William Lupo

Technology Review – Apache Lucene

Apache Lucene is an open-source, high-performance, full-text search engine library written entirely in Java. At high level, the Lucene library executes two tasks – indexing and searching text documents. The specific APIs Lucene employs to enact indexing that will be examined in further detail include the IndexWriter, Document, Analyzer and Field. The document search and results ordering APIs to be examined include the QueryParser, TopDocs, IndexSearcher and the IndexReader. The final topic to be covered is a discussion on implementing Lucene within a serverless application as a means of abstracting the search engine to an API within a greater application ecosystem.

The basic units in which content is modeled by Lucene for both indexing and searching are in the form of documents and fields. A document is an "atomic unit of indexing and searching" and is the containing model that holds one or more fields. Fields are essentially key value pairs consisting of an identifier and the text in which Lucene will index and later search on. In order to implement an index on raw content, the text must be converted into Lucene's documents and fields. It is then these documents and their fields that will be queried.

Indexing text with Lucene is accomplished in three steps. Text is first extracted from the original content, the document is initialized, and the content to be indexed is stored within the document fields. The text within the fields is then passed through an analyzer to create a token stream. Lastly, the tokens generated from the analyzer are then stored to the index itself. Assuming that the data to be indexed and later searched is in a state to be programmatically manipulated, loading the content into Lucene documents and fields is straight forward. The number fields per document should align with the needs to the application that will submit and display queries. For example, when creating documents on a collection of tweets collected from Twitter, possible fields include the username of the tweet author as well as the content of tweet itself. This allows users to query Lucene for all tweets for a specific user as well as query the content of the tweets.

The analysis step within the indexing process of Lucene is comprised of converting document field text data into a collection of terms that will later be used to determine which documents match a search query. Regarding the Lucene API, an Analyzer is an abstraction of the tasks performed during this analysis phase. The analyzer is capable of performing any number of common text cleansing operations including "extracting words, discarding punctuation, removing accents from characters, lowercasing (also called normalizing), removing common words, reducing words to a root form (stemming), or changing words into the basic form (lemmatization)." Lucene provides five "out of the box" analyzers - WhitespaceAnalyzer, SimpleAnalyzer, StopAnalyzer, KeywordAnalyzer, and StandardAnalyzer. The StandardAnalyzer is the most verbose and general-purpose providing support for alphanumerics, acronyms, company names, email addresses, computer hostnames, numbers, words with an interior apostrophe, serial numbers, IP addresses, and Chinese and Japanese characters as well as stop-word removal. Since the features of an analyzer are derived from the data in which it is tokenizing, it is common to extend the Analyzer API in order to create custom functions in order to best tokenize the data for indexing.

The primary components to querying a Lucene index are the IndexSearcher, Query, QueryParser, TopDocs, and ScoreDoc. The IndexSearcher is the class that facilitates all query searches for a given index. QueryParser is responsible for translating human-entered queries into Query objects to be provided to the search method of the IndexSearcher. The result set of matching documents is then returned as an ordered array of ScoreDoc that provide the numeric relevance of each document to the query as well as the matching document identifier.