruited by the International Military to lead the f...
747	1,767	rating=6.7; year=2013; title=Escape Plan; description=When a structural-security authority finds himself set up and incarcerated i...
790	1,767	rating=6.5; year=2013; title=Homefront; description=A former DEA agent moves his family to a quiet town, where he soon tangles wit...

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.