MAC - Labo 1

Guillaume Hochet et Vincent Guidoux

MAC - Labo 1
- D - Understanding the Lucene API
  - 1 - Does the command line demo use stopword removal ? Explain how you find out the answer.
  - 2 - Does the command line demo use stemming ? Explain how you find out the answer.
  - 3 - Is the search of the command line demo case insensitive? How did you find out the answer?
  - 4 - Does it matter whether stemming occurs before or after stopword removal? Consider this as a general question.
- F - Indexing and Searching the CACM collection
  - Indexing
    - 4 - Keep the publication id in the index and show it in the results set of your queries.
    - 5 - Which field type can be used for id, title, summary and author ?
    - 6 - What should be added to the code to have access to the term vector in the index?
    - Code of indexing
  - Using different Analyzers
  - Reading Index
    - 1- Author with maximum of publications
    - 2 - Top 10 terms in the title field
    - Code
  - Searching
    - compiler program
    - Information Retrieval
    - Information AND Retrieval
    - Retrieval AND Information~ -Database
    - Info*
    - Information Retrieval~5
    - Code
  - Tuning the Lucene Score
    - ClassicSimilarity()
    - MySimilarity()
    - Code

D - Understanding the Lucene API

1 - Does the command line demo use stopword removal ? Explain how you find out the answer.

Yes the command line demo uses stopword removal. We tested it simply by making the following requests:

- `the` : no results, mean empty request performed
- `kill` : 1 result
- `kill the` : 1 result (same)
- `the kill` : 1 result (same)

2 - Does the command line demo use stemming ? Explain how you find out the answer.

No, it doesn't use stemming, we checked it easily by performing the following two queries:

- `kill` : 1 result
- `killed` : 0 results

3 - Is the search of the command line demo case insensitive? How did you find out the answer?

Yes it is case insensitive, we checked it by performing the following queries:

- `kill`
- `KILL`
- `KiLl`

which all returned the same single result

4 - Does it matter whether stemming occurs before or after stopword removal? Consider this as a general question.

Well it depends if our list of stopwords includes stems or not. If so we should first apply stemming and then stop words removal (to stem those stop words that might not be removed).

Otherwise we should perform stop words removal to gain performance (avoid stemming stop words that are eventually removed)What should be added to the code to have access to the term vector in the index ?

F - Indexing and Searching the CACM collection

Indexing

4 - Keep the publication id in the index and show it in the results set of your queries.

```
TODO
```

5 - Which field type can be used for id, title, summary and author ?

- `id` : `LongPoint` and `StoredField`
- `title` : `Field` with `TextField.TYPE_STORED`
- `summary` :

```
FieldType sumType = new FieldType(TextField.TYPE_NOT_STORED);

//to have access to the term vector in the index

sumType.setIndexOptions(IndexOptions.DOCS_AND_FREQS_AND_POSITIONS_AND_OFFSETS);
sumType.setStoreTermVectors(true);
sumType.setStoreTermVectorPositions(true);
sumType.setStoreTermVectorOffsets(true);

sumType.setTokenized(true);
```

- `author` : `StringField` with `TextField.TYPE_STORED`

6 - What should be added to the code to have access to the term vector in the index?

https://lucene.apache.org/core/5_2_1/core/org/apache/lucene/index/IndexableFieldType.html#storeTermVectors

```
sumType.setIndexOptions(IndexOptions.DOCS_AND_FREQS_AND_POSITIONS_AND_OFFSETS);
sumType.setStoreTermVectors(true);
sumType.setStoreTermVectorPositions(true);
sumType.setStoreTermVectorOffsets(true);storeTermVectors();
```

Code of indexing

```
@Override
    public void onNewDocument(Long id, String authors, String title, String summary) {

        Document doc = new Document();

        //Keep the publication id in the index and show it in the results set of your
queries.
        doc.add(new LongPoint("id", id));
        doc.add(new StoredField("id", id));

        if (authors != null && authors.length() > 0)
            for (String author : authors.split(";"))
                doc.add(new StringField("authors", author, Field.Store.YES));

        doc.add(new Field("title", title, TextField.TYPE_STORED));

        if (summary != null && summary.length() > 0) {

            FieldType sumType = new FieldType(TextField.TYPE_NOT_STORED);

            //to have access to the term vector in the index

 sumType.setIndexOptions(IndexOptions.DOCS_AND_FREQS_AND_POSITIONS_AND_OFFSETS);
            sumType.setStoreTermVectors(true);
            sumType.setStoreTermVectorPositions(true);
```

```
            sumType.setStoreTermVectorOffsets(true);

            sumType.setTokenized(true);
            doc.add(new Field("summary", summary, sumType));
        }
        try {
            this.indexWriter.addDocument(doc);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Using different Analyzers

See table on the last page

Reading Index

1- Author with maximum of publications

Thacher Jr., H. C. with 38 publications

2 - Top 10 terms in the title field

Top ranking terms	Frequency in the title field
algorithm	961
computer	260
system	172
programming	154
method	124
data	110
systems	108
language	99
program	93
matrix	82

Code

```
public void printTopRankingTerms(String field, int numTerms) {

    Comparator comparator = new HighFreqTerms.DocFreqComparator();

    try {
        TermStats[] stats = HighFreqTerms.getHighFreqTerms(this.indexReader, numTerms,
field, comparator);
        System.out.println("Top ranking terms for field [" + field + "] are: ");
        for(TermStats stat : stats)
            System.out.println(stat.termtext.utf8ToString() + " with " + stat.docFreq +
" frequencies");
    } catch (Exception e) {
        System.out.println("Failed querying");
    }
}
```

Searching

compiler program

```
Searching for [compiler program]
Results found: 578
3189:An Algebraic Compiler for the FORTRAN Assembly Program(1.0367663)
1459:Requirements for Real-Time Languages(0.9427518)
2652:Reduction of Compilation Costs Through Language Contraction(0.93779767)
1183:A Note on the Use of a Digital Computerfor Doing Tedious Algebra and
Programming(0.8799802)
1465:Program Translation Viewed as a General Data Processing Problem(0.82941306)
1988:A Formalism for Translator Interactions(0.82941306)
1647:WATFOR-The University of Waterloo FORTRAN IV Compiler(0.8082413)
1237:Conversion of Decision Tables To Computer Programs(0.7542014)
2944:Shifting Garbage Collection Overhead to Compile Time(0.7542014)
637:A NELIAC-Generated 7090-1401 Compiler(0.74336267)
```

Information Retrieval

```
Searching for [Information Retrieval]
Results found: 188
1457:Data Manipulation and Programming Problemsin Automatic Information
Retrieval(1.180699)
891:Everyman's Information Retrieval System(1.1004101)
1935:Randomized Binary Search Technique(1.0225154)
1699:Experimental Evaluation of InformationRetrieval Through a
Teletypewriter(0.9628588)
3134:The Use of Normal Multiplication Tablesfor Information Storage and
Retrieval(0.9191686)
2307:Dynamic Document Processing(0.913785)
1032:Theoretical Considerations in Information Retrieval Systems(0.8855243)
1681:Easy English,a Language for InformationRetrieval Through a Remote Typewriter
Console(0.87130356)
2990:Effective Information Retrieval Using Term Accuracy(0.87130356)
2519:On the Problem of Communicating Complex Information(0.7863115)
```

Information AND Retrieval

```
Searching for [Information AND Retrieval]
Results found: 23
1457:Data Manipulation and Programming Problemsin Automatic Information
Retrieval(1.180699)
891:Everyman's Information Retrieval System(1.1004101)
1935:Randomized Binary Search Technique(1.0225154)
1699:Experimental Evaluation of InformationRetrieval Through a
Teletypewriter(0.9628588)
3134:The Use of Normal Multiplication Tablesfor Information Storage and
Retrieval(0.9191686)
2307:Dynamic Document Processing(0.913785)
1032:Theoretical Considerations in Information Retrieval Systems(0.8855243)
1681:Easy English,a Language for InformationRetrieval Through a Remote Typewriter
Console(0.87130356)
2990:Effective Information Retrieval Using Term Accuracy(0.87130356)
2519:On the Problem of Communicating Complex Information(0.7863115)
```

Retrieval AND Information~ -Database

```
Searching for [Retrieval AND Information~ -Database]
Results found: 1
2543:Reducing the Retrieval Time of Scatter Storage Techniques(0.5658677)
```

Info*

```
Searching for [Info*]
Results found: 193
222:Coding Isomorphisms(1.0)
272:A Storage Allocation Scheme for ALGOL 60(1.0)
396:Automation of Program  Debugging(1.0)
397:A Card Format for Reference Files in Information Processing(1.0)
409:CL-1, An Environment for a Compiler(1.0)
440:Record Linkage(1.0)
483:On the Nonexistence of a Phrase Structure Grammar for ALGOL 60(1.0)
616:An Information Algebra - Phase I Report-LanguageStructure Group of the CODASYL
Development Committee(1.0)
644:A String Language for Symbol Manipulation Based on ALGOL 60(1.0)
655:COMIT as an IR Language(1.0)
```

Information Retrieval~5

```
Searching for [Information Retrieval~5]
Results found: 191
1457:Data Manipulation and Programming Problemsin Automatic Information
Retrieval(0.6189328)
891:Everyman's Information Retrieval System(0.567616)
1935:Randomized Binary Search Technique(0.5360115)
3134:The Use of Normal Multiplication Tablesfor Information Storage and
Retrieval(0.50438124)
1699:Experimental Evaluation of InformationRetrieval Through a
Teletypewriter(0.49666396)
2307:Dynamic Document Processing(0.48708913)
1032:Theoretical Considerations in Information Retrieval Systems(0.4641996)
1681:Easy English,a Language for InformationRetrieval Through a Remote Typewriter
Console(0.4381463)
2990:Effective Information Retrieval Using Term Accuracy(0.4381463)
2519:On the Problem of Communicating Complex Information(0.42421836)
```

Code

```
public void query(String q) {

    try {
        QueryParser parser = new QueryParser("summary", this.analyzer);
        Query query = parser.par    ScoreDoc[] results =
 indexSearcher.search(query, 5000).scor    System.out.println("Searching for [" + q +
"]");
        System.out.println("Results found: " + results.length);

        int limit = results.length > 10 ? 10 : results.length;

        for (int i = 0; i < limit; i++) {
            ScoreDoc result = results[i];
            Document doc = indexSearcher.doc(result.doc);
            System.out.println(doc.get("id") + ":" + doc.get("title") + "(" +
result.score + ")");
```

```
        }
    } catch (Exception e) {
        System.out.println("Query couldnt be performed, error: " +
e.getMessage());
    }
}
```

Tuning the Lucene Score

ClassicSimilarity()

```
Searching for [compiler program]
Results found: 578
3189:An Algebraic Compiler for the FORTRAN Assembly Program(1.0367663)
1459:Requirements for Real-Time Languages(0.9427518)
2652:Reduction of Compilation Costs Through Language Contraction(0.93779767)
1183:A Note on the Use of a Digital Computerfor Doing Tedious Algebra and
Programming(0.8799802)
1465:Program Translation Viewed as a General Data Processing Problem(0.82941306)
1988:A Formalism for Translator Interactions(0.82941306)
1647:WATFOR-The University of Waterloo FORTRAN IV Compiler(0.8082413)
1237:Conversion of Decision Tables To Computer Programs(0.7542014)
2944:Shifting Garbage Collection Overhead to Compile Time(0.7542014)
637:A NELIAC-Generated 7090-1401 Compiler(0.74336267)
```

MySimilarity()

```
Searching for [compiler program]
Results found: 578
2534:Design and Implementation of a Diagnostic Compiler for PL/I(7.411339)
637:A NELIAC-Generated 7090-1401 Compiler(6.836161)
2652:Reduction of Compilation Costs Through Language Contraction(6.555643)
2923:High-Level Data Flow Analysis(6.465453)
1647:WATFOR-The University of Waterloo FORTRAN IV Compiler(6.258721)
1465:Program Translation Viewed as a General Data Processing Problem(5.8141103)
1988:A Formalism for Translator Interactions(5.8141103)
3189:An Algebraic Compiler for the FORTRAN Assembly Program(5.8141103)
1135:A General Business-Oriented Language Based on Decision Expressions*(5.0494967)
1237:Conversion of Decision Tables To Computer Programs(5.0494967)
```

Code

```
public class MySimilarity extends ClassicSimilarity {

    @Override
    public float lengthNorm(FieldInvertState state) {
        return 1;
    }

    @Override
    public float tf(float freq) {
```

```
        return (float)Math.log(freq) + 1;
    }

    @Override
    public float idf(long docFreq, long docCount) {

        return (float)(Math.log((docCount / (docFreq + 1)) + 1));
    }

    @Override
    public float coord(int overlap, int maxOverlap) {

        return (float)Math.sqrt(overlap / maxOverlap);
    }
}
```