Introduction to Lucene

Gary Marigliano

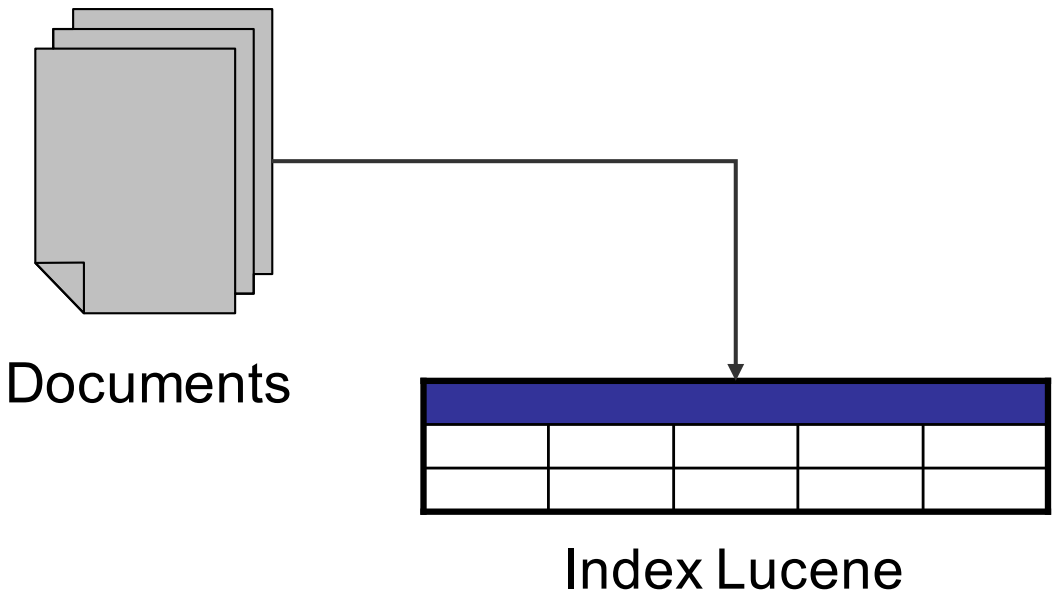
heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

What is Lucene?

- Powerful, high-performance, scalable full text search engine library
- Open source under Apache Software License
- Originally written in Java by Doug Cutting
- Ported to C#, C++, Delphi, Perl, Python, PHP, Ruby
- Initial release in 2000 (current version 7.0.0)
- We use Lucene version 6.6.1 in this lab

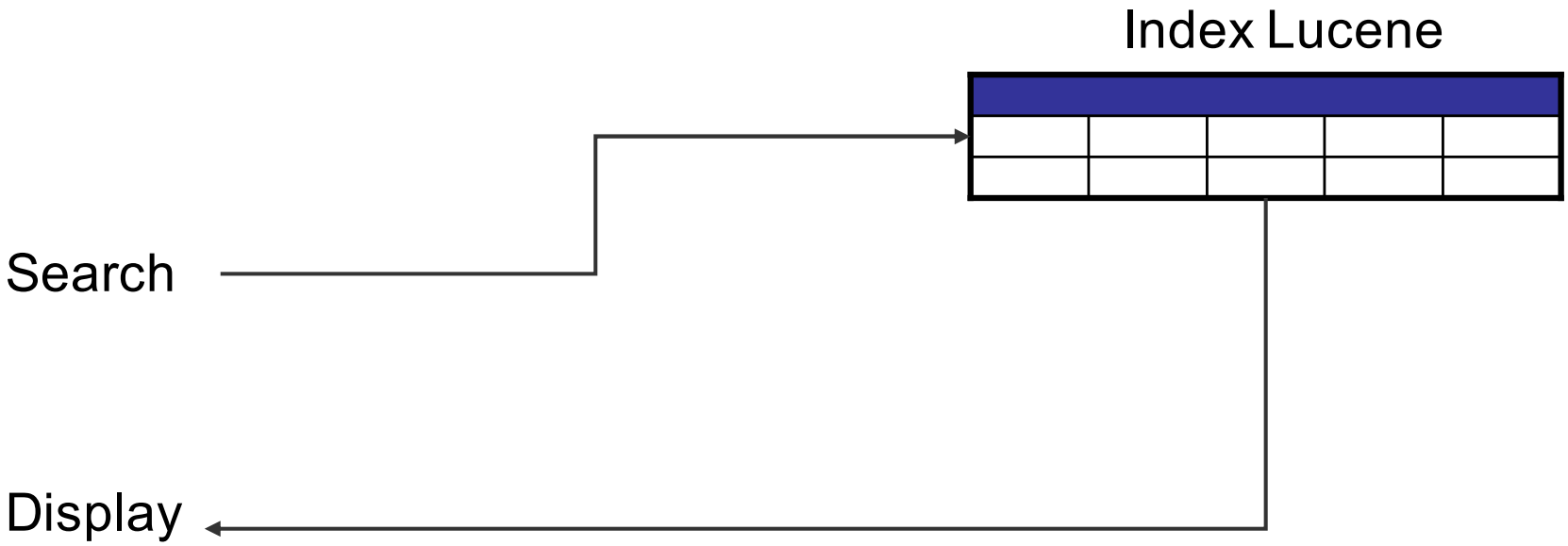
Building Applications using Lucene (1)

- Step 1: Index Data
  - Convert files to a format for quick look-up
  - Data structure that allows fast random access to words stored inside



Building Applications using Lucene (2)

- Step 2: Search
  - Lookup words to find the documents that are relevant for the search
  - Support for different type of queries
  - Display results: speed, ranking



Lucene Definitions

Fundamental concepts in Lucene:

- **Index:** contains a sequence of documents
- **Document:** is a sequence of fields
- **Field:** is a named sequence of terms
- **Term:** is a sequence of bytes

The terms are represented as a pair: the string naming the field, and the bytes within the field.

Lucene Classes (1)

- **Document:** `org.apache.lucene.document.Document`
  - Indexed data is organized into documents
- **IndexWriter:** `org.apache.lucene.index.IndexWriter`
  - Writes data / documents into index
- **IndexReader:** `org.apache.lucene.index.IndexReader`
  - Reads the index (abstract class)
- **DirectoryReader:** `org.apache.lucene.index.DirectoryReader`
  - Reads indexes in a directory
- **IndexSearcher:** `org.apache.lucene.search.IndexSearcher`
  - Searches the index (using the IndexReader)

Lucene Classes (2)

- **Field:** `org.apache.lucene.document.Field`
  - A field is a section of a Document. Each document can contain different named fields.
    - **IntPoint:** A field that indexes int values for efficient range filtering and sorting. If you also need to store the value, you should add a separate **StoredField** instance
    - **StringField:** A field that is indexed but not tokenized (the entire String value is indexed as a single token).
    - **TextField:** A field that is indexed and tokenized, without term vectors.
    - **Field:** A general purpose field that allows specifying each part of a field (name, value and type). Use this instead of TextField to be able to access the Term Vector of the field.

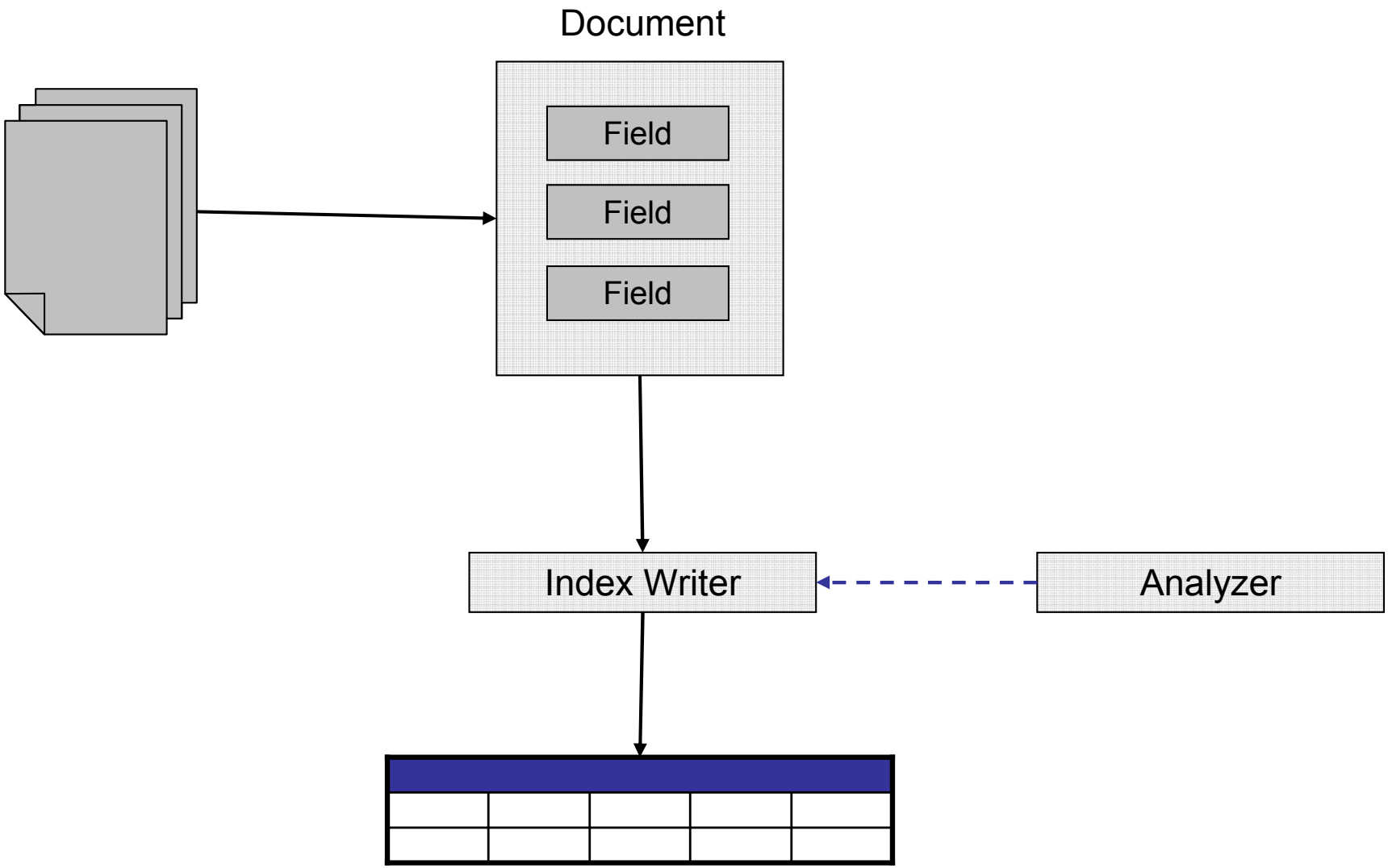
Lucene Analyzer (1)

- **Analyzer:** `org.apache.lucene.analysis.Analyzer`
  - Converts text into tokens for indexing / searching
  - Use the *same analyzer* for indexing and searching
  - Abstract class
- **WhitespaceAnalyzer:** `org.apache.lucene.analysis.core.WhitespaceAnalyzer`
  - Uses a whitespace tokenizer
- **StopAnalyzer:** `org.apache.lucene.analysis.core.StopAnalyzer`
  - LetterTokenizer: divides text at non-letters
  - Lowercase
  - Removes stopwords (predefined English stopwords)

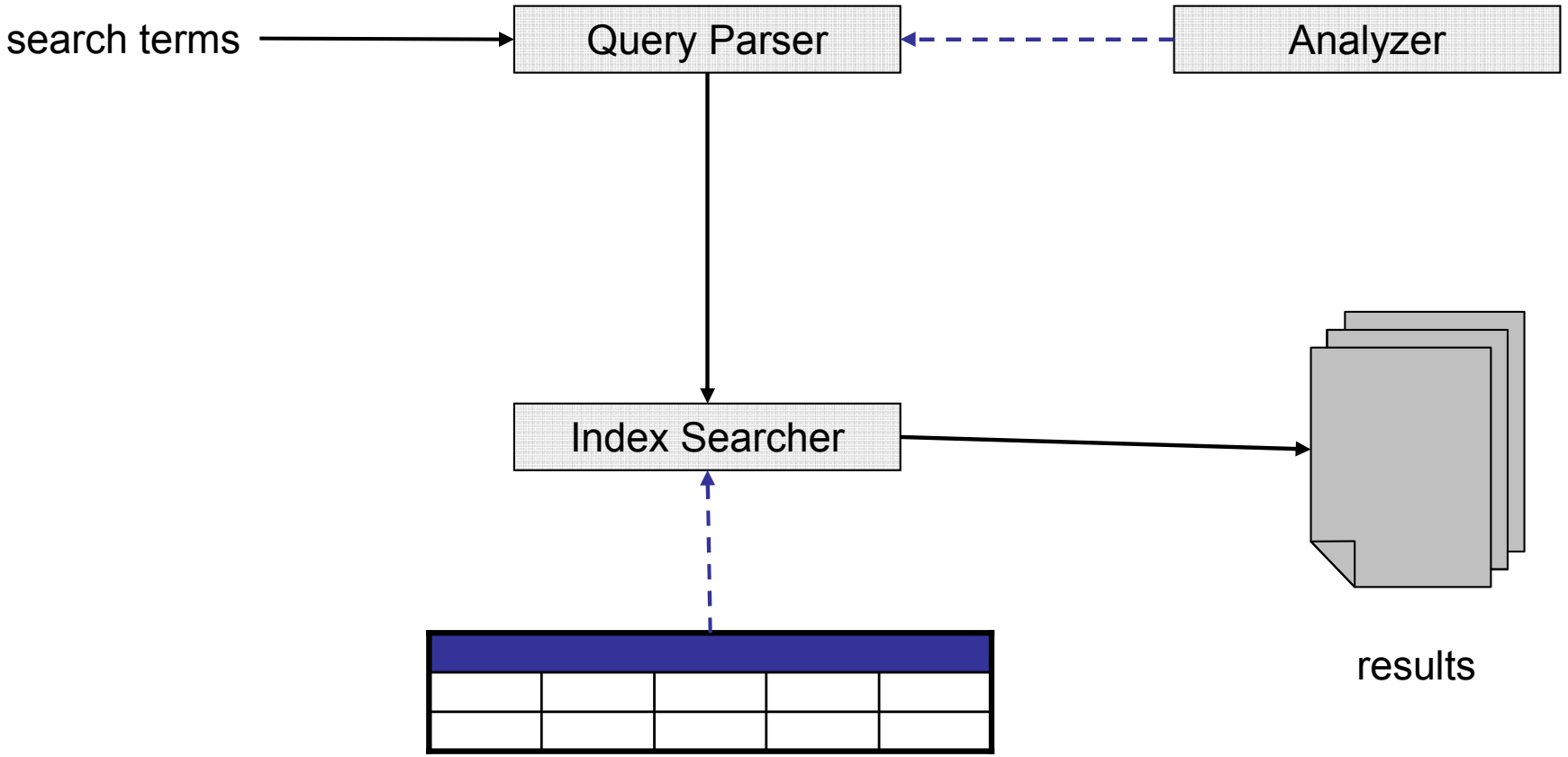
Lucene Analyzer (2)

- **StandardAnalyzer:**
  `org.apache.lucene.analysis.standard.StandardAnalyzer`
  - StandardTokenizer: grammar-based tokenizer
- **EnglishAnalyzer:**
  `org.apache.lucene.analysis.en.EnglishAnalyzer`
  - Stemming (e.g. studying -> study, administration -> administr)
  - Support for different languages: English, French, German, etc.
- **Shingling:**
  `org.apache.lucene.analysis.shingle.ShingleAnalyzerWrapper`
  - Standard analyzer + Shingling (e.g. "information retrieval")
  - Size of shingles (min and max size)

Lucene Indexing Flow



Lucene Searching Flow



Lucene Queries

- **TermQuery:** matches all the documents that contain the specified Term (which is a word that occurs in a certain field)
- **BooleanQuery:** contains multiple queries with an operator
  - SHOULD
  - MUST
  - MUST NOT
- **PhraseQuery:** finds documents containing certain phrases
- **Numeric Queries:** matches all documents that occur in a numeric range for example **IntPoint.newRangeQuery()**
- **PrefixQuery:** identifies all documents with terms that begin with a certain string
- **QueryParser:** converts the query into an index searchable form

Lucene Application

- Application: analysis of CACM publication list
  - {id \t {author";"} \t title \t [summary] \n}
- Index publication list using StandardAnalyzer
- Search on Field **summary** using QueryParser
  - User query: "compiler program"
  - Result format: {id: title "("lucene score")" \n}

```
3189: An Algebraic Compiler for the FORTRAN Assembly Program (1.2154248)
1465: Program Translation Viewed as a General Data Processing Problem (1.1353518)
123: Compilation for Two Computers with NELIAC (0.97233987)
1460: Evolution of the Meta-Assembly Program (0.94987315)
2534: Design and Implementation of a Diagnostic Compiler for PL/I (0.8594351)
1647: WATFOR-The University of Waterloo FORTRAN IV Compiler (0.8507974)
1788: Toward a General Processor for Programming Languages (0.8507974)
1215: Some Techniques Used in the ALCOR ILLINOIS 7090 (0.7693181)
730: MIRFAG: A Compiler Based on StandardMathematical Notation And Plain English (0.75989854)
1646: DITRAN-A Compiler Emphasizing Diagnostics (0.75989854)
```

Lucene Demo: Indexing

```
// 1.1. create an analyzer
Analyzer analyzer = new StandardAnalyzer();
// 1.2. create an index writer config
IndexWriterConfig iwc = new IndexWriterConfig(analyzer);
iwc.setOpenMode(OpenMode.CREATE); // create and replace existing index
iwc.setUseCompoundFile(false); // not pack newly written segments in a compound file:
//keep all segments of index separately on disk
// 1.3. create index writer
Path path = FileSystems.getDefault().getPath("index");
Directory dir = FSDirectory.open(path);
IndexWriter indexWriter = new IndexWriter(dir, iwc);
// 1.4. create document
Document doc = new Document();
// 1.5. create fields
FieldType fieldType = new FieldType(); // describes properties of a field
fieldType.setIndexOptions(IndexOptions.DOCS); // controls how much information
//is stored in the postings lists.
fieldType.setTokenized(true); // tokenize the field's contents using configured analyzer
fieldType.freeze(); // prevents future changes
Field field = new Field("summary", cacm.getSummary(), fieldType);
...
// 1.6. add fields to document
doc.add(field);
...
// 1.7. add document to index
indexWriter.addDocument(doc);
// 1.8 close index writer
indexWriter.close();
dir.close();
```

for each document

for each field

Lucene Demo: Searching

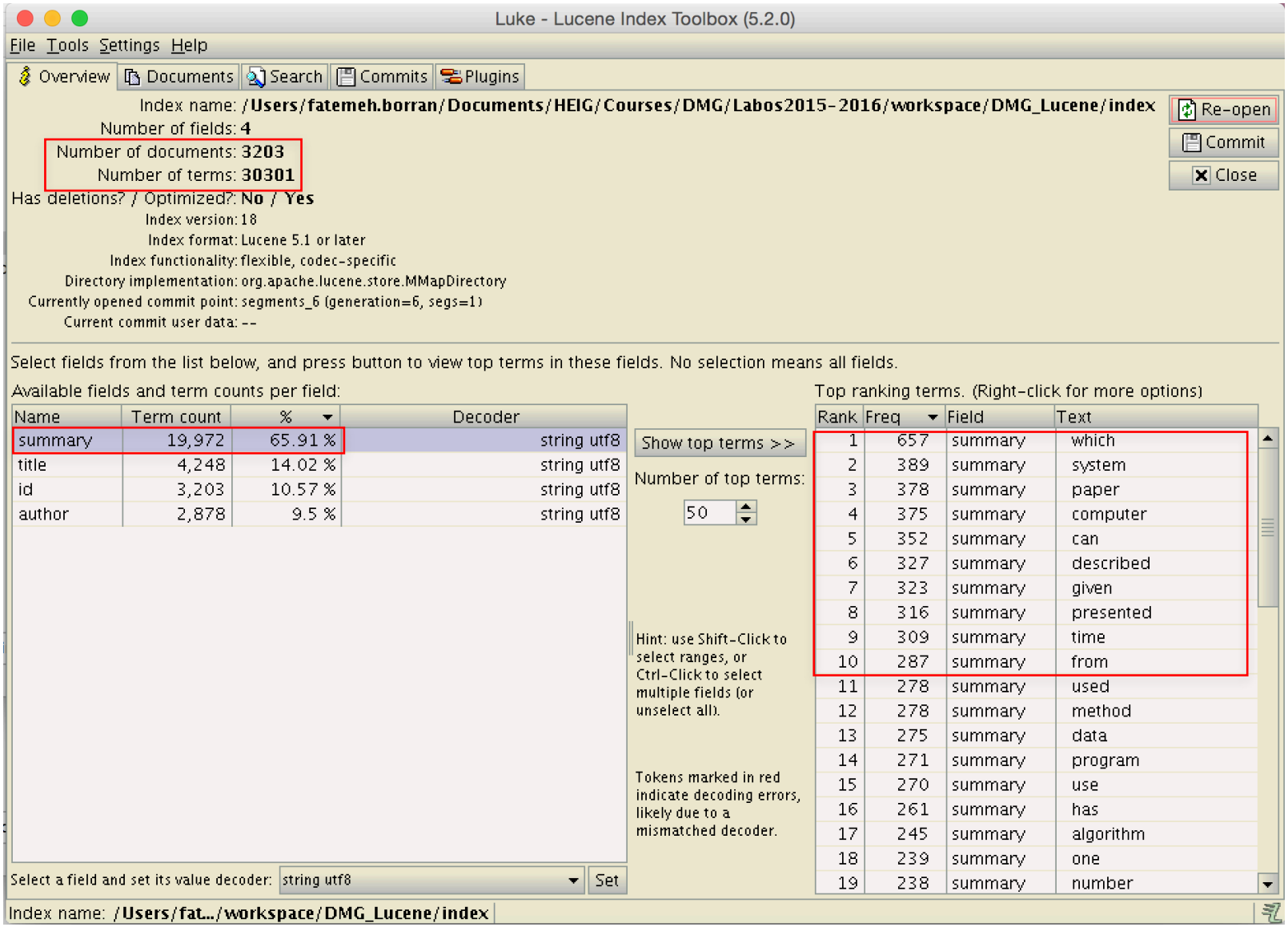
```
// 2.1. create query parser
QueryParser parser = new QueryParser("summary", analyzer);
// 2.2. parse query
Query query = parser.parse("compiler program");

// 3.1. create index reader
Path path = FileSystems.getDefault().getPath("index");
Directory dir = FSDirectory.open(path);
IndexReader indexReader = DirectoryReader.open(dir);
// 3.2. create index searcher
IndexSearcher indexSearcher = new IndexSearcher(indexReader);
// 3.3. search query
ScoreDoc[] hits = indexSearcher.search(query, 1000).scoreDocs;
// 3.4. retrieve results
System.out.println("Results found: " + hits.length);
for (ScoreDoc hit: hits) {
    Document doc = indexSearcher.doc(hit.doc)
    System.out.println(doc.get("id") + ": " + doc.get("title") + " (" + hit.score + ")");
}
// 3.5. close index reader
indexReader.close();
dir.close();
```

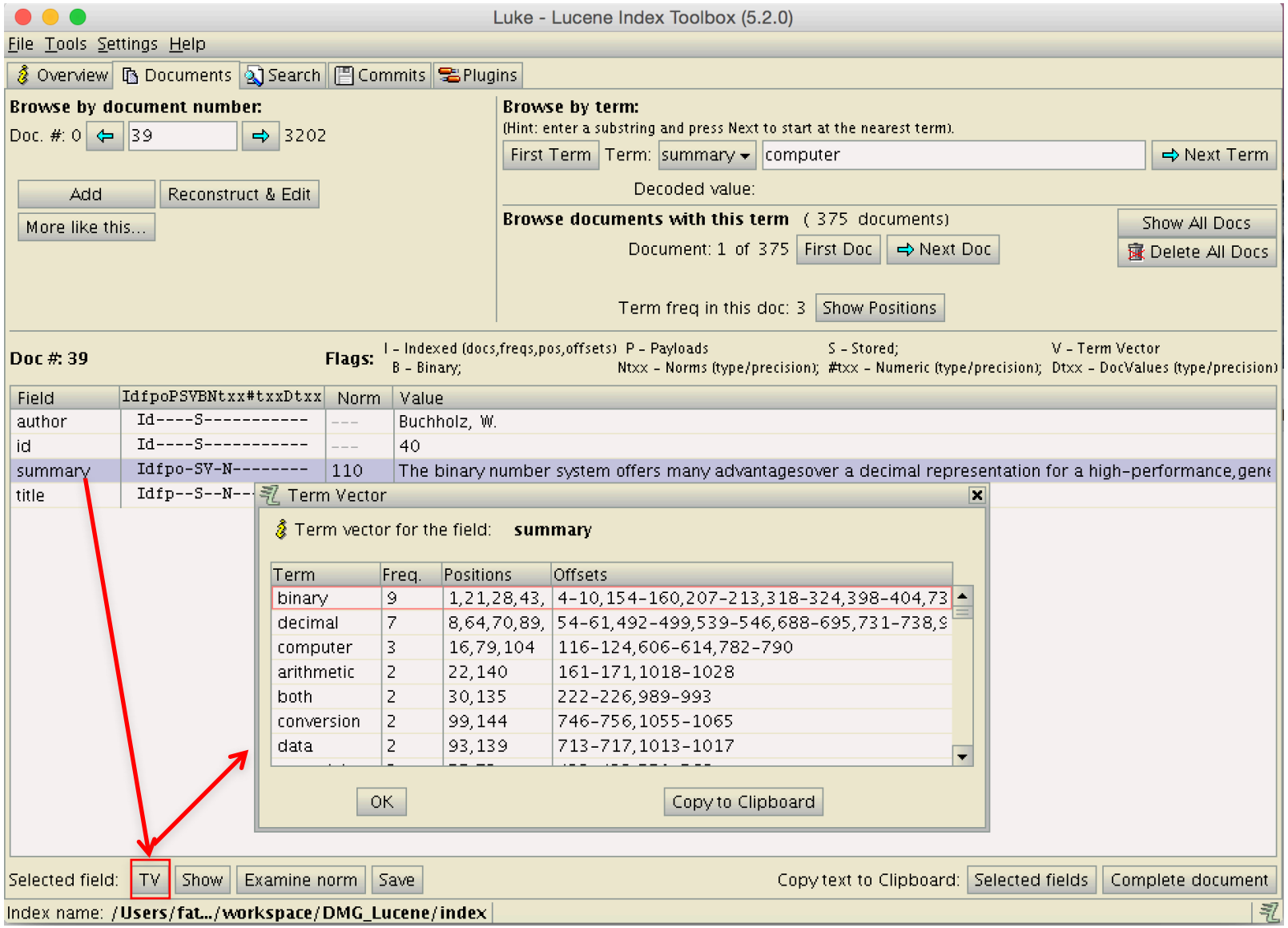
Luke

- A GUI tool written in Java
- Browse the contents of a Lucene index
- Examine individual documents
- Run queries over the index

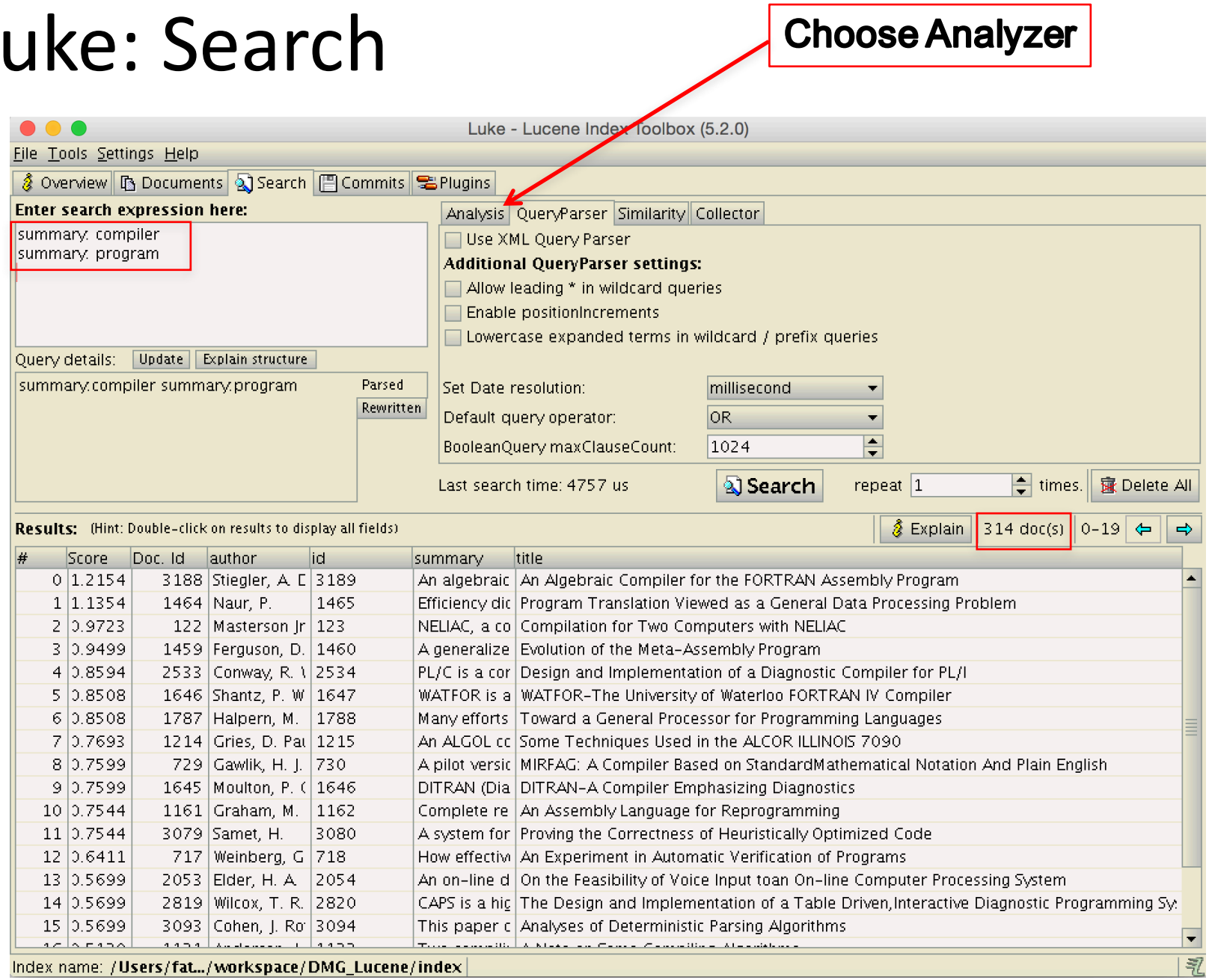
Luke: Index


Luke - Lucene Index Toolbox (5.2.0)
File Tools Settings Help
Overview Documents Search Commits Plugins
Index name: /Users/fatemeh.borran/Documents/HEIG/Courses/DMG/Labos2015-2016/workspace/DMG_Lucene/index
Number of fields: 4
Number of documents: 3203
Number of terms: 30301
Has deletions? / Optimized?: No / Yes
Index version: 18
Index format: Lucene 5.1 or later
Index functionality: flexible, codec-specific
Directory implementation: org.apache.lucene.store.MMapDirectory
Currently opened commit point: segments_6 (generation=6, segs=1)
Current commit user data: --
Re-open
Commit
Close
Select fields from the list below, and press button to view top terms in these fields. No selection means all fields.
Available fields and term counts per field:
Name Term count % Decoder
summary 19,972 65.91 % string utf8
title 4,248 14.02 % string utf8
id 3,203 10.57 % string utf8
author 2,878 9.5 % string utf8
Show top terms >>
Number of top terms:
50
Hint: use Shift-Click to select ranges, or Ctrl-Click to select multiple fields (or unselect all).
Tokens marked in red indicate decoding errors, likely due to a mismatched decoder.
Top ranking terms. (Right-click for more options)
Rank Freq Field Text
1 657 summary which
2 389 summary system
3 378 summary paper
4 375 summary computer
5 352 summary can
6 327 summary described
7 323 summary given
8 316 summary presented
9 309 summary time
10 287 summary from
11 278 summary used
12 278 summary method
13 275 summary data
14 271 summary program
15 270 summary use
16 261 summary has
17 245 summary algorithm
18 239 summary one
19 238 summary number
Select a field and set its value decoder: string utf8 Set
Index name: /Users/fat.../workspace/DMG_Lucene/index

Luke: TermVector



Luke: Search


Choose Analyzer
Luke - Lucene Index Toolbox (5.2.0)
File Tools Settings Help
Overview Documents Search Commits Plugins
Enter search expression here:
summary: compiler
summary: program
Query details: Update Explain structure
summary:compiler summary:program
Parsed
Rewritten
Analysis QueryParser Similarity Collector
Use XML Query Parser
Additional QueryParser settings:
Allow leading * in wildcard queries
Enable positionIncrements
Lowercase expanded terms in wildcard / prefix queries
Set Date resolution: millisecond
Default query operator: OR
BooleanQuery maxClauseCount: 1024
Last search time: 4757 us
Search
repeat 1 times.
Delete All
Results: (Hint: Double-click on results to display all fields)
Explain
314 doc(s)
0-19
Score Doc. Id author id summary title
0 1.2154 3188 Stiegler, A. E 3189 An algebraic An Algebraic Compiler for the FORTRAN Assembly Program
1 1.1354 1464 Naur, P. 1465 Efficiency dic Program Translation Viewed as a General Data Processing Problem
2 0.9723 122 Masterson Jr 123 NELIAC, a co Compilation for Two Computers with NELIAC
3 0.9499 1459 Ferguson, D. 1460 A generalize Evolution of the Meta-Assembly Program
4 0.8594 2533 Conway, R. \ 2534 PL/C is a cor Design and Implementation of a Diagnostic Compiler for PL/I
5 0.8508 1646 Shantz, P. W 1647 WATFOR is a WATFOR-The University of Waterloo FORTRAN IV Compiler
6 0.8508 1787 Halpern, M. 1788 Many efforts Toward a General Processor for Programming Languages
7 0.7693 1214 Gries, D. Pat 1215 An ALGOL cc Some Techniques Used in the ALCOR ILLINOIS 7090
8 0.7599 729 Gawlik, H. J. 730 A pilot versic MIRFAG: A Compiler Based on StandardMathematical Notation And Plain English
9 0.7599 1645 Moulton, P. (1646 DITRAN (Dia DITRAN-A Compiler Emphasizing Diagnostics
10 0.7544 1161 Graham, M. 1162 Complete re An Assembly Language for Reprogramming
11 0.7544 3079 Samet, H. 3080 A system for Proving the Correctness of Heuristically Optimized Code
12 0.6411 717 Weinberg, G 718 How effectiv An Experiment in Automatic Verification of Programs
13 0.5699 2053 Elder, H. A. 2054 An on-line d On the Feasibility of Voice Input toan On-line Computer Processing System
14 0.5699 2819 Wilcox, T. R. 2820 CAPS is a hig The Design and Implementation of a Table Driven,Interactive Diagnostic Programming Sy
15 0.5699 3093 Cohen, J. Ro 3094 This paper c Analyses of Deterministic Parsing Algorithms
Index name: /Users/fat.../workspace/DMG_Lucene/index

References

- Apache Lucene: http://lucene.apache.org/core/6_6_1/index.html
- Lucene Tutorials:
  - https://www.tutorialspoint.com/lucene/lucene_search_operation.htm
  - https://howtodoinjava.com/lucene/lucene-index-search-examples/
- Luke: https://github.com/DmitryKey/luke