Frank Salamone
Technology Review
CS 410 Fall 2022

Apache Lucene, developed and maintained by the Apache Lucene project, is an open source Java library created in 1999 by Doug Cutting that provides indexing and search functionality. The library allows for fast indexing at rates of more than 800 G/Hr on modern hardware. It also has minimal RAM requirements, typically a 1MB heap. Fast incremental indexing is supported, and compressed indices are roughly 20-30% the size of the text indexed. Lucene forms the foundation for Elasticsearch (“Apache Lucene Core”), (edurekaIN).

A search engine allows the retrieval of documents based on a query. In order to find documents quickly, Lucene creates an inverted index containing a list of tokens, roughly corresponding to words, and the corresponding documents that contain these words. When responding to the query, the library doesn't have to search through all of the documents to find matching terms, it can just search the much shorter index.

In Lucene, documents consist of a collection of fields. Each field is indexed in a separate postings list. Fields are stored separately from the inverted index in “field storage” or .fdt files (Will 1). A field may be indexed and stored, indexed but not stored, or stored but not indexed. Indexing but not storing a field may be useful if you are indexing a large document such as a book. A search result only provides a list of document IDs (edurekaIN).

Lucene allows tokenizing of documents. Tokenization is a process by which a document is broken into tokens. These tokens can also be referred to as terms. Tokens are strings roughly corresponding to words, although more formally they can be described as “characters in some particular document that are grouped together as a useful semantic unit for processing.” In this process, extraneous words, punctuation and garbage characters are removed. (Manning)

After tokenization, in Lucene, an instance of a term is stored in a term vector consisting of the term, its position, offset, and length. This process is performed by an instance of the Analyzer class. A SimpleAnalyzer splits tokens at non-letter characters and then changes capital letters to lowercase. Numeric strings are ignored. StandardAnalyzer also removes stop words. The standard analyzer also recognizes emails and URLs (Will 1).

Lucene also supports an n-gram tokenizer. Using this indices are generated for unigrams, bigrams, trigrams, etc. This allows the index to be searched as a user types in query text. So, if the user types “comp” as they are typing computer, the index will allow searching for this partially entered term (Pavlov).

After tokinization, the terms are added to an inverted index created by an instance of the IndexWriter class. Lucene indexes subsets of documents and then merges these “runs” or “segments” using merge sort (edurekaIN). In the inverted index, the key is the individual term which accesses the term frequency and a list of the documents containing the term. The inverted index is then compressed and optimized. Once created, an index cannot be changed. When a document is deleted from the index, it is marked for deletion in a separate file. It is only actually deleted when the individual segment that containing the document is merged. All of this overhead makes Lucene inefficient for frequently updated documents (Will 2).

To create an index, in Java (from Will 2):

```
try {
	String indexPath = “c:\\myindex”;
	Directory dir = FSDirectory.open(indexPath);
	Analyzer analyzer = new StandardAnalyzer(Version.LUCENE_46);
	IndexWriterConfig iwc = new IndexWriterConfig(Version.LUCENE_46, analyzer);
	IndexWriter writer = new IndexWriter(dir, iwc)
	Document doc = new Document();
	doc.add(new TextFied(“content”, “blah blah bob”)
	writer.addDocument(doc);
	writer.close();
} except (IOException ex) {
	// report error
}
```

Lucene has an expression language that allows you to make queries. The expression is passed to a QueryParser that passes a Query object to an IndexSearcher object. Queries can consist of a number of query types including term, wildcard, prefix, fuzzy, phrase, range, and boolean queries. Each IndexReader only sees a snapshot of the index at the time of its creation (Will 2).

(From Will 2)

```
try{
	String indexPath = “c:\\myindex”;
	Directory dir = FSDirectory.open(indexPath);
	DirectoryReader reader = DirectoryReader.open(dir);
	IndexSearcher searcher = new IndexSearcher(reader);
	reader.close();
} except (IOException ex){
	// report error{
}

TermQuery tq = new TermQuery ((new Term(“content”, “doorknob”));
TopDocs  results = searcher.query(tq, 20);
for (ScoreDoc scoreDoc : results.scoreDocs) {
	System.out.println(
		“document ID: “ + Integer.toString(scoreDoc.doc)
	);
}
```

When doing a query, by default, Lucene uses a vector space model with TF-IDF weighting.

There are a number of libraries and systems associated with Lucene. PyLucene allows the use of Lucene with Python, and Lucene.NET is an implementation of Lucene in C#. Solr is a Java server that provides search services based on Lucene. Luke is a tool for looking at Lucene indexes.

Lucene is a powerful library that developers can use to integrate text search into their applications and websites. It forms the basis for many search services as it is fast, efficient, and easy to use. Apache

Lucene is powerful enough to have been used by services such as Twitter, Netflix, and Instagram. This review only touches on some of the capabilities of Lucene. Other capabilities include highlighting, language analysis, and much more (Białecki).

References:

"Apache Lucene Core." *Apache Lucene*, Apache Software Foundation, 2022, https://lucene.apache.org/core/.

Białecki, A., et al. *[PDF] Apache Lucene 4: Semantic Scholar*. https://www.semanticscholar.org/paper/Apache-Lucene-4-Bialecki-Muir/2795d9d165607b5ad6d8b9718373b82e55f41606.

edurekaIN. "Introduction to Apache Lucene | Why Lucene | Apache Lucene Tutorial | Edureka." *YouTube*, YouTube, 22 Oct. 2014, https://www.youtube.com/watch?v=vLEvmZ5eEz0&list=PLw5h0DiJ-9PBik3jfh1rfquL4J0iTdST8.

Manning, Christopher. *Tokenization*, https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html.

Pavlov, Ivaylo. "Introduction to Apache Lucene & Elasticsearch." *YouTube*, YouTube, 31 Jan. 2021, https://www.youtube.com/watch?v=BvgGgkN3clI.

Will, Brian. "Text Search with Lucene (1 of 2)." *YouTube*, YouTube, 4 Mar. 2014, https://www.youtube.com/watch?v=x37B_lCi_gc.

Will, Brian. "Text Search with Lucene (2 of 2)." *YouTube*, YouTube, 4 Mar. 2014, https://www.youtube.com/watch?v=fCK9U3L7c8U.