Technology Review - Lucene

1 Introduction

Lucene is a Java library providing powerful indexing and search features.

1.1 Classes

Lucene provides basic classes[1] that are relavant to the concepts introduced in our lectures.

Class	Description
Field	A Field has three parts: name, type, value.
Document	Unit of indexing and search. A Document is a set of Fields.
Directory	Provides an abstraction layer for storing a list of files. Usually represents the storage location of the index.
Analyzer	Represents a policy for extracting index terms from text. Usually consists of a tokenizer and a filter.
IndexWriter	Creates and maintains an index.
IndexSearcher	Searches the index.
Query	Base class for queries. Most used subclass is TermQuery.
QueryParser	Provides method parse(String), which returns a Query.

2 Indexing

2.1 Building Index

To build the index from text files, we need to:

1) Create a IndexWriter, which needs a Directory (for storing index on disk) and a Analyzer (tokenizer and filter), if no Analyzer is specified, by default StandardAnalyzer() would be used.

For each text file:

2) Create a Field with the text, then create a Document with the Field.

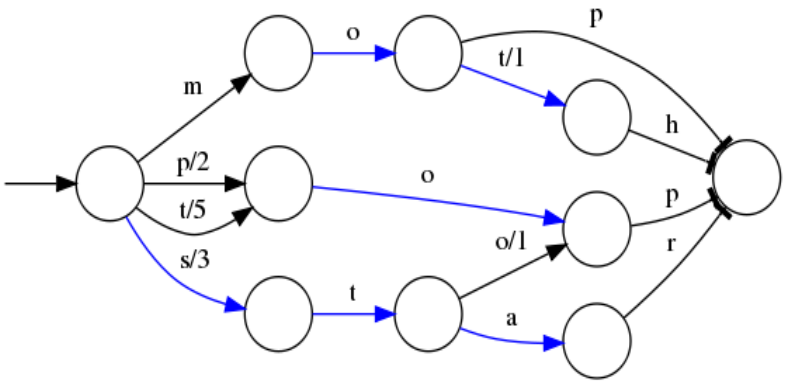
3) Add the Document to the IndexWriter.

2.2 Dictionary

An inverted index consists of a dictionary and postings. The dictionary is loaded in RAM and postings are stored on the disk. For the dictionary, a data structure like hash table, B-tree, trie.. is used for fash search.

Lucene uses Finite State Transducers (FST) to store the dictionary in RAM.[2]

1. Overview (Lucene 8.0.0 API) (apache.org)
2. Changing Bits: Using Finite State Transducers in Lucene (mikemccandless.com)



This is one possible FST for dictionary {mop: 0, moth: 1, pop: 2, star: 3, stop: 4, top: 5}. If we try to lookup for "stop", we follow arcs $s/3 \rightarrow t \rightarrow o/1 \rightarrow p$, and it sums up to $3+1=4$.

Compared to a hash table, a FST should be slower since hash table takes O(1) lookup time and FST takes O(len(str)) time to traverse len(str) arcs. However FSTs need less memory since prefixs/suffixes can be shared between terms.

Memory usuage is more important than speed in this case since a dictionary is usually large with millions of terms. While speed of dictionary lookup is less important since we expect the time spent on postings lookup (on disk) would be significantly longer. The overall query execution time is dominated by postings lookup.

2.3 Segment

Lucene indexes may be composed of multiple segments. Each segment is a fully independent index, which could be searched separately.[3]

Once a segment is created, it cannot be modified (i.e. read-only). We can only add new terms to a new segment. Deletion is achieved by keeping a record of deleted terms, then filter out deleted terms when executing queries.

Immutable segments are cache friendly and multithread friendly, but it may waste disk space and drag the query, especially when the deletion record is large (i.e. only small portion of segment is not deleted). Luence will merge some segments to improve performance.

3 Search

IndexSearcher provides search(Query q, int k) function for finding the top-k hits for query q. Users can set the Similarity class used by IndexSearcher. Similarity defines how Lucene weights terms. By default, BM25Similarity is used.

To implement custom scoring function, users can extend Similarity class or SimilarityBase class. SimilarityBase is simplified API where users only need to implement function score(BasicStats stats, double freq, double docLen) function. Similarity allows user to define per-document value at index time via computeNorm(FieldInvertState). It is usally used for encoding length normalization information.[4]

3. org.apache.lucene.codecs.lucene80 (Lucene 8.0.0 API)
4. Similarity (Lucene 8.0.0 API) (apache.org)

4 Summary

Lucene offers powerful indexing and search features, which are the building blocks of some popular open source search engine like Solr and Elasticsearch. Lucene provides scalable indexing features by using FST to store dictionary and managing read/write by segments. Also Lucene provides highly customizable search features for users to implement their own scoring function.

5 Reference

1. Overview (Lucene 8.0.0 API) (apache.org)
2. Changing Bits: Using Finite State Transducers in Lucene (mikemccandless.com)
3. org.apache.lucene.codecs.lucene80 (Lucene 8.0.0 API) (Apache Lucene - Index File Formats)
4. Similarity (Lucene 8.0.0 API) (apache.org)
5. Lucene Tutorial (tutorialspoint.com)
6. algorithm - How does lucene index documents? - Stack Overflow