# Búsqueda con Lucene

Sergio Samaniego Martínez

# Índice


- Introducción
- Explicación del código
- Ejemplos de uso

# Introducción

En esta segunda parte de la práctica de búsqueda con Lucene, se va a proceder a realizar el proceso de búsqueda sobre un índice ya creado de Lucene.
Para facilitar el uso de nuestro programa, se creará una interfaz de usuario que permitirá al usuario realizar distintos tipos de búsquedas (libre, booleana y por rango), poder visualizar los resultados sobre una tabla y aplicar filtros de facetas sobre dichos resultados.

# Explicación de código

Tendremos 3 clases fundamentales:

- Sample.fxml : Este archivo será el encargado de introducir y definir los distintos elementos que se van a usar en la interfaz gráfica.
- Main.class : Esta clase será la encargada de inicializar la la ventana de la interfaz gráfica cargando el contenido de sample.fxml
- Controller.class : Esta clase será la encargada de controlar y realizar las distintas acciones que la aplicación va a realizar.

# Sample.fxml

```
<children>
  <Label fx:id="searchlabel" layoutX="60" layoutY="30">Write the free search: </Label>
  <TextField fx:id="textfield" layoutX="50.0" layoutY="50.0"  prefHeight="0" prefWidth="200" />
  <Label fx:id="fieldlabel" layoutX="260" layoutY="30">Select the field search: </Label>
  <ComboBox fx:id="fieldbox" layoutX="260.0" layoutY="50.0" prefWidth="150.0" promptText="choose">
      <items>
          <FXCollections fx:factory="observableArrayList" >
              <String fx:value="All" />
              <String fx:value="Author" />
              <String fx:value="Title" />
              <String fx:value="Year" />
              <String fx:value="Source Title" />
              <String fx:value="Link" />
              <String fx:value="Resume" />
              <String fx:value="AuthorKeywords" />
              <String fx:value="IndexKeywords" />
              <String fx:value="EID" />
          </FXCollections>
      </items>
  </ComboBox>
  <Button fx:id="btn" layoutX="420.0" layoutY="50.0" mnemonicParsing="false" onAction="#btn1handle" text="Button" />
```

# Main.class

```
public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
        primaryStage.setTitle("Information Retrieval Search");
        primaryStage.setScene(new Scene(root, 1500, 1000));

        primaryStage.show();
    }


    public static void main(String[] args) { launch(args); }

}
```

# Controller.class

Veremos los distintos métodos que componen esta clase como son:

- BtnHandler
- Métodos de las distintas búsquedas
- addFacets
- FilterAction

# btnHandler

```
@FXML
void btn1handle(ActionEvent event) throws IOException {
    tabledocs.setEditable(true);

    //Obtenemos el id del botón que ha lanzado el evento
    String clicked = ((Control)event.getSource()).getId();

    author_col.setCellValueFactory(new PropertyValueFactory<>( s: "Author"));
    title_col.setCellValueFactory(new PropertyValueFactory<>( s: "Title"));
    year_col.setCellValueFactory(new PropertyValueFactory<>( s: "Year"));
    source_col.setCellValueFactory(new PropertyValueFactory<>( s: "Source_Title"));
    link_col.setCellValueFactory(new PropertyValueFactory<>( s: "Link"));
    authorkey_col.setCellValueFactory(new PropertyValueFactory<>( s: "AuthorKeywords"));
    indexkey_col.setCellValueFactory(new PropertyValueFactory<>( s: "IndexKeywords"));


    //Dependiendo del botón que haya sido clickado realizaremos una función determinada

    if(clicked.equals("btn")) {

        search = textfield.getText();
        if (fieldbox.getValue() == null)
            field_select = "All";
        else
            field_select = fieldbox.getValue().toString();

        doSearch();
    }else if(clicked.equals("btnbool")) {
        search = textfieldbool.getText();
        if (fieldboxbool.getValue() == null)
            field_select = "All";
        else
            field_select = fieldboxbool.getValue().toString();
        doSearchBoolean();
    }else {
        search = textfieldint.getText();
        if (fieldboxint.getValue() == null)
            field_select = "All";
        else
            field_select = fieldboxint.getValue().toString();
        doIntSearch();
    }
}
```

# doSearch

Con un array que contenga los distintos tokens, se recorre para ir añadiendo a bqbuilder los distintos tokens de la consulta realizada

```
    try {
        Path p1 = Paths.get(index_path);
        FSDirectory dir = FSDirectory.open(p1);
        IndexReader ireader = DirectoryReader.open(dir);
        IndexSearcher searcher = new IndexSearcher(ireader);


        //Al ser una consulta libre, lo que hacemos es que todos los tokens deben aparecer

        BooleanQuery.Builder bqbuilder = new BooleanQuery.Builder();
        for(int i=0; i<result.size(); i++) {
            TermQuery q1 = new TermQuery(new Term(field, result.get(i)));
            BooleanClause bc1 = new BooleanClause(q1, BooleanClause.Occur.MUST);

            bqbuilder.add(bc1);

        }
        BooleanQuery bq = bqbuilder.build();
        this.bq = bq;

        TopDocs tdocs = searcher.search(bq,  n: 50);

        System.out.println("Hay "+tdocs.totalHits+" docs");

        ScoreDoc[] hits = tdocs.scoreDocs;

        data.clear();

        for(int i=0; i<hits.length; i++){
            org.apache.lucene.document.Document hitDoc = searcher.doc(hits[i].doc);

            data.add(new IndexDocs(hitDoc.get("author").toString(),
                    hitDoc.get("title").toString(),
                    hitDoc.get("year_s").toString(),
                    hitDoc.get("source_title").toString(),
                    hitDoc.get("link").toString(),
                    hitDoc.get("AuthorKeywords").toString(),
                    hitDoc.get("IndexKeywords").toString()));

            //System.out.println(hitDoc.get("title").toString());
        }

        tabledocs.setItems(data);

        addFacets(bq);

    }
    catch (IndexNotFoundException e){
        System.out.println("No se ha podido encontrar el indice en el directorio indicado");
    }
}
```

# doIntSearch

Al realizar la consulta se tendrán que separar los dos extremos por “,”.
Con esto hecho únicamente tenremos que pasarle ambos extremos al nuevo newRangeQuery

```
try {
    Path p1 = Paths.get(index_path);
    FSDirectory dir = FSDirectory.open(p1);
    IndexReader ireader = DirectoryReader.open(dir);
    IndexSearcher searcher = new IndexSearcher(ireader);
    Integer startpoint = Integer.valueOf(result.get(0));
    Integer endpoint = Integer.valueOf(result.get(result.size()-1));

    if(startpoint > endpoint){
        Integer aux = startpoint;
        startpoint = endpoint;
        endpoint = aux;
    }

    Query bq = IntPoint.newRangeQuery(field, startpoint, endpoint);


    TopDocs tdocs = searcher.search(bq,  n: 50);

    System.out.println("Hay "+tdocs.totalHits+" docs");

    ScoreDoc[] hits = tdocs.scoreDocs;

    data.clear();

    for(int i=0; i<hits.length; i++){
        org.apache.lucene.document.Document hitDoc = searcher.doc(hits[i].doc);

        data.add(new IndexDocs(hitDoc.get("author").toString(),
                hitDoc.get("title").toString(),
                hitDoc.get("year_s").toString(),
                hitDoc.get("source_title").toString(),
                hitDoc.get("link").toString(),
                hitDoc.get("AuthorKeywords").toString(),
                hitDoc.get("IndexKeywords").toString()));

        //System.out.println(hitDoc.get("title").toString());
    }

    tabledocs.setItems(data);


}
catch (IndexNotFoundException e){
    System.out.println("No se ha podido encontrar el índice en el directorio indicado");
}
```

# doBooleanSearch

- Tendremos que ser capaces de identificar los tres operadores AND, OR y NOT en nuestra consulta.
- Puesto que utilizamos diferentes analizadores, para cada tipo de campo, será necesario tratar las consultas de formas diferentes dependiendo del Analyzer utilizado

# WhiteSpaceAnalyzer

```
try {
    TokenStream stream  = analyzer.tokenStream( fieldName: null,  new StringReader(search));
    OffsetAttribute offsetAtt = stream.addAttribute(OffsetAttribute.class);
    CharTermAttribute cAtt= stream.addAttribute(CharTermAttribute.class);
    stream.reset();
    String cad ="";
    Integer offsetfinal = 0;
    while (stream.incrementToken()) {

        if(cAtt.toString().toLowerCase().equals("and")){
            booleans.add("AND");
            result.add(cad);
            cad = "";
        }
        else if(cAtt.toString().toLowerCase().equals("or")){
            booleans.add("OR");
            result.add(cad);
            cad = "";
        }
        else if(cAtt.toString().toLowerCase().equals("not")){
            booleans.add("NOT");
            result.add(cad);
            cad = "";
        }
        else{
            cad += cAtt.toString()+" ";
        }

    }

    result.add(cad);
    stream.end();
} catch (IOException e) {
    // not thrown b/c we're using a string reader...
    throw new RuntimeException(e);
}


break;
```

# Construcción de la búsqueda booleana

Con un array que contenga los operadores utilizados, lo iremos recorriendo para conocer de qué tipo va a ser la ocurrencia para cada uno de los tokens

```
//Con la consulta tokenizada, pasamos a realizar las consultas
try {
    Path p1 = Paths.get(index_path);
    FSDirectory dir = FSDirectory.open(p1);
    IndexReader ireader = DirectoryReader.open(dir);
    IndexSearcher searcher = new IndexSearcher(ireader);

    BooleanQuery.Builder bqbuilder = new BooleanQuery.Builder();

    //Como tenemos almacenados los operadores utilizados, dependiendo del operador que estemos leyendo
    // usaremos un BooleanQuery.Occur diferente
    for(int i=0; i<result.size(); i++) {
        TermQuery q1 = new TermQuery(new Term(field, result.get(i).trim()));
        BooleanClause bc1;

        if(i == 0){
            bc1 = new BooleanClause(q1, BooleanClause.Occur.MUST);
        }
        else {
            if ((booleans.get(i-1)).equals("AND"))
                bc1 = new BooleanClause(q1, BooleanClause.Occur.MUST);
            else if ((booleans.get(i-1)).equals("OR"))
                bc1 = new BooleanClause(q1, BooleanClause.Occur.SHOULD);
            else
                bc1 = new BooleanClause(q1, BooleanClause.Occur.MUST_NOT);
        }

        bqbuilder.add(bc1);

    }
    BooleanQuery bq = bqbuilder.build();
    this.bq = bq;

    TopDocs tdocs = searcher.search(bq,  n: 50);

    System.out.println("Hay "+tdocs.totalHits+" docs");

    ScoreDoc[] hits = tdocs.scoreDocs;

    data.clear();
```

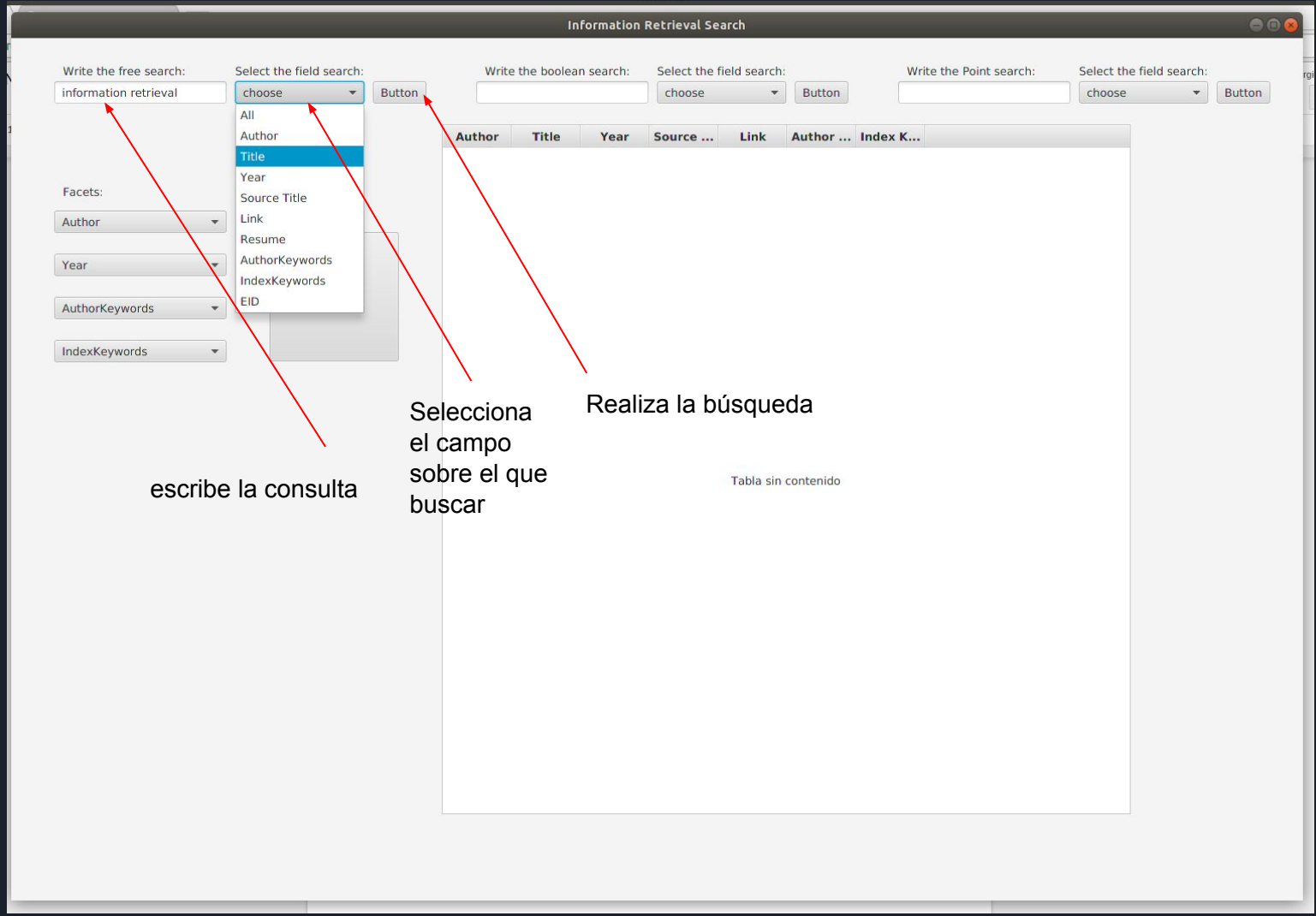
# AddFacets

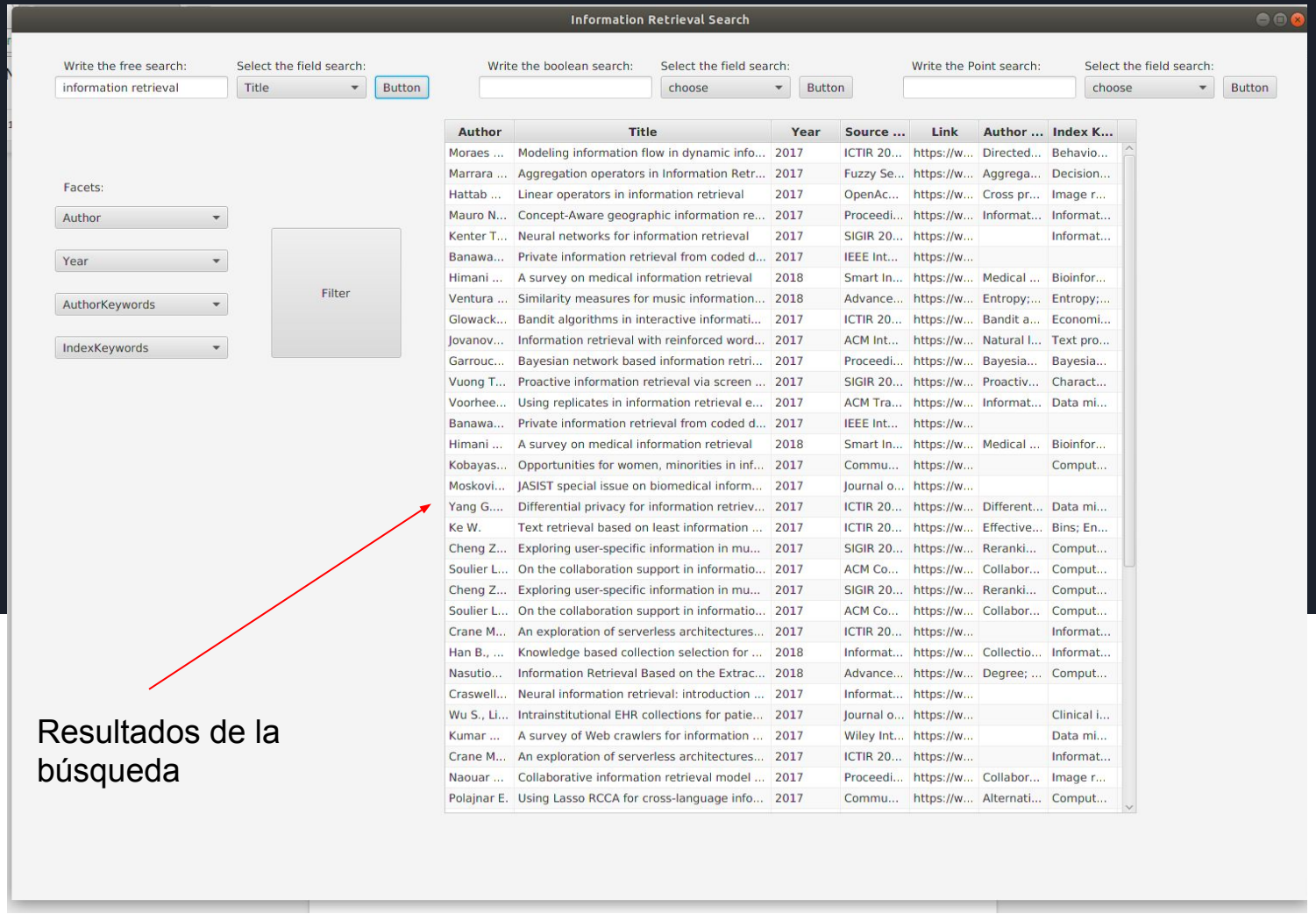
```
    fconfig.setHierarchical("Index Keywords", true);


    //Creamos FacetsCollector que encontrará las facetas de la búsqueda
    FacetsCollector fc = new FacetsCollector();
    TopDocs tdc = FacetsCollector.search(searcher, bq, n: 10, fc);

    for (ScoreDoc sd : tdc.scoreDocs){
        Document d = searcher.doc(sd.doc);
        //System.out.println(sd.score + d.get("title"));
    }


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

    List<FacetResult> TodasDims = facetas.getAllDims( i: 100);
    //System.out.println("Categorias totales " + TodasDims.size());

    AuKeyComboBox.getItems().clear();
    AuthorComboBox.getItems().clear();
    InKeyComboBox.getItems().clear();
    YearComboBox.getItems().clear();

    //Cada una de las dimensiones de la faceta se almacenará en un combobox que nos permitirá el filtrado posterior
    for(FacetResult fr : TodasDims){
        //System.out.println("Categoria " + fr.dim);
        LabelAndValue[] lv = fr.labelValues;

        switch (fr.dim){
            case "Author Keywords":
                for(int i=0; (i<20 && i< fr.labelValues.length); i++){
                    AuKeyComboBox.getItems().add(lv[i].label);
                }
                break;
            case "Authors":
                for(int i=0; (i<20 && i< fr.labelValues.length); i++){
                    AuthorComboBox.getItems().add(lv[i].label);
                }
                break;
            case "Index Keywords":
                for(int i=0; (i<20 && i< fr.labelValues.length); i++){
                    InKeyComboBox.getItems().add(lv[i].label);
                }
                break;
            case "Year":
                for(int i=0; (i<20 && i< fr.labelValues.length); i++){
                    YearComboBox.getItems().add(lv[i].label);
                }
                break;
        }
    }

    FacetResult fresult = facetas.getTopChildren( i: 10, s: "title");

}
```

# filterAction

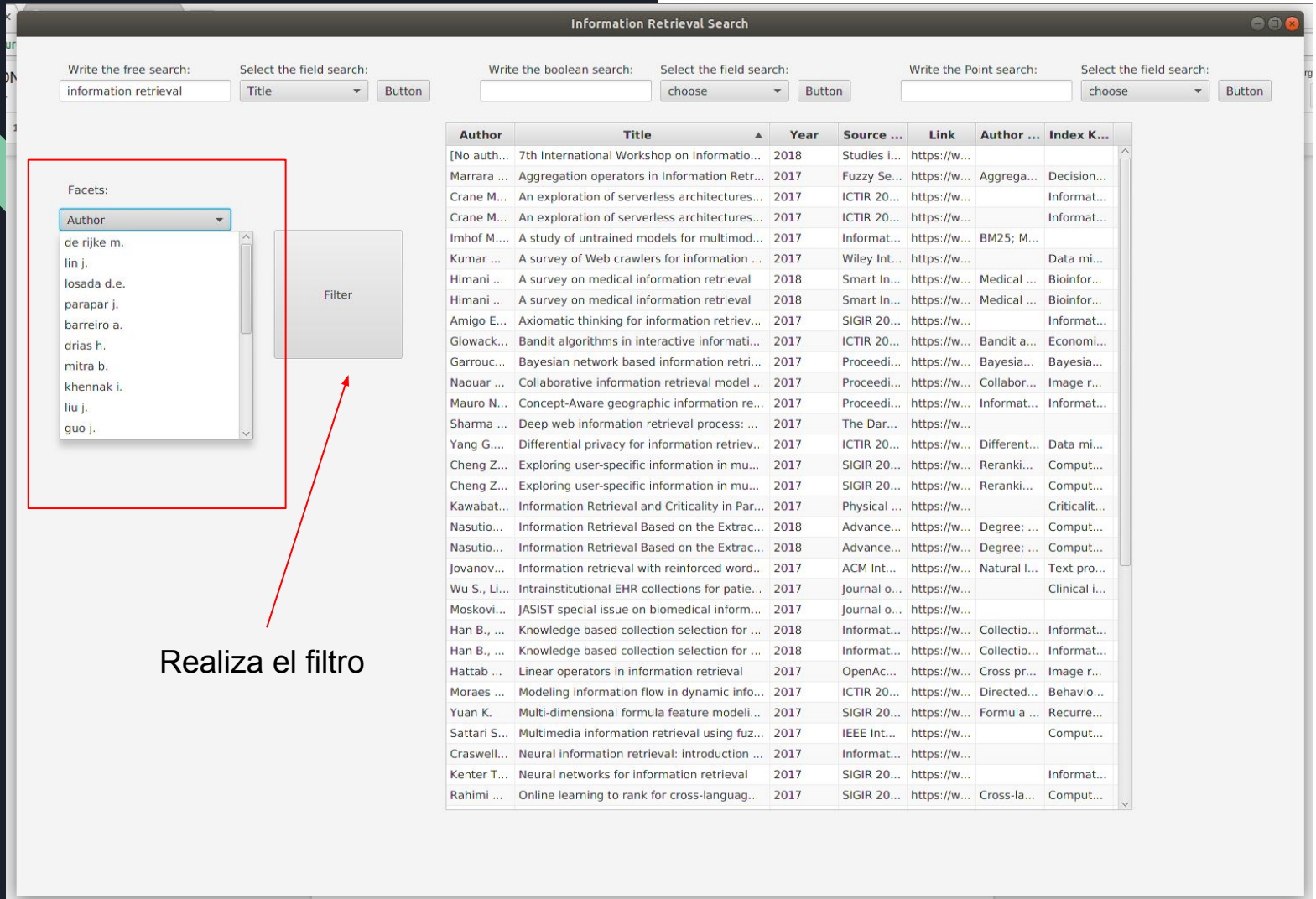
- La función se llamará al pulsar el botón **Filter** de la Interfaz.
- Esta función lo único que va a realizar será crear un DrillDownQuery y un DrillSideWays que se ejecutarán sobre la consulta realizada para aplicar los filtros impuestos.

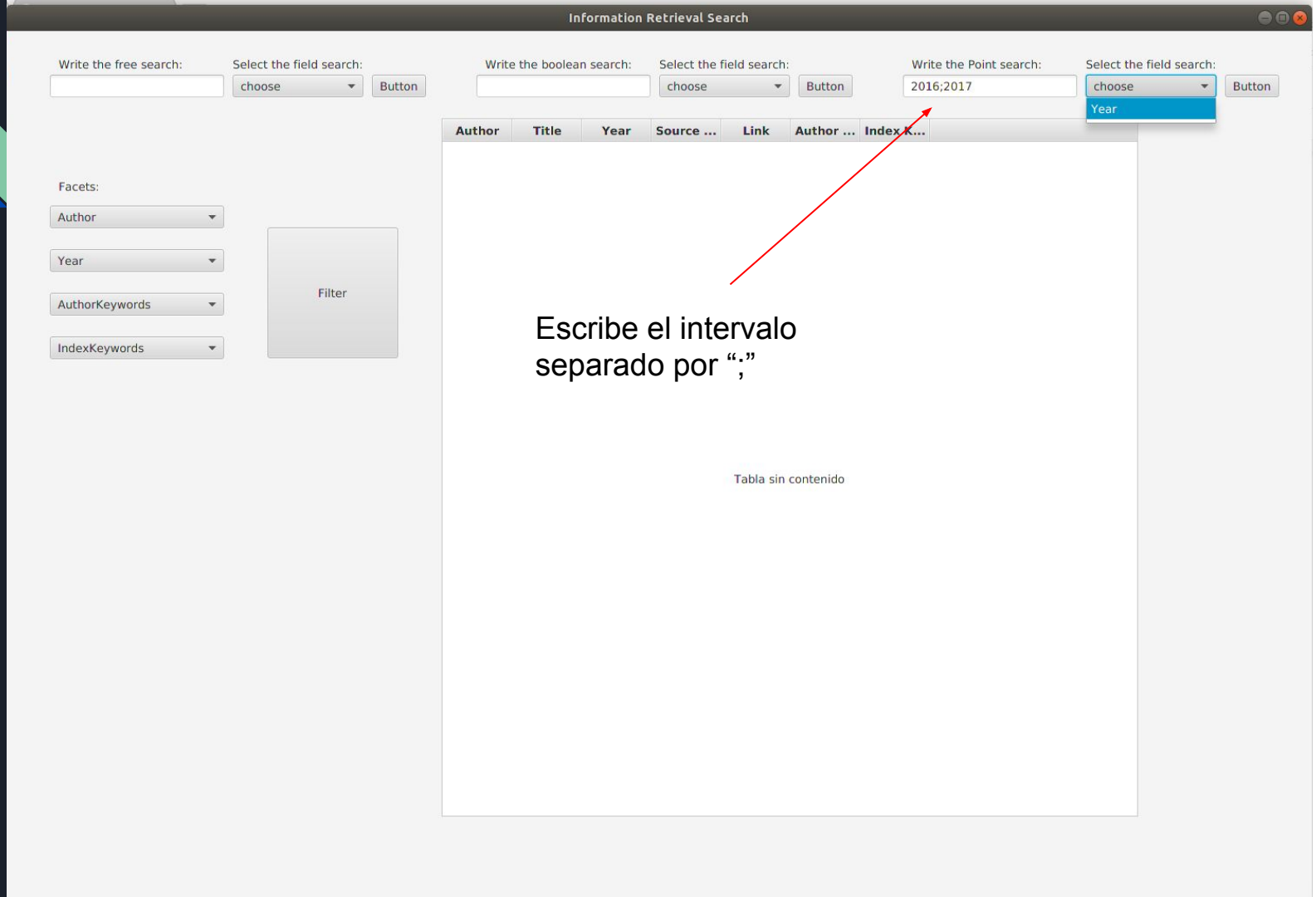
# Manual de usuario

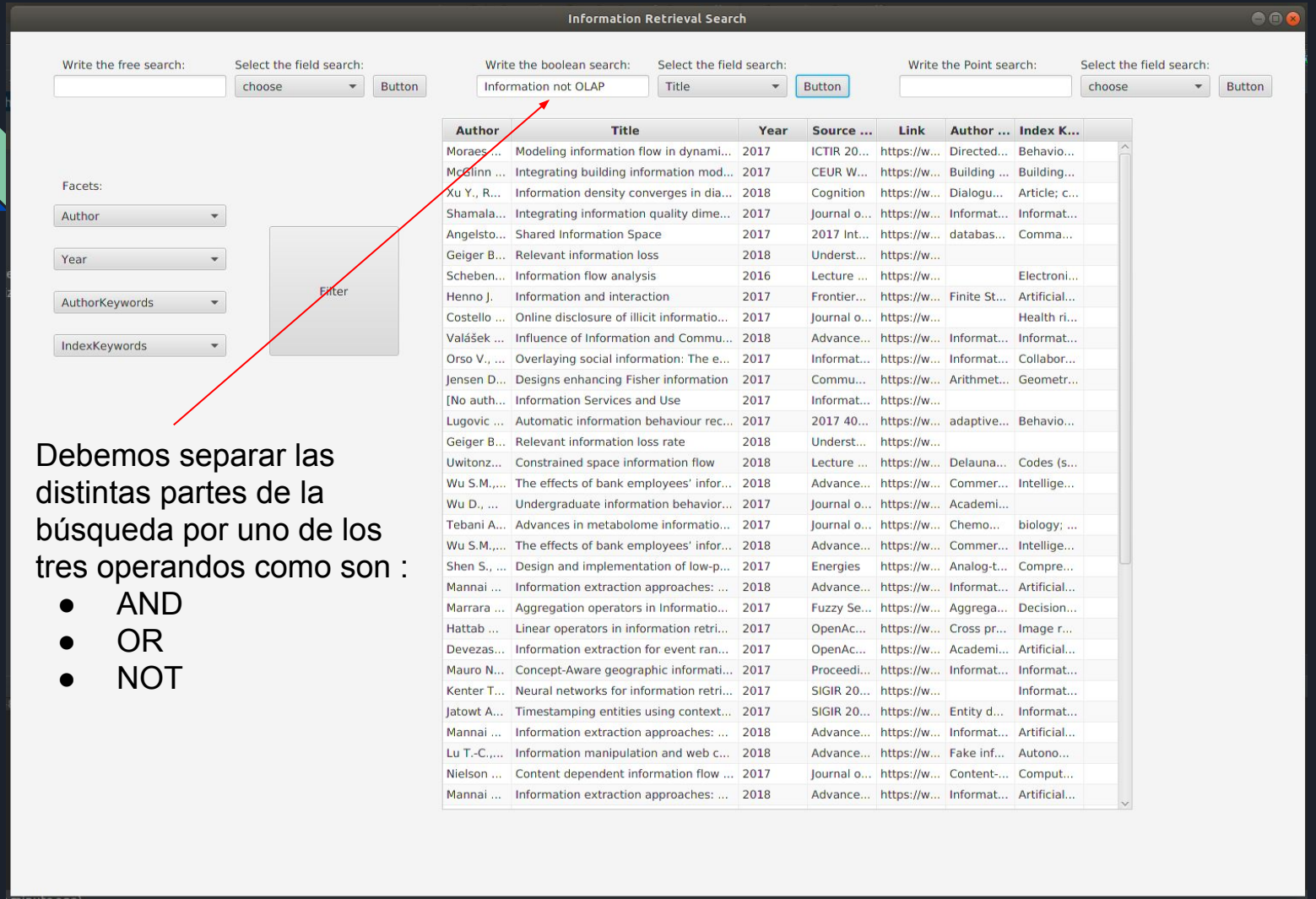

Information Retrieval Search
Write the free search:
information retrieval
Select the field search:
choose
All
Author
Title
Year
Source Title
Link
Resume
AuthorKeywords
IndexKeywords
EID
Button
Write the boolean search:
Select the field search:
choose
Button
Write the Point search:
Select the field search:
choose
Button
Facets:
Author
Year
AuthorKeywords
IndexKeywords
Author
Title
Year
Source ...
Link
Author ...
Index K...
Tabla sin contenido
escribe la consulta
Selecciona el campo sobre el que buscar
Realiza la búsqueda




Resultados de la búsqueda


Information Retrieval Search
Write the free search:
information retrieval
Select the field search:
Title
Button
Write the boolean search:
Select the field search:
choose
Button
Write the Point search:
Select the field search:
choose
Button
Facets:
Author
de rijke m.
lin j.
losada d.e.
parapar j.
barreiro a.
drias h.
mitra b.
khennak i.
liu j.
guo j.
Filter
Author
Title
Year
Source ...
Link
Author ...
Index K...
[No auth... 7th International Workshop on Informatio... 2018 Studies i... https://w...
Marrara ... Aggregation operators in Information Retr... 2017 Fuzzy Se... https://w... Aggrega... Decision...
Crane M... An exploration of serverless architectures... 2017 ICTIR 20... https://w... Informat...
Crane M... An exploration of serverless architectures... 2017 ICTIR 20... https://w... Informat...
Imhof M.... A study of untrained models for multimod... 2017 Informat... https://w... BM25; M...
Kumar ... A survey of Web crawlers for information ... 2017 Wiley Int... https://w... Data mi...
Himani ... A survey on medical information retrieval 2018 Smart In... https://w... Medical ... Bioinfor...
Himani ... A survey on medical information retrieval 2018 Smart In... https://w... Medical ... Bioinfor...
Amigo E... Axiomatic thinking for information retriev... 2017 SIGIR 20... https://w... Informat...
Glowack... Bandit algorithms in interactive informati... 2017 ICTIR 20... https://w... Bandit a... Economi...
Garrouc... Bayesian network based information retri... 2017 Proceedi... https://w... Bayesia... Bayesia...
Naouar ... Collaborative information retrieval model ... 2017 Proceedi... https://w... Collabor... Image r...
Mauro N... Concept-Aware geographic information re... 2017 Proceedi... https://w... Informat... Informat...
Sharma ... Deep web information retrieval process: ... 2017 The Dar... https://w...
Yang G.... Differential privacy for information retriev... 2017 ICTIR 20... https://w... Different... Data mi...
Cheng Z... Exploring user-specific information in mu... 2017 SIGIR 20... https://w... Reranki... Comput...
Cheng Z... Exploring user-specific information in mu... 2017 SIGIR 20... https://w... Reranki... Comput...
Kawabat... Information Retrieval and Criticality in Par... 2017 Physical ... https://w... Criticalit...
Nasutio... Information Retrieval Based on the Extrac... 2018 Advance... https://w... Degree; ... Comput...
Nasutio... Information Retrieval Based on the Extrac... 2018 Advance... https://w... Degree; ... Comput...
Jovanov... Information retrieval with reinforced word... 2017 ACM Int... https://w... Natural l... Text pro...
Wu S., Li... Intrainstitutional EHR collections for patie... 2017 Journal o... https://w... Clinical i...
Moskovi... JASIST special issue on biomedical inform... 2017 Journal o... https://w...
Han B., ... Knowledge based collection selection for ... 2018 Informat... https://w... Collectio... Informat...
Han B., ... Knowledge based collection selection for ... 2018 Informat... https://w... Collectio... Informat...
Hattab ... Linear operators in information retrieval 2017 OpenAc... https://w... Cross pr... Image r...
Moraes ... Modeling information flow in dynamic info... 2017 ICTIR 20... https://w... Directed... Behavio...
Yuan K. Multi-dimensional formula feature modeli... 2017 SIGIR 20... https://w... Formula ... Recurre...
Sattari S... Multimedia information retrieval using fuz... 2017 IEEE Int... https://w... Comput...
Craswell... Neural information retrieval: introduction ... 2017 Informat... https://w...
Kenter T... Neural networks for information retrieval 2017 SIGIR 20... https://w... Informat...
Rahimi ... Online learning to rank for cross-languag... 2017 SIGIR 20... https://w... Cross-la... Comput...
Realiza el filtro


Information Retrieval Search
Write the free search:
Select the field search:
choose
Button
Write the boolean search:
Select the field search:
choose
Button
Write the Point search:
2016;2017
Select the field search:
choose
Year
Button
Author
Title
Year
Source ...
Link
Author ...
Index K...
Facets:
Author
Year
AuthorKeywords
IndexKeywords
Filter
Escribe el intervalo separado por “;”
Tabla sin contenido




Debemos separar las distintas partes de la búsqueda por uno de los tres operandos como son :

- AND
- OR
- NOT

# ¡Gracias!