Introduction to

Information Retrieval

Lucene Tutorial

Overview

- Open source IR system
- Lucene Intro
- Installation
- Lucene core classes
- Scoring
- Index format
- Useful resources.

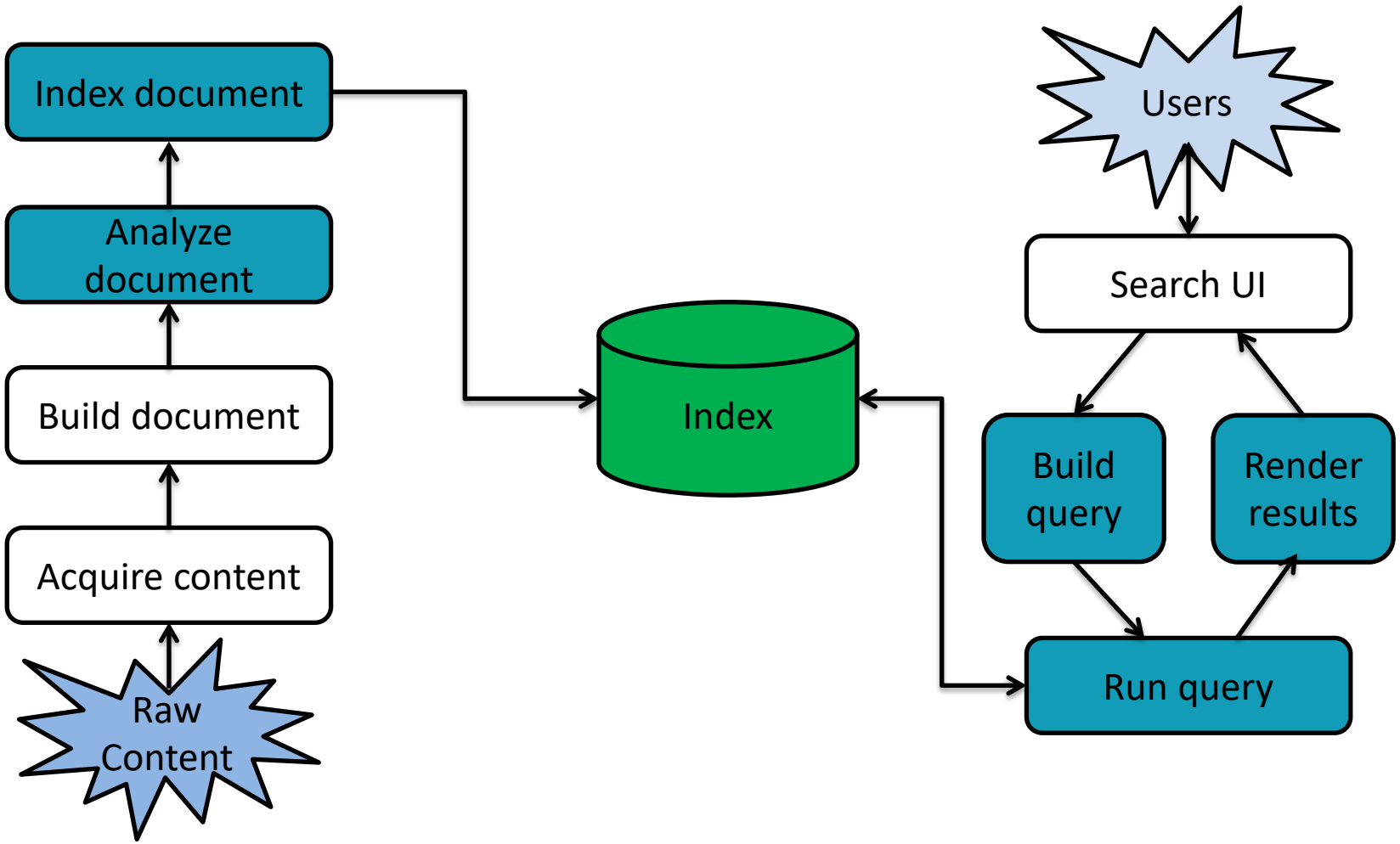
Open source IR systems

- Widely used academic systems
  - Terrier (Java, U. Glasgow) http://terrier.org
  - Indri/Galago/Lemur (C++ (& Java), U. Mass & CMU)
  - Zettair (RMIT)..
- Widely used non-academic open source systems
  - **Lucene**
    - Things built on it: Solr, ElasticSearch
  - A few others (Xapian, …)
  - Whoosh (python library)

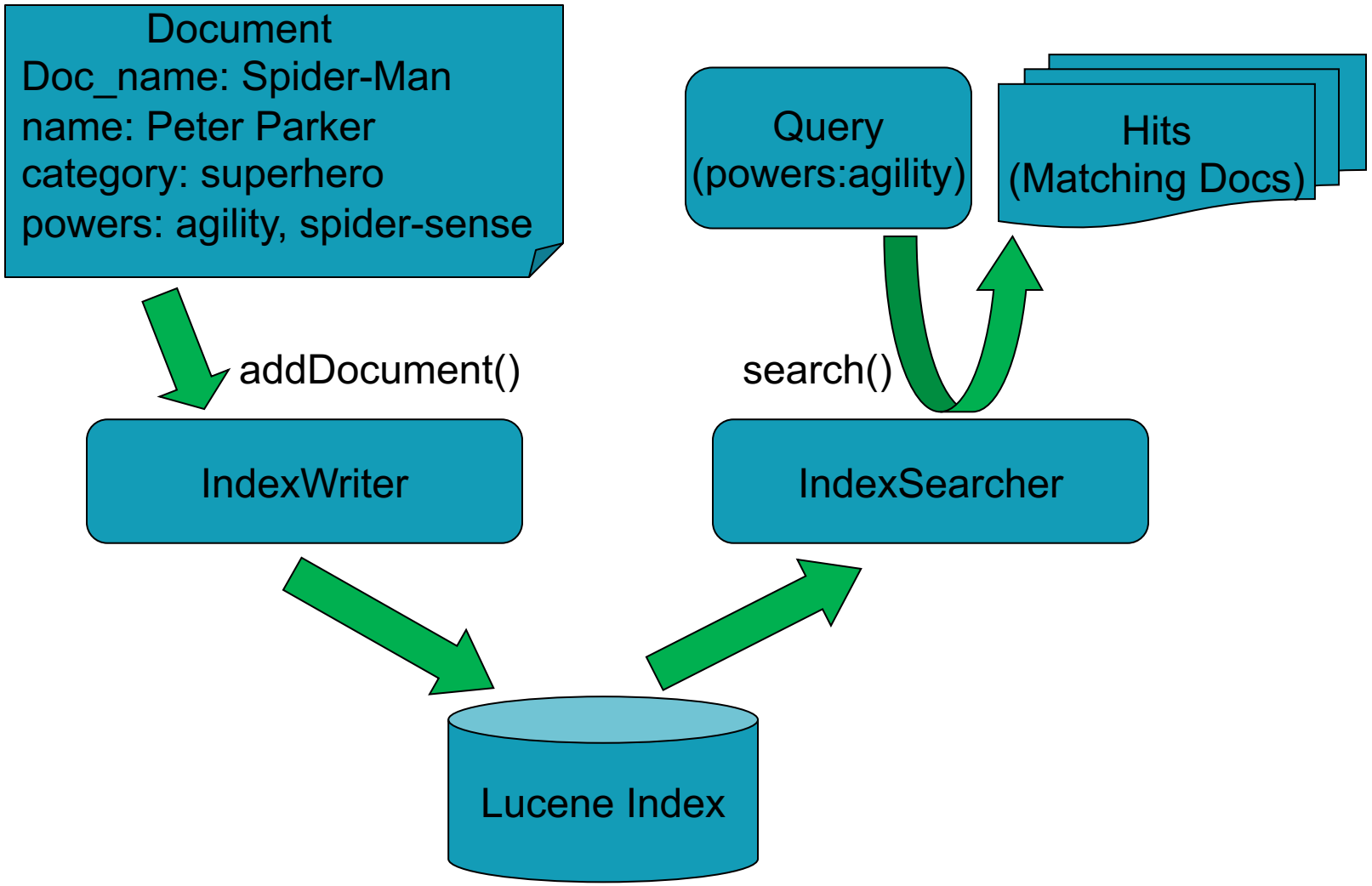
Lucene

- Open source Java library for indexing and searching
  - Lets you add search to your application
  - Not a complete search system by itself
  - Written by Doug Cutting
- Used by: Twitter, LinkedIn, Zappos, CiteSeer, Eclipse, …
  - … and many more (see http://wiki.apache.org/lucene-java/PoweredBy)
- Ports/integrations to other languages
  - C/C++, C#, Ruby, Perl, Python (Pylucene), PHP, …

Lucene in a search system



Basic Application



Installation

- Apache Lucene (http://lucene.apache.org/core/downloads)

  after downloading extract the files to the desktop.

  https://archive.apache.org/dist/lucene/java/7.2.1/

- JDK/JRE 7/8
- Eclipse

Add External Jar Files

- …\lucene-7.2.1\analysis\common\lucene-analyzers-common-7.2.1.jar
- …\lucene-7.2.1\core\lucene-core-7.2.1.jar
- …\lucene-7.2.1\demo\lucene-demo-5.0.0.jar
- …\lucene-7.2.1\queryparser\lucene-queryparser-7.2.1.jar

Demo Files

- Link to find the code for the search files and the index files
- Here you will find two files.
- IndexFiles.java- Code to create a Lucene index.
- SearchFiles.java- Code to search a Lucene Index.

https://lucene.apache.org/core/7_2_1/demo/overview-summary.html

Set directory Paths

- IndexFiles

  set docsPath= “Documents folder path”

  set indexPath = “Index folder path”

- SearchFiles

  set index= “Index folder path”

- Write a query
- Top results

https://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html

Core indexing classes

- `IndexWriter`
  - Central component that allows you to create a new index, open an existing one, and add, remove, or update documents in an index
  - Built on an `IndexWriterConfig` and a `Directory`
- `Directory`
  - Abstract class that represents the location of an index
- `Analyzer`
  - Extracts tokens from a text stream

Creating an `IndexWriter`

```
Import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.store.Directory;
...

private IndexWriter writer;

public Indexer(String dir) throws IOException {
    Directory indexDir = FSDirectory.open(new File(dir));
    Analyzer analyzer = new StandardAnalyzer();
    IndexWriterConfig cfg = new IndexWriterConfig(analyzer);
    cfg.setOpenMode(OpenMode.CREATE);
    writer = new IndexWriter(indexDir, cfg)
}
```

Core indexing classes (contd.)

- `Document`
  - Represents a collection of named `Fields`. Text in these `Fields` are indexed.
- `Field`
  - `StringFields` are indexed but not tokenized
  - `TextFields` are indexed and tokenized

A Document contains Fields

```
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
...
protected Document getDocument(File f) throws Exception {
    Document doc = new Document();
    doc.add(new TextField("contents", new FileReader(f)))
    doc.add(new StringField("filename",
                            f.getName(),
                            Field.Store.YES));
    doc.add(new StringField("fullpath",
                            f.getCanonicalPath(),
                            Field.Store.YES));
    return doc;
}
```

Index a Document with IndexWriter

```
private IndexWriter writer;
...
private void indexFile(File f) throws
     Exception {
   Document doc = getDocument(f);
   writer.addDocument(doc);
}
```

Indexing a directory

```
private IndexWriter writer;
...
public int index(String dataDir,
                 FileFilter filter)
      throws Exception {
   File[] files = new File(dataDir).listFiles();
   for (File f: files) {
      if (... &&
          (filter == null || filter.accept(f))) {
         indexFile(f);
      }
   }
   return writer.numDocs();
}
```

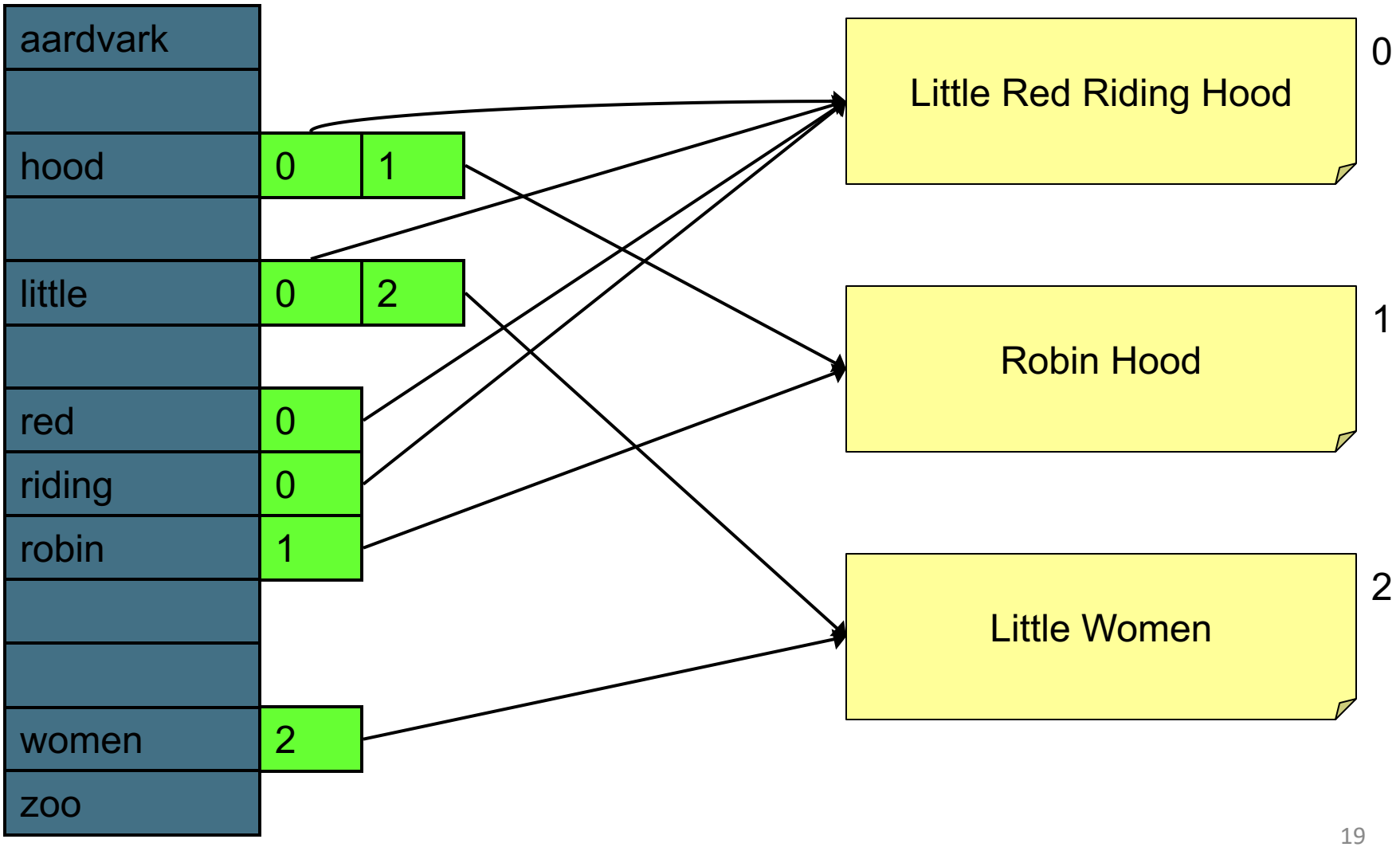
Closing the `IndexWriter`

```
private IndexWriter writer;
...
public void close() throws IOException {
   writer.close();
}
```

The Index

- The Index is the kind of inverted index we know and love
  - natural ordering of docIDs
  - encodes both term frequencies and positional information
- APIs to customize the codec

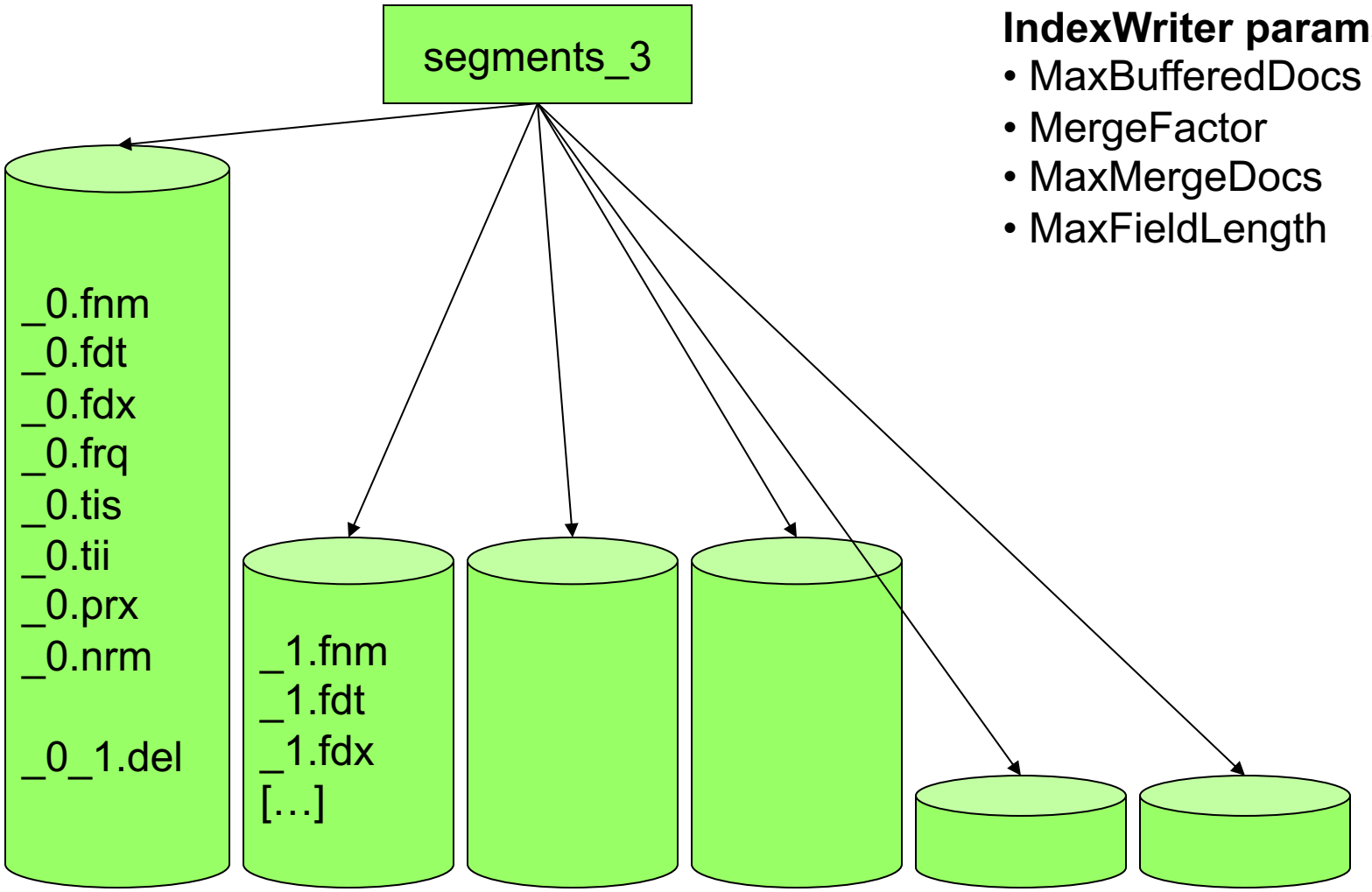
Inverted Index



Index format

- Each Lucene index consists of one or more segments
  - A segment is a standalone index for a subset of documents
  - All segments are searched
  - A segment is created whenever `IndexWriter` flushes adds/deletes
- Periodically, `IndexWriter` will merge a set of segments into a single segment
  - Policy specified by a `MergePolicy`
- You can explicitly invoke `forceMerge()` to merge segments

Index Structure



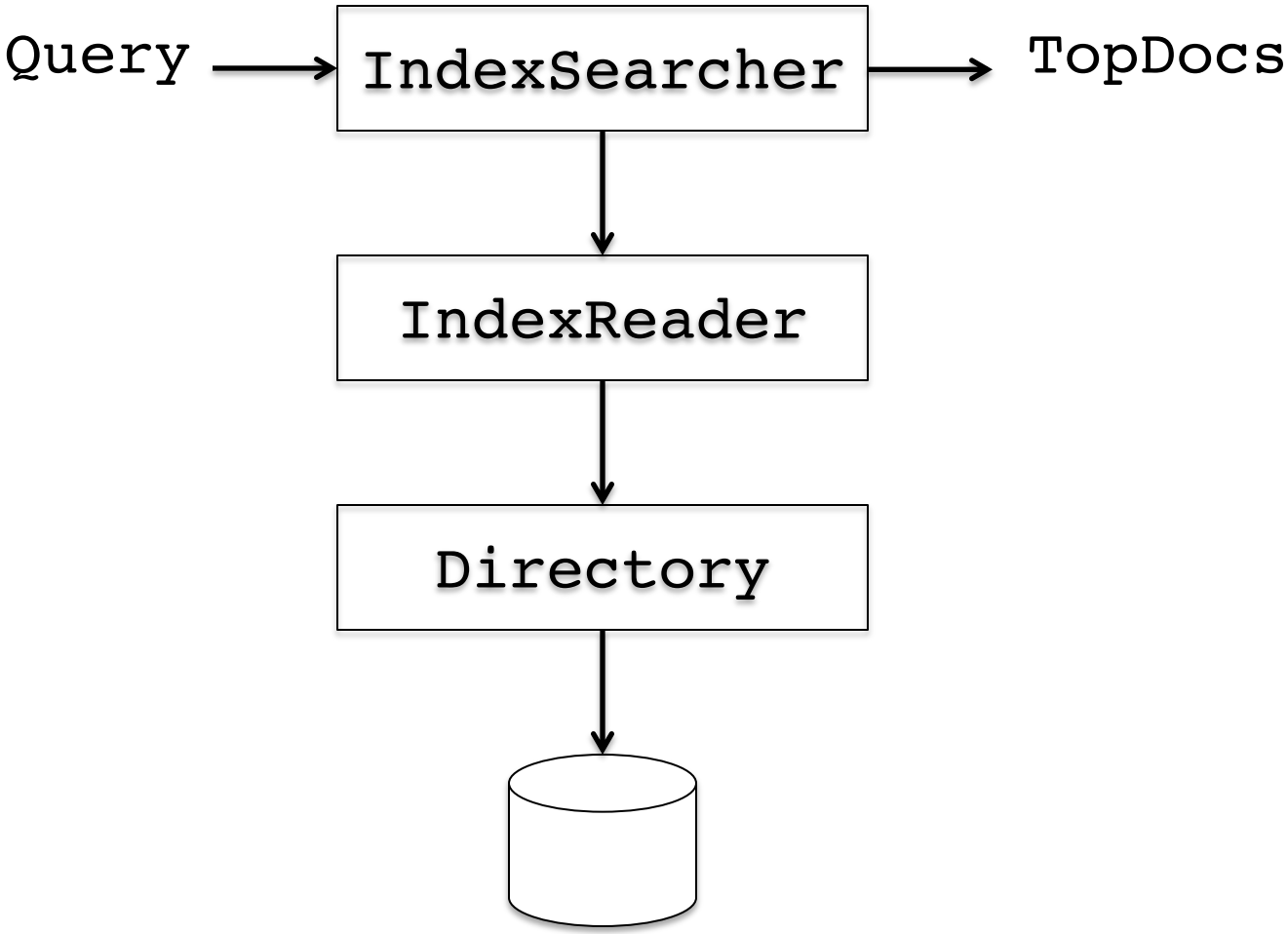
IndexWriter params

- MaxBufferedDocs
- MergeFactor
- MaxMergeDocs
- MaxFieldLength

Core searching classes

- `IndexSearcher`
  - Central class that exposes several search methods on an index
  - Accessed via an `IndexReader`
- `Query`
  - Abstract query class. Concrete subclasses represent specific types of queries, e.g., matching terms in fields, boolean queries, phrase queries, …
- `QueryParser`
  - Parses a textual representation of a query into a `Query` instance

IndexSearcher



Creating an IndexSearcher

```
import org.apache.lucene.search.IndexSearcher;
...
public static void search(String indexDir,
                          String q)
    throws IOException, ParseException {
  IndexReader rdr =
    DirectoryReader.open(FSDirectory.open(
                    new File(indexDir)));
  IndexSearcher is = new IndexSearcher(rdr);
  ...
}
```

Query and QueryParser

```
import org.apache.lucene.queryParser.QueryParser;
import org.apache.lucene.search.Query;
...
public static void search(String indexDir, String q)
      throws IOException, ParseException
   ...
   QueryParser parser =
      new QueryParser("contents”,
                      new StandardAnalyzer());
   Query query = parser.parse(q);
   ...
}
```

Core searching classes (contd.)

- `TopDocs`
  - Contains references to the top documents returned by a search
- `ScoreDoc`
  - Represents a single search result

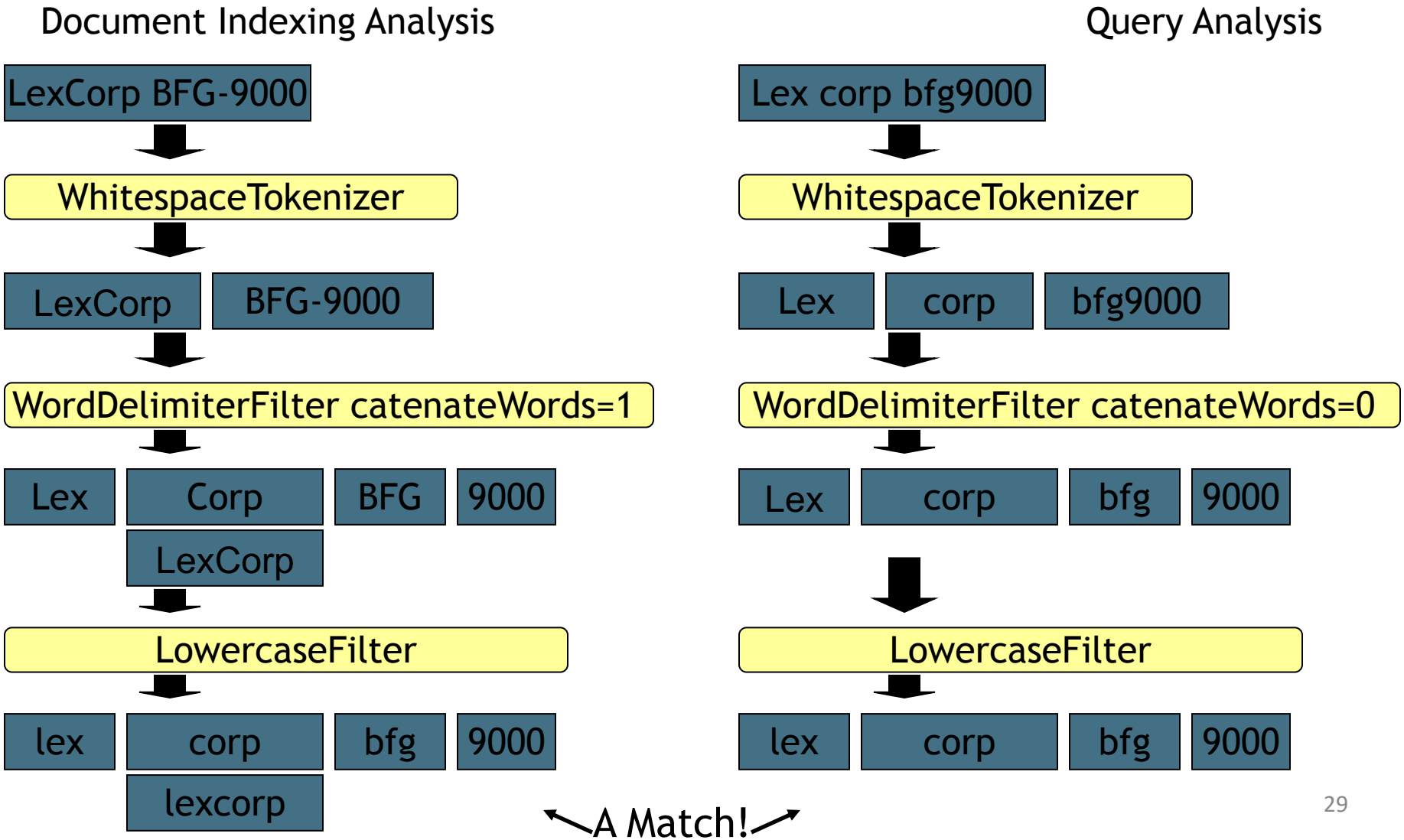
search( ) returns TopDocs

```
import org.apache.lucene.search.TopDocs;
...
public static void search(String indexDir,
                          String q)
    throws IOException, ParseException
  ...
  IndexSearcher is = ...;
  ...
  Query query = ...;
  ...
  TopDocs hits = is.search(query, 10);
}
```

Closing IndexSearcher

```
public static void search(String indexDir,
                          String q)
    throws IOException, ParseException
  ...
  IndexSearcher is = ...;
  ...
  is.close();
}
```

Analysis & Search Relevancy



Scoring

- VSM – Vector Space Model
- tf – term frequency: numer of matching terms in field
- lengthNorm – number of tokens in field
- idf – inverse document frequency
- coord – coordination factor, number of matching terms
- document boost
- query clause boost

http://lucene.apache.org/java/docs/scoring.html

Lucene 5.0 Scoring

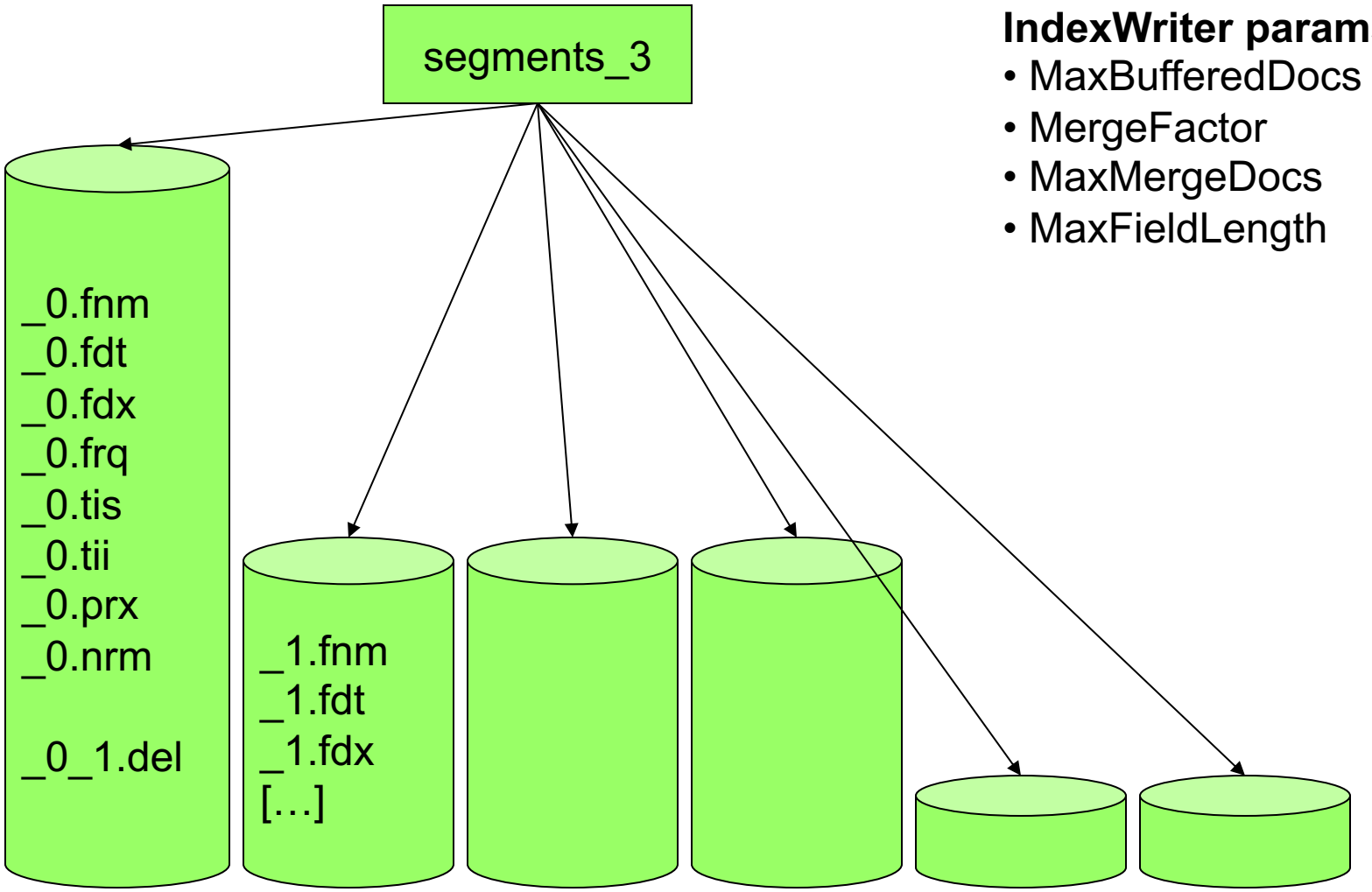
- As well as traditional tf.idf vector space model, Lucene 5.0 has:
  - BM25
  - drf (divergence from randomness)
  - ib (information (theory)-based similarity)

```
indexSearcher.setSimilarity(
      new BM25Similarity());
BM25Similarity custom =
      new BM25Similarity(1.2, 0.75); // k1, b
indexSearcher.setSimilarity(custom);
```

Index format

- Each Lucene index consists of one or more segments
  - A segment is a standalone index for a subset of documents
  - All segments are searched
  - A segment is created whenever `IndexWriter` flushes adds/deletes
- Periodically, `IndexWriter` will merge a set of segments into a single segment
  - Policy specified by a `MergePolicy`
- You can explicitly invoke `forceMerge()` to merge segments

Index Structure



IndexWriter params

- MaxBufferedDocs
- MergeFactor
- MaxMergeDocs
- MaxFieldLength

Useful Resources

- https://lucene.apache.org/core/7_0_1/index.html
- https://lucene.apache.org/core/3_5_0/scoring.html
- https://lucene.apache.org/core/2_9_4/fileformats.pdf
- https://lucene.apache.org/core/7_2_1/demo/overview-summary.html
- https://www.youtube.com/watch?v=pVDVURw_AJQ

Open Source Search Libraries

- Apache Lucene
- Apache Elastic Search
- Apache Solr
- Indri
- Whoosh (Python)