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

T-InfData

Introduction to Lucene

Autumn 2020

Prof. Nastaran Fatemi
Christopher Meier

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

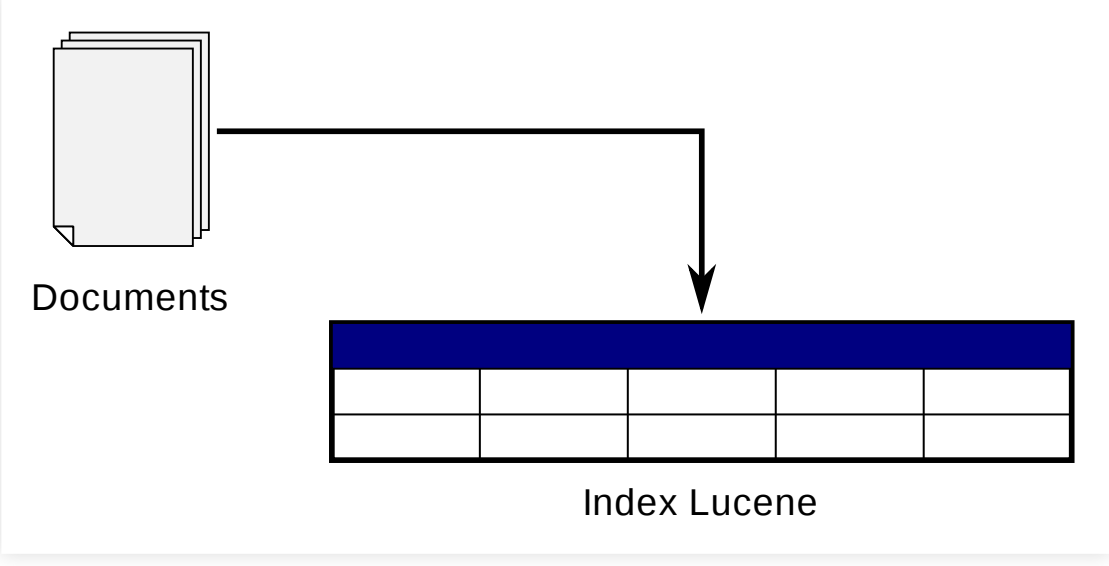
Initial release in 2000 (current version 8.6.2)

We use Lucene version 8.6.2 in this lab

Building Applications using Lucene (1)

Step 1: Index Data

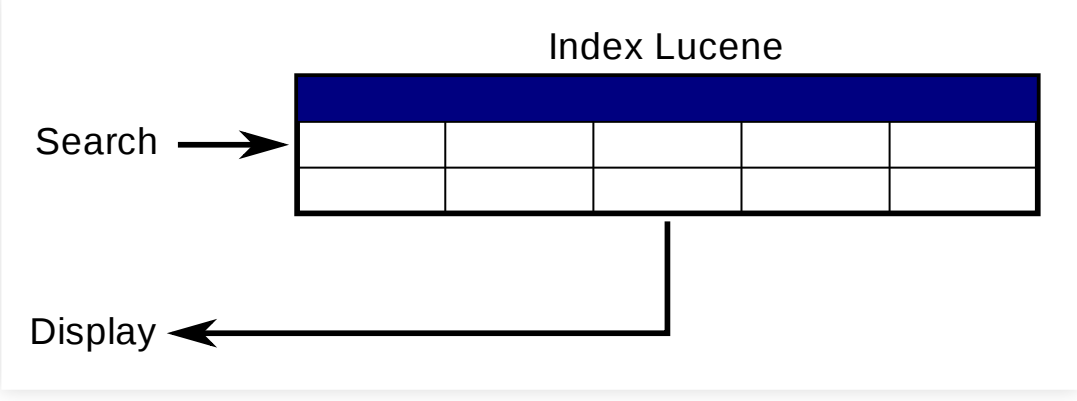
- Convert files to a format for quick look-up
- Data structure that allows fast random access to words stored inside



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)

A field (`org.apache.lucene.document.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

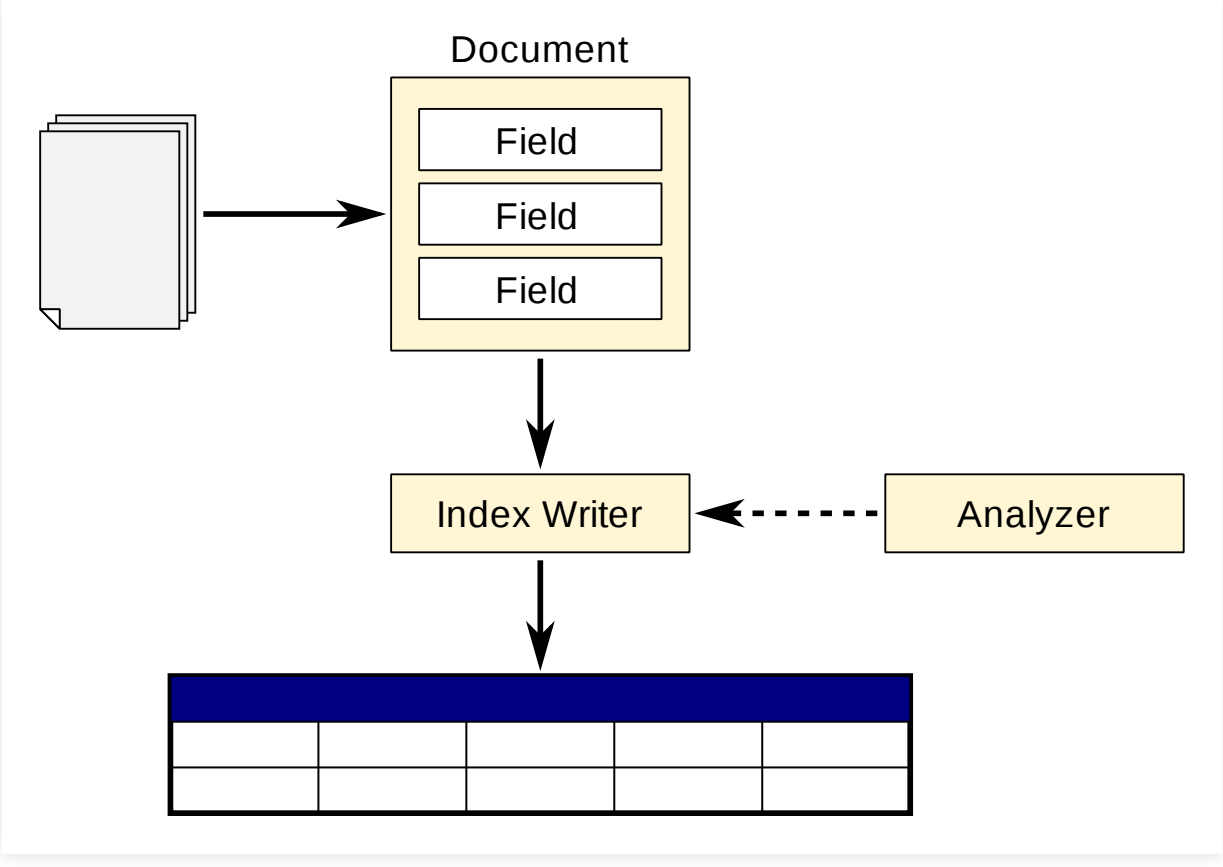
EnglishAnalyzer `org.apache.lucene.analysis.en.EnglishAnalyzer`

- Stemming (e.g. studying → study, administration → administr)
- Support for different languages: English, French, German, etc.

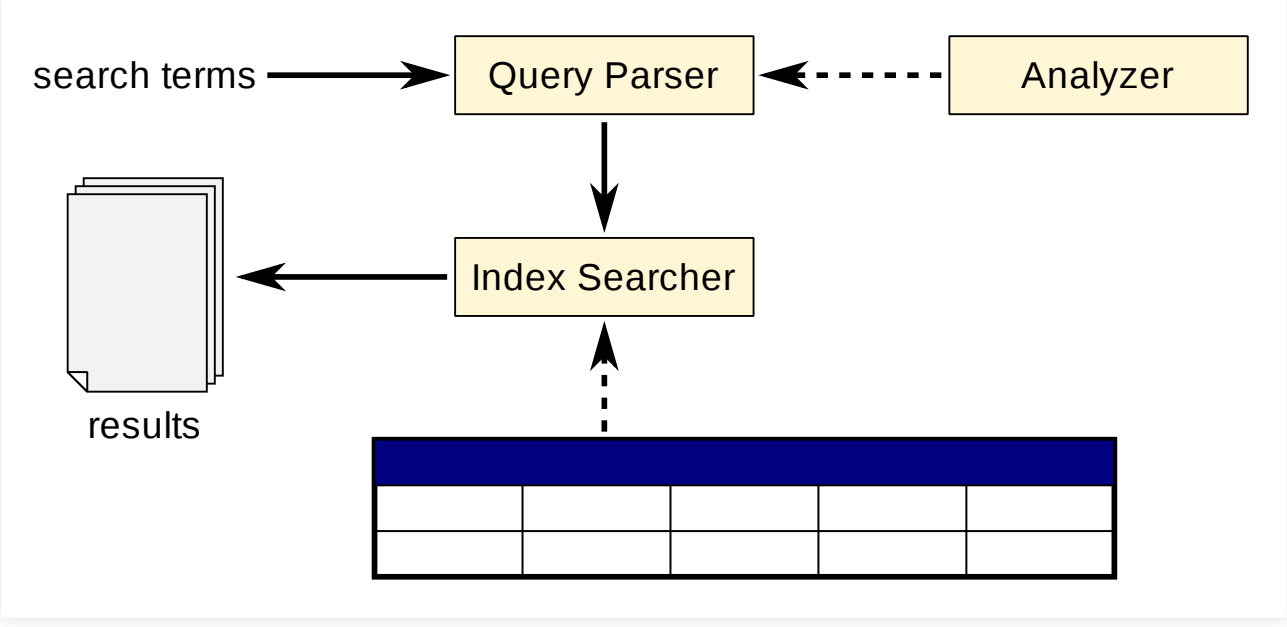
Shingling `org.apache.lucene.analysis.shingle.ShingleAnalyzerWrapper`

- 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 Demo: Indexing (1)

First we create and open the index:

```
// 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);
```

Lucene Demo: Indexing (2)

Then we add each document and each field in a document:

```
// 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);
```

Lucene Demo: Indexing (3)

Finally, we close the index:

```
// 1.8. close index writer
indexWriter.close();
dir.close();
```

Lucene Demo: Searching (1)

First, we create a query and open the index for search:

```
// 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);
```

Lucene Demo: Searching (2)

Then we search the query on the index and display the results:

```
// 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 + ")");
}
```

Finally, we close the index:

```
// 3.5. close index reader
indexReader.close();
dir.close();
```

Luke

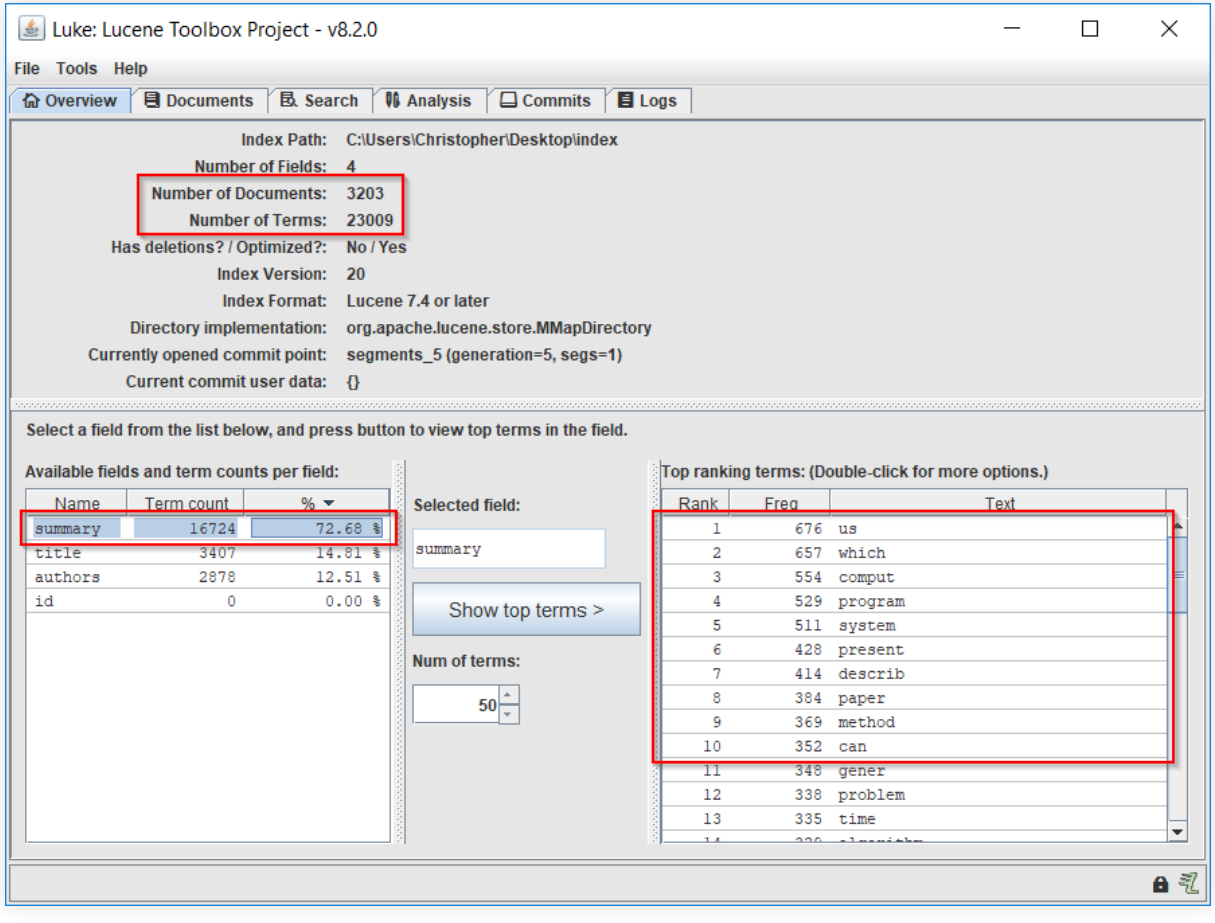
A GUI tool written in Java

Browse the contents of a Lucene index

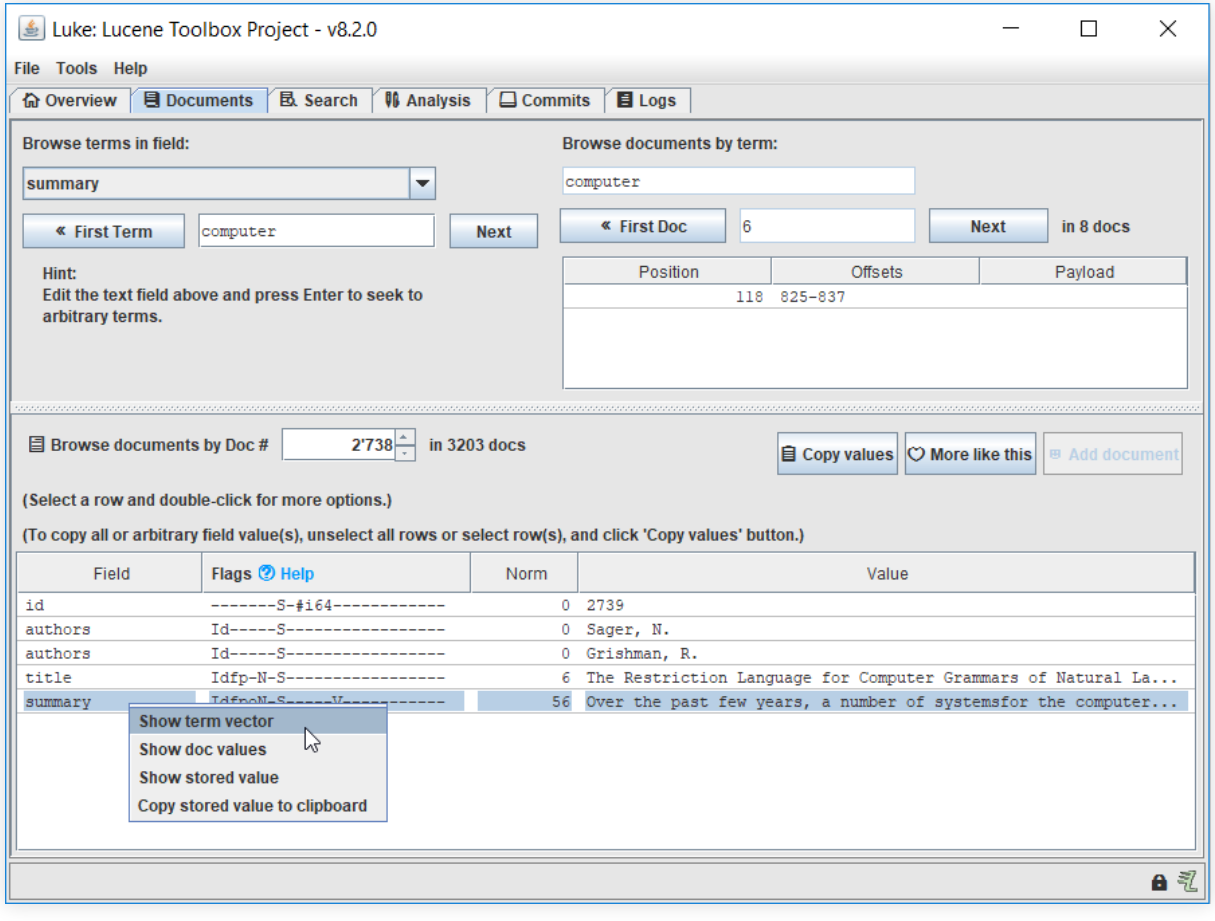
Examine individual documents

Run queries over the index

Luke: Index


Luke: Lucene Toolbox Project - v8.2.0
File Tools Help
Overview
Documents
Search
Analysis
Commits
Logs
Index Path: C:\Users\Christopher\Desktop\index
Number of Fields: 4
Number of Documents: 3203
Number of Terms: 23009
Has deletions? / Optimized?: No / Yes
Index Version: 20
Index Format: Lucene 7.4 or later
Directory implementation: org.apache.lucene.store.MMapDirectory
Currently opened commit point: segments_5 (generation=5, segs=1)
Current commit user data: {}
Select a field from the list below, and press button to view top terms in the field.
Available fields and term counts per field:
Name Term count %
summary 16724 72.68 %
title 3407 14.81 %
authors 2878 12.51 %
id 0 0.00 %
Selected field:
summary
Show top terms >
Num of terms:
50
Top ranking terms: (Double-click for more options.)
Rank Freq Text
1 676 us
2 657 which
3 554 comput
4 529 program
5 511 system
6 428 present
7 414 describ
8 384 paper
9 369 method
10 352 can
11 348 gener
12 338 problem
13 335 time

Luke: Documents



Luke: Term Vector

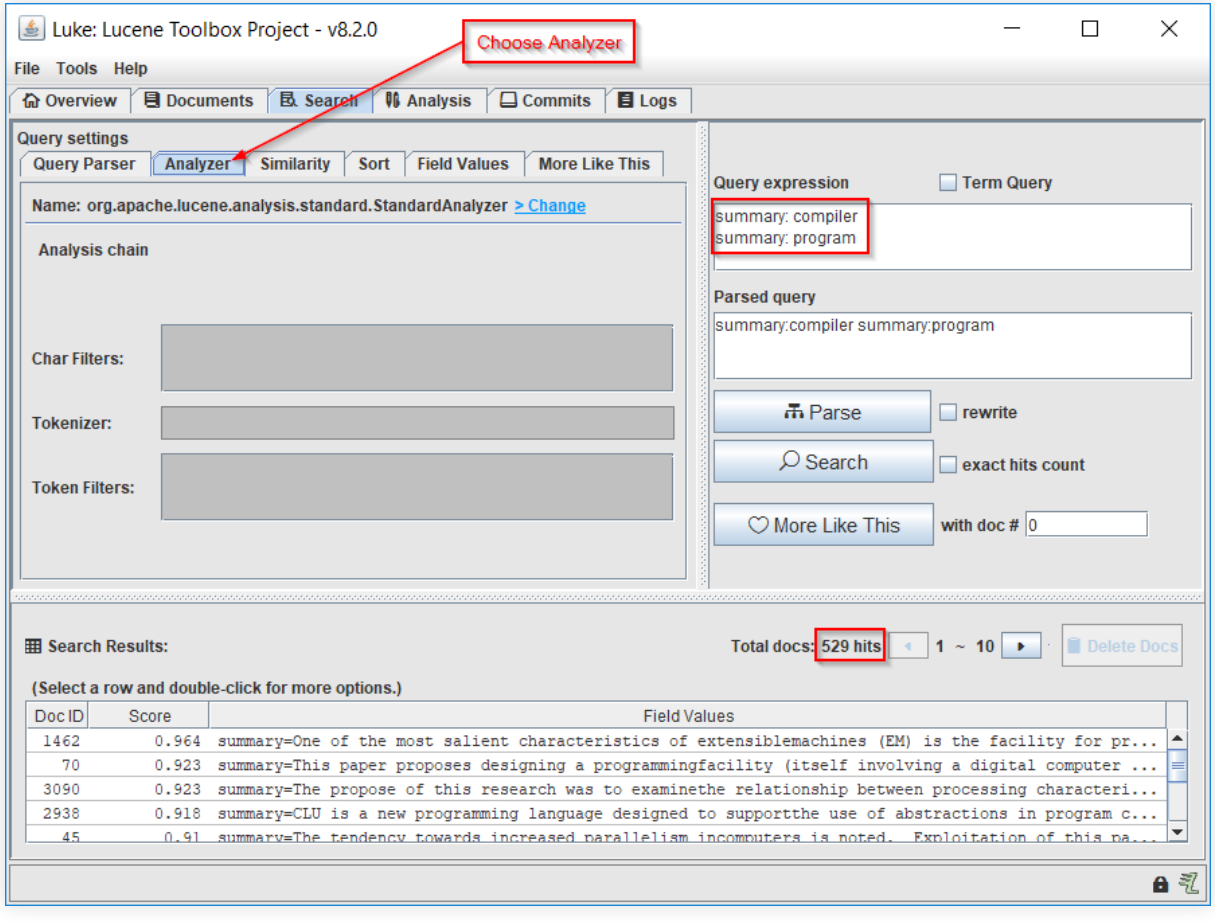
Term Vector

Term vector for field: summary

Term	Freq	Positions	Offsets
adequ	1	116	812-820
analysi	2	11,51	61-69,311-319
aset	1	28	187-191
augment	1	19	118-127
base	1	17	109-114
basedon	1	86	595-602
been	1	16	104-108
compact	1	74	508-515
comprehens	1	94	656-669
comput	1	10	52-60
computer	1	118	825-837
conform	1	44	272-279
context	2	20,23	128-135,152-159

Close

Luke: Search


Luke: Lucene Toolbox Project - v8.2.0
Choose Analyzer
File Tools Help
Overview
Documents
Search
Analysis
Commits
Logs
Query settings
Query Parser
Analyzer
Similarity
Sort
Field Values
More Like This
Name: org.apache.lucene.analysis.standard.StandardAnalyzer > Change
Analysis chain
Char Filters:
Tokenizer:
Token Filters:
Query expression
Term Query
summary: compiler
summary: program
Parsed query
summary:compiler summary:program
Parse
rewrite
Search
exact hits count
More Like This
with doc # 0
Search Results:
Total docs: 529 hits
1 ~ 10
Delete Docs
(Select a row and double-click for more options.)
Doc ID
Score
Field Values
1462 0.964 summary=One of the most salient characteristics of extensiblemachines (EM) is the facility for pr...
70 0.923 summary=This paper proposes designing a programmingfacility (itself involving a digital computer ...
3090 0.923 summary=The propose of this research was to examinethe relationship between processing characteri...
2938 0.918 summary=CLU is a new programming language designed to supportthe use of abstractions in program c...

References

Apache Lucene: http://lucene.apache.org/core/8_6_2/index.html

Tutorials

- https://www.ionos.fr/digitalguide/serveur/configuration/apache-lucene/