Text Search Application with Lucene

Demonstration of Application

How does it work

I. Corpus: Data for the Searching

1. Wikipedia: XML dump for english
2. converting into xml format and chunking into text files

II. Lucene Framework for

1. Building an Index
2. Searching in the Index

II. User Interface with java swing

Corpus

Wikipedia artickles for German:

- Wikipedia XML dump: https://dumps.wikimedia.org/dewiki/20160601/

Creating of the text files from the XML dump:

- Converting into xml
- Chunking into the text files

Creating of the text files from XML dump

1. Converting into xml with Wikiforia:
   - Clone or download repository of Wikiforia
   - Change into dist/ directory and run
   - java -jar wikiforia-1.0-SNAPSHOT.jar

     -pages [path to the file ending with multistream.xml.bz2]

     -output [output.xml]

2. Chunking into the text files: WikipediaXMLParser.java

Lucene: Build an Index

4 Steps to build an Index:

1. Build an IndexWriter Instance
2. Indexing a Directory: Loop through the all files to be indexed
3. Create the Document Instance for each document (define their field(s))
4. Add the document content to the index

Build an Index - Step 1: Build an IndexWriter Instance

Creating an **IndexWriter** Instance with 2 Arguments:

1. **Directory:** Name of directory, where the Index will be created
2. **StandartAnalyzer:** tokenizing, lower case, stop words...

```
Directory dir = FSDirectory.open(Paths.get(indexPath));
Analyzer analyzer = new StandardAnalyzer();
IndexWriterConfig config = new IndexWriterConfig(analyzer);
IndexWriter writer = new IndexWriter(dir, config);
```

Build an Index - Step 2: Indexing a Directory

Loops through the all files to be indexed and puts the file content to the Lucene-Document or updates it :

```
if (Files.isDirectory(docDir)) {
    // Index all files in directory
    File[] files = new File(docDir.toString()).listFiles();
    for (File f : files) {
        indexDoc(writer, f.toPath()); }
} else {  indexDoc(writer, docDir); }
```

Build an Index - Step 3: Create the Document Instance

Create the Document Instance for each file and define their field(s) (each document has at least one field):

```
Document doc = new Document();

Field pathField = new StringField("path", docDir.toString(),
Field.Store.YES);

doc.add(pathField);

doc.add(new TextField("contents",

new BufferedReader(new InputStreamReader(stream,
StandardCharsets.UTF_8))));
```

Build an Index - Step 4: Adding document content to the index

If it is new index, so we just add the document:

```
if (writer.getConfig().getOpenMode() == OpenMode.CREATE) {
    System.out.println("adding " + docDir);
    writer.addDocument(doc); }
```

If it is existing index, so we use update Document:

```
else {
    System.out.println("updating " + docDir);
    writer.updateDocument(new Term("path", docDir.toString()), doc); }
```

How does look like an Index?

Inverted Index:

Word1: DocId1 → DocId3 → DocId5 → ...

Word2: DocId6 → DocId70 → DocId89 → …

…

Wordn: DocId1 → DocId8 → DocIdn ...

Lucene: Search in the Index

3 Steps for Searching in the Index:

1. Build an **IndexSearcher Instance** (for Searching in the Index)
2. Build **QueryParser**
3. **Return** the matching Documents (**Hits**)

Search in the Index – Step 1: Build an IndexSearcher Instance

Build an IndexSearcher Instance:

1. Open the directory to the index
2. Creat the searcher for this index

```
IndexReader reader =
DirectoryReader.open(FSDirectory.open(Paths.get(index)));
IndexSearcher searcher = new IndexSearcher(reader);
```

Search in the Index – Step 2: Build a QueryParser

Build QueryParser with two arguments:

1. field: where must be searched
2. analyzer: same as in the IndexWriter

```
Analyzer analyzer = new StandardAnalyzer();
QueryParser parser = new QueryParser(field, analyzer);
Query query = parser.parse(input);
```

Search in the Index – Step 3: Returning matching Hits

1. Builds the Document Collector and puts all matching documents in the collector

2. Ranks and outputs the dokuments

```
int hitsPerPage = 10;

TopScoreDocCollector collector =
TopScoreDocCollector.create(hitsPerPage);

searcher.search(query, collector);

ScoreDoc[] hits = collector.topDocs().scoreDocs;

System.out.println(hits.length + " Documents founded.");
```

Grafical User Interface

- Simple GUI with java swing.
- Gets the Search String and calls the indexSercher() method.
- Output the results and give the option for viewing the text file content.