PRÁCTICA III:

...

Implementación de un Sistema de Recuperación de Información utilizando Lucene

...

Indexación de Facetas

22 de noviembre de 2023

Índice

1. Objetivo

El objetivo de esta práctica es profundizar en el uso de facetas categorías en un SRI. Para ello, el alumno deberá de incluir dicha estrategia de búsqueda en la aplicación que esta diseñando. Aunque la búsqueda por facetas es mas lenta que una búsqueda normal (son muchos los cálculos internos que tenemos que hacer), su eficiencia se mantiene proporcional al número de hits (documentos que emparejan con la consulta).

El paquete facetas de Lucene hace fácil añadir categorías a una aplicación de búsqueda, dotándolo de características interesantes como muestro aleatorio (para el caso en que son muchos los documentos que emparejan con la consulta, distintos mecanismos de agregación, permitir tener las facetas en RAM, interfaces personalizados, etc.).

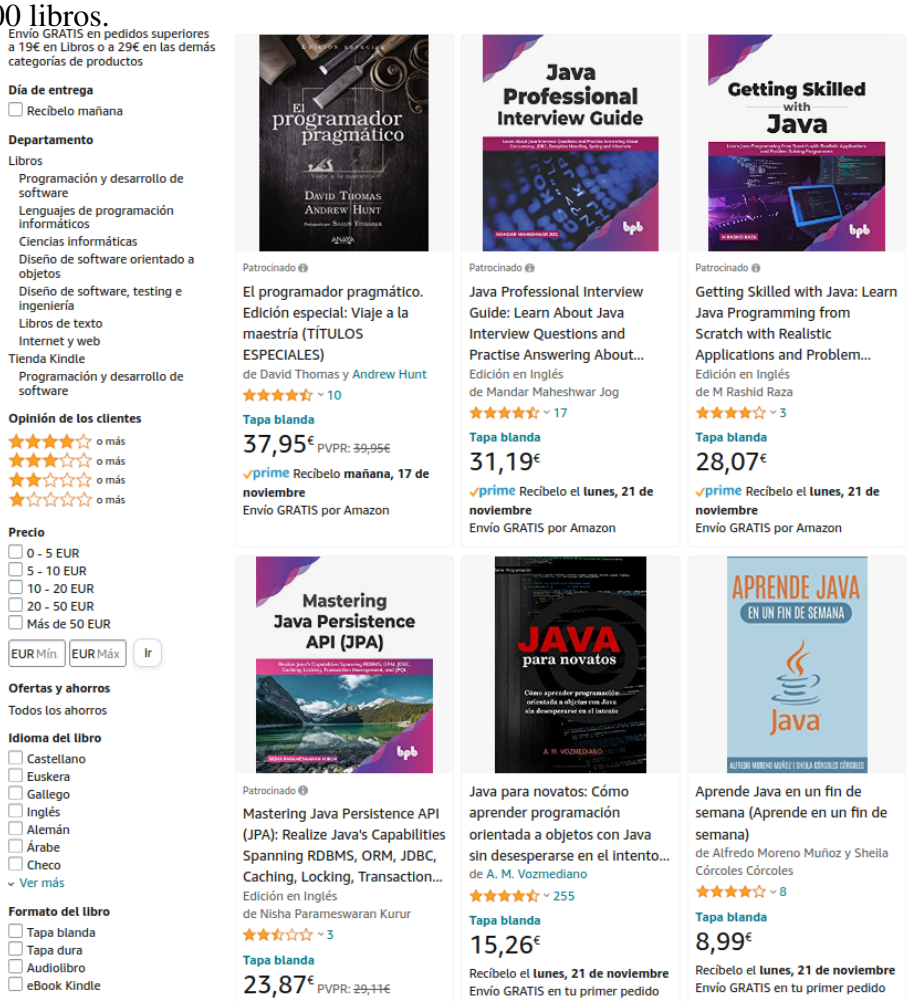
2. Búsqueda por Facetas

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 electrónico 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.

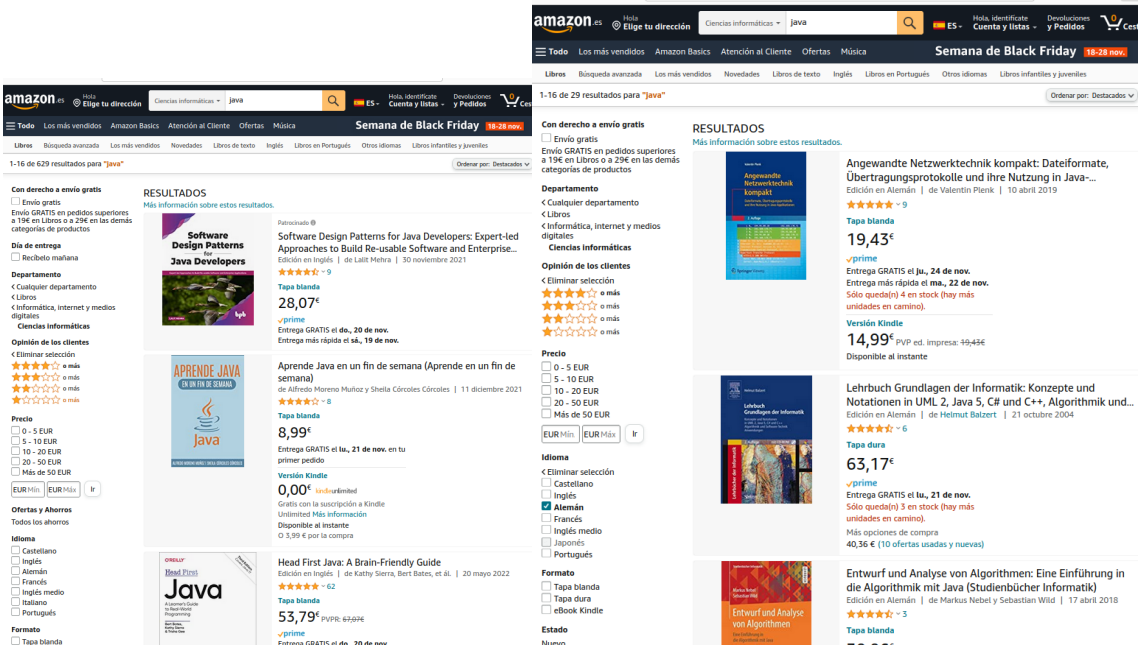
Esta peculiaridad ha hecho que la búsqueda por facetas sea muy común en sitios de comercio electrónico, como por ejemplo Amazon. En la Figura 1 podemos ver cómo ante la consulta “Java” encontramos más de 20000 libros en el portal de ventas Amazon. A la izquierda de la misma encontramos un frame en el 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 más de 20000 libros.



Así, podemos centrar la búsqueda dentro de la categoría Computer-Science

Figura 2: Consulta: “Java”, restringimos la búsqueda a los libros que se encuadran dentro de la categoría computer-science (ciencias informáticas) (izq.) o computer-science ->Alemán (dcha.) Se encuentran un total de 29 libros.



(imagen a la izquierda de la Figura 2), encontrando un total de 629 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.

Otro ejemplo lo podemos considerar si buscamos un libro de novela histórica ambientado en España. Para ello, podríamos hacer la búsqueda en Amazon "Novela histórica ambientada en España", obteniendo los resultados en la figura **??** donde de nuevo podemos ver a la izquierda las posibles categorías o facetas por los que podemos afinar la búsqueda.

Si analizamos con detalle lo que nos muestra la página de Amazon, encontramos a su derecha que los resultados se encuentran ordenados por el criterio "Destacados", otras alternativas son precio o valoración de los usuarios. Ya vimos en la práctica anterior cómo podríamos realizar la ordenación.

Si nos fijamos en las categorías, podemos ver como se nos nuestra una especie de árbol del Departamento. En concreto, en la figura vemos como nos encontramos dentro del departamento de Literatura y ficción $->$ Acción y Aventura, que

Figura 3: Búsqueda de libros en Amazon, consulta: “Novel Histórica España”. Se encuentran más de 20000 libros.

a su vez se divide en distintas subcategorías. De alguna forma, se ha establecido una jerarquía sobre las categorías por la cual podremos navegar.

Entonces, nos podemos preguntar cómo se podría filtrar y quedarnos por ejemplo con novelas `Acción bélica y militar`. Una alternativa sería recorrer el conjunto de documentos recuperados (libros) y filtrar aquellos que tuviesen almacenado en un campo una etiqueta asociada a la categoría `Acción bélica y militar`. Esto sería un proceso costoso, por lo que Lucene otra alternativa, el uso de facetas.

3. Facetas en Lucene

La facetas fueron incluidas en Lucene en la versión 3.4, y podemos encontrar una descripción detallada en `http://lucene.apache.org/core/4_2_0/facet/org/apache/lucene/facet/doc-files/userguide.html`. Aunque dicha guía ha sido borrada de la documentación de Lucene hace años, y por tanto no se encuentra actualizadas, es posible encontrar ejemplos de su uso en la versión 9.3.0 de Lucene en `https://lucene.apache.org/core/9_3_0/demo/src-html/org/apache/lucene/demo/facet/`.

Para poder realizar una búsqueda con facetas es necesario añadirlas en tiempo de indexación. Para ello, Lucene utiliza dos "índices" distintos: uno que almacena la información sobre las categorías (facetas) y otro, el índice normal, que almacena los documentos Lucene. Por tanto, necesitaremos de dos Directory distintos, que nos permitirán almacenar cada uno de estos índices por separado y dos writers diferentes, IndexWriter y DirectoryTaxonomyWriter, que se encargarán de crear las estructuras necesarias para los índices. El DirectoryTaxonomyWriter es un TaxonomyWriter que utiliza un Directory para almacenar la información de las categorías (taxonomía) en disco.

```
1  FSDirectory indexDir = new FSDirectory.open(Paths.get(
       INDEX_DIR));
2  FSDirectory taxoDir =  new FSDirectory.open(Paths.get(
       FACET_DIR));
3
4  IndexWriterConfig iwconfig = new IndexWriterConfig(new
       WhitespaceAnalyzer());
5  iwconfig.setOpenMode(OpenMode.CREATE);
6  FacetsConfig fconfig = new FacetsConfig();
7
8  IndexWriter indexWriter = new IndexWriter(indexDir, iwconfig);
9  DirectoryTaxonomyWriter taxoWriter = new
       DirectoryTaxonomyWriter(taxoDir);
```

Al igual que podemos configurar el IndexWriter, podremos configurar las facetas utilizando la clase FacetsConfig, que permite almacenar la configuración para las distintas facetas. Por defecto, si no se indica nada, la configuración de una faceta tendrá

- un único valor (por ejemplo, si consideramos el año de nacimiento como faceta estamos hablando de valor único, pero si consideramos que una asignatura esta puede ser impartida por varios profesores podemos hablar de una faceta multivaluada)
- una única dimensión (el año tendrá una única dimensión, pero el género puede tener dos o mas como cuando consideramos un segundo nivel dentro de la categoría novela, como por ejemplo romántica, aventuras, etc.)

Para ello, podemos utilizar los "setters" de FacetsConfig, como por ejemplo:

```
1 fconfig.setMultiValued("Profesor", true);
2 fconfig.setHierarchical("Titulacion", true);
```

Una vez que hemos configurado las facetas, podemos pasar a asignarles su valor para cada uno de los documentos e indexarlas.

Cuando construimos las facetas debemos de crearnos un FacetFields, que permite asignar pares categoría-valor a cada documento. El siguiente código muestra cómo podemos añadir distintos elementos (facetas o no) a un documento Lucene, el cual puede pasar a ser indexado (línea 20). Es importante indicar que en este caso no indexamos el documento Lucene tal cual lo hemos creado, sino el documento Lucene que se construye cuando consideramos toda la información asociada a las facetas, mediante el método build de FacetsConfig.

```
1
2 Document doc = new Document();
3 .....
4 // Incluimos los campos de indexacion
5
6 doc.add(new StringField("Codigo","123456",Field.Store.YES));
7 doc.add(new TextField("Nombre","Nombre de la Asignatura ....",
      Field.Store.YES));
8 doc.add(new TextField("Resumen","Esta asignatura presenta ....",
      Field.Store.NO));
9
10 // Incluimos las  facetas
11
12 doc.add(new FacetField("Profesor", "nombre1 apellido1"));
13 doc.add(new FacetField("Profesor", "nombre2 apellido2"));
```

```
14 doc.add(new FacetField("Profesor", "nombre3 apellido3"));
15
16 doc.add(new FacetField("Titulacion","Grado en Informatica", "
       Sistemas de Informacion"));
17
18 // Indexamos el documento.
19
20 indexWriter.addDocument(fconfig.build(taxoWriter,doc));
```

Finalmente, tendremos que cerrar los writers que hemos utilizado

```
1 indexWriter.close();
2 taxoWriter.close();
```

4. Búsqueda y Facetas

El módulo de facetas en Lucene se caracteriza por realizar la mayor parte del trabajo en tiempo de indexación. Para cada uno de los documentos indexados, Lucene analiza todas sus facetas, algunas de las cuales pueden ser jerárquicas, y mapea cada una de ellas con un único identificador (entero), los cuales se almacenan dentro de un campo del tipo docValues, que en resumen permiten obtener eficientemente todos los valores de un campo en distintos documentos. Un índice separado (TaxonomyWriter) almacena estos emparejamientos asegurándose que siempre la misma etiqueta sea asociada al mismo identificador.

En tiempo de búsqueda el costo es mínimo: Para cada uno de los documentos en la lista de resultados, se consultan todos los identificadores y se agregan los conteos en un array, resumiendo los resultados al final si es necesario quedarnos con las top N facetas. En el caso de no hacerlo así, como por ejemplo lo hacen las implementaciones de Solr o ElasticSearch, el obtener las facetas asociadas a un documento hay que hacerlo en tiempo de búsqueda, siendo por tanto menos eficiente. En cualquier caso, esta segunda aproximación permitirá una mayor flexibilidad.

La documentación de facetas en Lucene 9.3 la podemos encontrar en `https://lucene.apache.org/core/9_3_0/facet/org/apache/lucene/facet/package-summary.html`

Como es lógico, para realizar una búsqueda por facetas debemos de inicializar algunas componentes que nos permitirán leer los dos índices, el índice invertido y el índice que contiene los árboles de categorías. Ambos deben ser abierto es modo lectura.

```
1 DirectoryReader indexReader = DirectoryReader.open(indexDir);
2 IndexSearcher searcher = new IndexSearcher(indexReader);
3 TaxonomyReader taxoReader =new DirectoryTaxonomyReader(taxoDir);
```

DirectoryReader e indexReader se utilizan para realizar las búsquedas sobre el índice (como hemos visto en el apartado de búsquedas de las prácticas). TaxonomyReader es el encargado de realizar la búsqueda sobre las facetas.

En general, la búsqueda por facetas implica contar, de entre los documentos recuperados, cuántos documentos emparejan con una determinada faceta o categoría. Por tanto, requiere el uso de colectores específicos que permitan agilizar dicha tarea. Tanto es así, que es el colector el que realiza la búsqueda para asegurar que obtiene de forma eficiente la información que necesita.

Para ejecutar las búsqueda con facetas y recuperar las categorías para un determinada consulta debemos utilizar la clase FacetCollector `https://lucene.apache.org/core/9_3_0/facet/org/apache/lucene/facet/FacetsCollector.html`. Recibe una lista de nodos de un árbol de categorías y devuelve los valores (número de hits) para cada uno de los descendientes en el árbol. Un nodo puede ocupar cualquier posición dentro del árbol de categoría (por ejemplo puede ser la raíz como "/Titulación/" o un descendiente como "/Titulacion/Grado en Informática/".

```
1 FacetsCollector fc  = new FacetsCollector( );
2 TopDocs tdc = FacetsCollector.search(searcher, query, 10, fc);
```

Para realizar la búsqueda por facetas debemos utilizar uno de los distintos métodos `search*` de la clase FacetCollector. Los parámetros del método `search` son el IndexSearcher, la consulta Lucene, el número de documentos recuperados y el objeto de la clase FacetCollector encargado de recoger las facetas asociadas a los documentos que emparejan con la consulta. Este objeto nos permitirá realizar la agrupación de resultados en las distintas categorías a las que pertenecen (faceting).

Tras ejecutar la búsqueda se obtienen dos salidas distintas:

- Los documentos que emparejan con la consulta, los resultados de la búsqueda Lucene, son devueltos en un objeto de la clase TopDocs o TopFieldDocs. Con este objeto podemos iterar sobre todos los documentos recuperados de la forma usual y recuperar los campos que nos interesen almacenados en el documento Lucene.

```
1 for (ScoreDoc sd : tdc.scoreDocs) {
2     Document d = searcher.doc(sd.doc);
3     System.out.println(sd.score + "Asig: " + d.get("
          asignatura") );
4 }
```

- Las facetas son recogidas dentro del collector que se pasa como parámetro. Una vez que hemos recogido las facetas, podemos instanciar una de las distintas subclases de Facets para realizar el conteo de las mismas (lo normal es contar cuantos documentos se asocian a una determinada faceta, pero también podemos estar interesados en asociar a cada faceta la suma de los scores de los documentos recuperados).

  Para realizar el conteo se utiliza la clase `FastTaxonomyFacetCounts` que recupera el número de documentos que emparejan con cada faceta y las devuelve en un objeto de tipo `Facets` que podrá ser utilizado para mostrar los elementos..

```
1 Facets facetas = new FastTaxonomyFacetCounts(taxoReader,
      fconfig, fc);
```

4.1. Mostramos los resultados obtenidos para las facetas

Una vez que tenemos el objeto de tipo Facets podemos realizar las siguientes acciones:

- `List<FacetResults> getAllDims(int topN)` para devolver una lista con los topN FacetResults (estructura que entre otros contiene las etiquetas y los contadores -valores- asociados) de cada dimensión o categoría, ordenadas por el número de hits con los que emparejan. Se suele utilizar

cuando no todos los documentos son clasificados bajo las mismas categoría, como por ejemplo en e-comercio donde se dispongan de múltiples tipos de productos.

- `FacetResults getTopChildren(int topN, String dim, String... path)` Devuelve los topN FacetResults de una determinada categoría (podemos dar el camino hasta la misma). Por ejemplo, clasificamos las asignaturas bajo la dimensión "Titulacion", y permitimos la jerarquía "Grado/Especialidad/" las siguientes llamadas nos permiten recuperar las facetas para todas las titulaciones, o sólo para el "Grado en Informatica".

```
1  getTopChildren(10,"Titulacion");
2  getTopChildren(10, "Titulacion", "Grado en Informatica");
```

- `Number getSpecificValue(String dim, String... path)` Devuelve el número (o valor numérico computado) para una categoría concreta.

El siguiente ejemplo nos muestra como mostrar los distintos resultados obtenidos. Para ello, podemos recorrer la lista de FacetsResult pudiendo obtener entre otros datos, el nombre de la categoría o dimensión (por ejemplo, Profesorado) y para cada dimensión consultar para cada una de sus etiquetas (nombre del profesor) y valor (número de asignaturas que emparejan con la consulta)

```
1  Facets facetas = new FastTaxonomyFacetCounts(taxoReader, fconfig
       , fc);
2
3  // Iterando sobre las facetas obtenidas //
4
5  List<FacetResult> TodasDims = facetas.getAllDims(100);
6  System.out.println("Categorias totales " + TodasDims.size());
7  for (FacetResult fr: TodasDims) {
8              System.out.println("Categoria " + fr.dim);
9              for (LabelAndValue lv : fr.labelValues){
10                 System.out.println("    Etiq: " + lv.label + ",
                       valor (#n)->" + lv.value);
11             }
12         }
```

```
13
14 FacetResult fresult = facetas.getTopChildren(10, "Profesores"));
```

Así, una posible salida de este código ante una consulta sería:

```
1  Categorias totales 3
2  Categoria Centro
3      Etiq: ETSII, valor (#n)->4
4      Etiq: FacDoc, valor (#n)->1
5  Categoria cuatrimestre
6      Etiq: segundo, valor (#n)->3
7      Etiq: primero, valor (#n)->2
8  Categoria Profesores
9      Etiq: Silvia Acid, valor (#n)->4
10     Etiq: Juan Huete, valor (#n)->2
11     Etiq: Nicolas Perez, valor (#n)->1
12     Etiq: Andres Cano, valor (#n)->1
13     Etiq: Francisco Cortijo, valor (#n)->1
```

4.2. Otros mecanismos de agregación para las facetas

Como hemos comentado, en algunas aplicaciones es posible que deseemos considerar otros mecanismos -mas allá del simple conteo- para cuantificar la importancia de una faceta ante la consulta. En

`https://lucene.apache.org/core/9_3_0/demo/src-html/org/apache/lucene/demo/facet/`

se encuentran distintos ejemplos para el uso de facetas y la posible agregación de resultados. De ellos destacaremos:

- `SimpleFacetsExample`. Ejemplo simple que utiliza un FastTaxonomyFacetCounts
- `RangeFacetsExample` donde se muestra como acumular valores por rangos definidos por el usuario.
- `SimpleSortedSetFacetsExample`. Ejemplo donde se muestra como calcular conteos de facetas sobre un campo Faceta que utiliza un SortedSetDocValues, esto es, un SortedSetDocValuesFacetField. Se utiliza cuando

estamos interesados en mantener un orden sobre las categorías (múltiples valores con un orden específico)

- `ExpressionAggregationFacetsExample` que nos muestra como acumular valores utilizando una expresión generada por el usuario. Por ejemplo, podemos acumular los valores almacendados en un determinado campo

4.2.1. Ejemplo: Agrupando valores por rango definidos por el usuario

Este ejemplo nos muestra como agrupar los valores almacenados en los DocValues por rangos definidos por el usuario como puede ser el precio de los libros en la búsquedas que hemos realizado con Amazon (ver figuras 1 y 3. El siguiente ejemplo de código nos muestra como realizarlo.

```
1 LongRange[] ranges = new LongRange[3];
2 ranges[0] = new LongRange("10-30", 10L, true, 30L, false);
3 ranges[1] = new LongRange("21-59", 31L, true, 60L, false);
4 ranges[2] = new LongRange("60-100", 60L, true, 100L, true);
5
6 FacetsCollector fcRango = new FacetsCollector();
7 FacetsCollector.search(searcher, new MatchAllDocsQuery(), 10,
     fcRango);
8 LongRangeFacetCounts facets = new LongRangeFacetCounts("precio",
      fcRango, ranges);
9 FacetResult resultado  = facets.getTopChildren(0, "precio");
```

Otro ejemplo, lo podemos encontrar en `https://lucene.apache.org/core/9_3_0/demo/src-html/org/apache/lucene/demo/facet/RangeFacetsExam html`

5. Filtrando la búsqueda

Como hemos comentado, en una aplicación típica que use facetas, podemos ver una columna a la izquierda con los conteos del número de documentos recuperados que empareja con cada faceta. Esto nos indica cuántos documentos podemos

recuperar si restringimos (DrillDownQuery) la búsqueda a esa faceta, tras añadirla como restricción a la búsqueda actual. Por tanto, DrillDown añade filtros a la búsqueda, pudiendo utilizar múltiples categorías (OR lógico)

```
1  Query q1 = new MatchAllDocsQuery();
2
3  DrillDownQuery ddq = new DrillDownQuery(fconfig,q1 );
4  ddq.add("Centro","ETSII");
5  //ddq.add("Profesores","Juan Huete"); //Icluimos una segunda
       restriccion OR
6
7  System.out.println("  Filtramos query[ "+ ddq.toString() +" ]");
8
9  FacetsCollector fc1 = new FacetsCollector();
10 TopDocs td2 = FacetsCollector.search(searcher, ddq, 10, fc1);
11 System.out.println("     Total hits = " + td2.totalHits);
12 Facets fcCount2 = new FastTaxonomyFacetCounts(taxoReader,
       fconfig, fc1);
13 List<FacetsResults> allDims = fcCount2.getAllDims(100);
14
15
16 System.out.println("Categorias "+ allDims.size());
17 for ( FacetResult fr : allDims) {
18
19     System.out.println("Dimension " + fr.dim);
20     for (LabelAndValue lv: fr.labelValues){
21        System.out.println("    "+lv.label + ":: #->" + lv.value)
              ;
22     }
23 }
24
25 for (ScoreDoc scoreDoc : td2.scoreDocs) {
26          doc = searcher.doc(scoreDoc.doc);
27          System.out.println("  Docs Score-> " + scoreDoc.score +
                "::"   + doc.get("asignatura"));
28        }
```

Si hacemos esto aquellas facetas que no concuerdan dentro de la opción de búsqueda desaparecen, por ejemplo se eliminan del árbol los resultados asociados a centros distintos de la ETSII, lo cual puede parecer normal. Pero si este proceso

lo repetimos reiteradamente sobre las distintas opciones de búsqueda, al final si queremos deshacer algunos de los cambios nos resultará imposible, teniendo que volver a empezar la navegación desde el principio o utilizar el botón de vuelta atrás.

Una alternativa es utilizar DrillSideWays, que nos asegura no perder los conteos de otras facetas al restringir la búsqueda sobre una de ellas. Esto es, podemos dirigir la búsqueda sobre múltiples categorías.

```
1  System.out.println("Ahora con DrillSideWays");
2  q1 = new MatchAllDocsQuery();
3  System.out.println("QUERY "+q1.toString());
4
5  dq = new DrillDownQuery(fconfig,q1 );
6  dq.add("Centro", "ETSII");
7  System.out.println("    Filtramos query[ "+ dq.toString() +" ]")
       ;
8
9  DrillSideways ds = new DrillSideways(searcher, fconfig,
       taxoReader);
10 DrillSidewaysResult dsresult = ds.search(dq,10);
11 System.out.println("dsw  hits "+ dsresult.hits.totalHits);
12 System.out.println(dsresult.facets.getAllDims(10).toString());
13
14 for (ScoreDoc scoreDoc : dsresult.hits.scoreDocs) {
15    doc = searcher.doc(scoreDoc.doc);
16    System.out.println("       Docs Score-> " + scoreDoc.score +
          "::"   + doc.get("asignatura"));
17 }
```

6. Gestionando los árboles de facetas

En principio, todas las facetas son añadidas a un único índice, con nombre por defecto $facets. Cada faceta de las añadidas es un nodo en dicho árbol. En cualquier caso, Lucene permite separar las facetas en distintos árboles, pudiendo tratar cada uno de ellos por separado, lo que permite que en tiempo de búsqueda nos podamos centrar un un subconjunto de las facetas de nuestro interés. Para

ello, debemos de configurar las facetas de forma adecuada utilizando el método `setIndexFieldName` que recibe el nombre de la faceta y el nombre del índice (para las facetas que no se indica nada, se ubican dentro del índice $facets)

Podemos pensar que en grandes tiendas de comercio electrónico, con muchos departamentos distintos, las facetas de cada departamento se almacenasen en su propio árbol de facetas, que sería el que se utilizaría en función de los resultados de la búsqueda realizada por el usuario.

```
1  FacetsConfig fconfig = new FacetsConfig();
2  fconfig.setIndexFieldName("Titulacion", "facet_tit"); // Faceta
       Titulacion la gestiona dentro del factet_tit
3  fconfig.setIndexFieldName("Alumnos", "facet_alumnos"); // Faceta
       alumnos la gestiona facet_alumnos
4
5  fconfig.setHierarchical("Titulacion", true);
6  ...
7
8  Document doc = new Document();
9  doc.add(new FacetField("Centro", "ETSII")); //a $facets
10 doc.add(new FacetField("Cuatrimestre", "primero")); //a $facets
11 doc.add(new FacetField("Titulacion", "Grado en Informatica",
12                                      "Sistemas de Informacian",
13                                      "Obligatoria")); //a
                                           facet_tit
14 doc.add(new NumericDocValuesFacetField("alumnos", 35L));  // a
       facet_alumnos
```

En este caso, en tiempo de búsqueda podemos restringir el cálculo de las facetas a uno de los índices, indicándole al método que realiza los conteos con el que se desea trabajar, por defecto, sin no se indica nada, es $facets.

```
1  // Para buscar en $facets
2  Facets facets = new FastTaxonomyFacetCounts(taxoReader, fconfig,
       fcollector);
3  // Para buscar en "facet_tit"
4  Facets ftit = new FastTaxonomyFacetCounts("facet_tit",
       taxoReader, fconfig, fcollector);
5
6  // Listamos categorias de nivel Titulacion
```

```
7 System.out.println(ftit.getTopChildren(3, "Titulacion").toString
    ());
8 // Listamos categorias de nivel Titulacion/GradoEnInformatica
9 System.out.println(ftit.getTopChildren(3, "Titulacion", "Grado
    en Informatica").toString());
```

7. Entrega de la práctica

Se deberá seleccionar varios campos para ilustrar el uso de categorías sobre las consultas que se han diseñado. También se debe considerar el uso de categorías por rango. Esta información deberá ser proporcionada al usuario, permitiendo realizar consultas DrillDown

Una vez completada la práctica se deberá generar una documentación completa sobre el problema concreto que se está resolviendo considerando el tipo de consultas que se permiten así como los criterios utilizados para garantizar devolver a los usuarios los elementos más relevantes (descripción detallada de los campos utilizados y el procesamiento que se tiene de cada uno de ellos). Se incluirá un manual de usuario para las dos aplicaciones (indexación y búsqueda).

Adicionalmente se deberán entregar los ficheros .java originales que deberán ser convenientemente documentados así como un fichero donde se indiquen las dependencias del proyecto.

7.1. Fecha de Entrega

La fecha de entrega del proyecto, inamovible, es el martes 12 de Diciembre. A partir del jueves 14 se procederá a realizar la defensa de las prácticas, siendo en este caso **necesaria** la presencia del estudiante para realizar dicha defensa.