Information Retrieval

Assignment 3:
Boolean Information Retrieval with Lucene

Patrick Schäfer (patrick.schaefer@hu-berlin.de)

Marc Bux (buxmarcn@informatik.hu-berlin.de)

Lucene

- Open source, Java-based information retrieval engine.
- Requires two steps:
  - Indexing: Create a Lucene index on the documents.
  - Searching: Parse a query and lookup the index.

- Different flavors & extensions: Solr, Nutch, Tika ...
- But you have to use the java Lucene core library.

Task

- Implement Boolean IR as in assignment 2, but this time use Lucene:
  - Parse the IMDB movie plots (reuse existing code).
  - Treat all text in lowercase for indexing and searching.
  - Support the full Lucene Query Parser Syntax[1].
  - Use word tokenization and stop word removal, but no stemming.

- You can use your movie plot file parser from Assignment 2.
- But use Lucene (v6.3)[2] for indexing and searching!

[1] https://lucene.apache.org/core/6_3_0/queryparser/org/apache/lucene/queryparser/classic/package-summary.html#package.description

[2] http://lucene.apache.org/core/

Example Queries

1. title:"game of thrones" AND type:episode AND (plot:Bastards OR (plot:Jon AND plot:Snow)) -plot:son
2. title:"Star Wars" AND type:movie AND plot:Luke AND year:[1977 TO 1987]
3. plot:Berlin AND plot:wall AND type:television
4. plot:men~1 AND plot:women~1 AND plot:love AND plot:fool AND type:movie
5. title:westworld AND type:episode AND year:2016 AND plot:Dolores
6. plot:You AND plot:never AND plot:get AND plot:A AND plot:second AND plot:chance
7. plot:Hero AND plot:Villain AND plot:destroy AND type:movie
8. (plot:lover -plot:perfect) AND plot:unfaithful* AND plot:husband AND plot:affair AND type:movie
9. (plot:Innocent OR plot:Guilty) AND plot:crime AND plot:murder AND plot:court AND plot:judge AND type:movie
10. plot:Hero AND plot:Marvel -plot:DC AND type:movie
11. plot:Hero AND plot:DC -plot:Marvel AND type:movie

Basic Concepts in Lucene

- Lucene builds inverted indices and allows queries on these indices.
- A Document is the unit of search and index.
- Indexing involves adding documents to an IndexWriter.
- Searching involves retrieving documents via an IndexSearcher.
- A document consists of one or more fields.
- A field is a key-value pair.

Understanding Concepts in Lucene

- Tokenizers: break field data into lexical units, or tokens.
- Filters: examine a stream of tokens and keep them, transform or discard them, or create new ones.
- Analyzers: tokenizers and filters may be combined. This combination is called an analyzer. The output of an analyzer is used to query or build indices.

- Use the same analyzer for querying and building indices.

Lucene Analyzers

- Lucene provids multiple default tokenizers, i.e.:
  - LetterTokenizer: divide text at non-characters.
  - WhiteSpaceTokenizer: divide text at whitespace characters.
  - StandardTokenizer: grammar-based tokenizer.

- Lucene provides multiple default filters. i.e.:
  - LowerCaseFilter: converts any uppercase letters to lowercase.
  - Word Stemming filters (Kstem, Hunspell, Snowball Porter, …)

- Lucene provides multiple default analyzers, i.e.:
  - SimpleAnalyzer: LetterTokenizer, LowerCaseFilter.
  - StandardAnalyzer: StandardTokenizer, LowerCaseFilter, English stop words
  - WhiteSpaceAnalyzer: WhiteSpaceTokenizer.
  - StopAnalyzer: LetterTokenizer, LowerCaseFilter, English stop words.

- You just have to use the corresponding analyzer (compare slide 3&6).

Indexing in Lucene

- Specify the analyzer to use

```
Analyzer myAnalyzer = …;
```

- Specify a directory and an index writer

```
Directory index = FSDirectory.open(new File(directory).toPath());
IndexWriterConfig config = new IndexWriterConfig(myAnalyzer);
IndexWriter writer = new IndexWriter(index, config);
```

- Create a document and add this document to the index:

```
Document doc = new Document();
doc.add(new StringField(“id”, id, StringField.Store.YES));
doc.add(new TextField(“title”, title, TextField.Store.YES));
writer.addDocument(doc);
```

- Close Index writer:

```
writer.commit()
writer.close();
```

StringField and TextField

- StringField vs TextField:
  - TextFields will be tokenized. Used for texts that needs to be tokenized.
  - StringFields will be treated as a single term. Used for atomic values that are not to be tokenized.

- Many other typed fields:
  - IntPoint: int indexed for exact/range queries.
  - LongPoint: long indexed for exact/range queries.
  - FloatPoint: float indexed for exact/range queries.
  - DoublePoint: double indexed for exact/range queries.
  - …

- Field.Store.YES : indexed & returned as result.
- Field.Store.NO : indexed but not returned as result.

Querying in Lucene

- Open Lucene index for searching

```
IndexReader indexReader = DirectoryReader.open(index);
IndexSearcher indexSearcher = new IndexSearcher(indexReader);
```

- Parse title:”querystr” using the analyzer

```
Query q = new QueryParser("title", myAnalyzer).parse(querystr);
```

- Retrieve all results

```
TopDocs hits = indexSearcher.search(q, Integer.MAX_VALUE);
```

Revisited: Query Syntax

- You have to support the Query Parser syntax[1]:
  - term query syntax:

```
title:Game
```

  - phrase query syntax:

```
title:”Game of Thrones”
```

  - AND query, OR query

```
title:”Game of Thrones” AND (plot:Baelish OR plot:Jon)
```

  - NOT queries
  - Wildcards
  - Proximity
  - Range searches
  - …
- There is a built-in Query Parser for this in Lucene.

[1] http://www.lucenetutorial.com/lucene-query-syntax.html

Revisited: Query Syntax

- Searchable fields are as follows:
  - title
  - plot (if a document has multiple plot descriptions they can be appended)
  - type (movie, series, episode, television, video, videogame; see next slide)
  - year (optional)
  - episodetitle (optional, only for episodes)

- There is a built-in MultiFieldQueryParser[1] for this in Lucene.

[1]https://lucene.apache.org/core/6_3_0/queryparser/org/apache/lucene/queryparser/classic/MultiFieldQueryParser.html

Revisited: Corpus

- Reuse the corpus plot.list[1]
  - plain text, roughly 400 MB, updated version every Friday.
  - you can reduce the size of the corpus by using the HEAD and TAIL tools: head -n 10000 plot.list > small.list

- supported document types and their syntax in the corpus:
  - movie: MV: <title> (<year>)
  - series: MV: "<title>" (<year>)
  - episode: MV: "<title>" (<year>) {<episodetitle>}
  - television: MV: <title> (<year>) (TV)
  - video: MV: <title> (<year>) (V)
  - videogame: MV: <title> (<year>) (VG)

[1] http://www.imdb.com/interfaces

Preprocessing

- Corpus: the corpus text has to tokenized.
- Phrase search: the query has to be tokenized, too.

- Convert all words to lower case (case-insensitive search and indexing) and remove English stop words.

- There are built-in “Analyzers” for this in Lucene.

Getting started

- Download Apache Lucene v6.3.

- Extract the zip file, copy the two jars to your project and add them to the build-path:
  - lucene-core-6.3.0.jar
  - lucene-queryparser-6.3.0.jar

- in BooleanSeachLucene.java, implement the functions:
  - public void buildIndices(String plotFile)
    (used to parse the file and build the lucene index)
  - public Set<String> booleanQuery(String queryString)
    (parses the query string and returns the title lines of any entries in the plotFile matching the query)
  - public void close()
    (can be used to close Lucene index, Threadpool, etc.)

Test your Program

- we provide you with a modified:
  - a queries_lucene.txt file containing exemplary queries
  - a results_lucene.txt file containing the expected results of running these queries
  - a main method for testing your code (which expects as parameters the corpus file, the queries file and the results file)

- additionally, you can write your own test queries:
  - check the plausibility of your results using GREP:
    grep " <search-token> " <corpus-file>
  - use -G or -P parameter for regular expressions

Deliverables

- by Thursday, 05.01.17, 23:59 (midnight)
- submission: archive (zip, tar.gz)
  - contains Java source files, any used libraries, and your compiled jar named BooleanQueryLucene.jar
  - file name (of submitted archive): your group name
- upload to https://box.hu-berlin.de/u/d/6421657d18/
  - if this doesn't work, send via mail to buxmarcn@informatik.hu-berlin.de
- test your jar before submitting by running our queries on gruenau2
  - java -jar BooleanQueryLucene.jar <plot list file> <queries file> <results file>
  - you might have to increase the JVM's heap size (e.g., -Xmx8g)
  - your jar must run and answer all test queries in 'queries.txt' correctly

Presentation of Solutions

- you are be able to pick when and what you‘d like to present (first-come-first-served):
  - monday: https://dudle.inf.tu-dresden.de/inforet_ue3_mo/
  - tuesday: https://dudle.inf.tu-dresden.de/inforet_ue3_tu/

- presentation will be given on 9./10.01.17

- One team can present their Lucene query parser.
- One team can present their Lucene indexer (+custom analyzer).

Competition

- Index as fast as possible.
- See http://www.lucenetutorial.com/lucene-nrt-hello-world.html for possible optimizations…
- Note that everybody uses the same indexer (Lucene).
- stay under 50 GB memory usage.
- we will call the program using our eval tool:
  - we will use different queries and -Xmx50g parameter
- We will evaluate twofold:
  a) The total query time.
  b) The total time for building the index.

Checklist

again, before submitting your results, make sure that you

1. did not change or remove any code from BooleanQueryLucene.java
2. did not alter the functions‘ signatures (types of params, return values)
3. only use the default constructor and don‘t change its parameters
4. did not change the class or package name
5. named your jar BooleanQueryLucene.jar
6. tested your jar on gruenau2 by running

   ```
   java -jar BooleanQueryLucene.jar plot.list queries.txt results.txt
   ```

   (you might have to increase Java heap space, e.g. -Xmx6g)
7. ascertained that the queries in queries.txt were answered correctly
8. Make sure to upload a zip file named by your group name.

Next Steps

- this week: evaluation of assignment 2

- next weeks: academic holidays starting from Dec 19th to Jan 2nd. There is just one Q/A session on Jan 3rd for assignment 3.

- Upload your solution by Thursday, 05.01.17, 23:59 (midnight)