Apache Lucene

Tech Review by Akshat Agarwal
akshata4@illinois.edu

Lucene is a library for enabling text based search. It is most commonly used for implementing search engines and local single-site searching. It is open-source, highly scalable, fast and easy to integrate. Lucene is not a crawler or a search application.

In a traditional search engine application, there are the following components: the search UI, building the query, running the query, indexing, the index document, document analysing, rendering search results, document building, and document gathering. Out of these, Lucene provides the query building, running, indexing, index document, document analysis and rendering results.

Since Lucene only cares about text search, it does not work on document formats like PDF, XML, etc. However, there are extractors available that can simplify the data collection part of the integration, including the Apache Tika.

A Lucene index is a collection of documents. A document is the atomic unit of indexing and searching. A document consists of fields. Each field has some content and associated metadata. Each field may be indexed. It is up to the developer using Lucene to pick which fields to index. Lucene derives tokens from these indexed fields and then analyses them to form an index. The indexed fields can optionally store term vectors, which together form a miniature inverted index for that one field. This provides advanced capabilities like searching for “similar” documents. Fields can also just be stored, meaning the values are stored to be retrieved verbatim when searched for.

Lucene provides the user the power to perform optimisations on the search index in order to improve retrieval performance. It does come at a cost of resource utilisation though and should be done sparingly.

Lucene has some similarities to a database, but there is a key difference: Lucene has a flexible schema. Each document can have a different schema. A single index can hold data representing many different entities. Another difference is that Lucene requires you to flatten your content before indexing.

Originally Lucene could only operate on text fields, but recent versions have allowed numeric fields as well. Lucene also provides near-real-time search, meaning searching on documents immediately after they were added.

The Lucene Analysis consists of 3 types of classes: Analyzer, Tokenizer and TokenFilter. Their role is to together tokenize each document in order to create the index. There are some default choices like StandardAnalyzer, WhitespaceAnalyzer, etc. You also have the flexibility of also implementing your own analyser. The Tokenizer is responsible for chunking the input into Tokens.

TokenFilters can further modify the Tokens produced by the Tokenizer, including: removing them (stop words), stemming, etc.

Since Lucene is open source, it has a variety of contributed modules that solve some common problems like spell checking.

There is another analyser at work that parses the user's input query, which be plain text or some expression of the Lucene query syntax. This is referred to as the parser, and not the analyser, although you may choose to use the same analyser for both document analysis and query parsing. The Lucene Query Parser converts strings into Java objects that can be used for searching (See syntax). Query objects can also be constructed programmatically. There is native support for many types of queries like keyword, phrase, wildcard, etc.

Lucene also allows boosting documents or search fields. This kind of boosting is done during indexing and is preprocessed for storage efficiency. This is useful when we want to boost the scores of certain documents or certain field values over others.

Lucene is written in Java, but has bindings in other languages including C/C++, python, ruby, .NET and PHP. It provides a Boolean + Vector Space Model implementation of search. It uses the Boolean model to first narrow down the documents that need to be scored based on the use of boolean logic in the Query specification. It then uses the VSM implementation with TF-IDF weighting along with other modifications to provide for boosting, fuzziness and boolean searching to compute the score of documents (see Similarity for scoring formula)

Apache Lucene is a feature-complete, time-tested, extensible text search library with countless applications in the industry including Akamai, Netflix, LinkedIn, Technorati, HotJobs, Epiphany, FedEx, Mayo Clinic, MIT, New Scientist Magazine, Krugle, etc. Used in conjunction with other open source tools, it makes it very easy for developers to erect a fully functional and customised search engine over the dataset of their choice.

References

1. https://i-share-uiu.primo.exlibrisgroup.com/permalink/01CARLI_UIU/gpjosq/alma99954930874005899
2. https://web.archive.org/web/20120131154001/http://trijug.org/downloads/TriJug-11-07.pdf
3. https://lucene.apache.org/core/2_9_4/queryparsersyntax.html#Fuzzy+Searches
4. https://www.baeldung.com/lucene-analyzers
5. https://lucene.apache.org/core/4_9_0/core/org/apache/lucene/analysis/package-summary.html
6. https://lucene.apache.org/core/2_9_4/scoring.html