Open-Source Search Engines and Lucene/Solr

UCSB 293S, 2017. Tao Yang

Slides are based on Y. Seeley, S. Das, C. Hostetter
Open Source Search Engines

• Why?
  ▪ Low cost: No licensing fees
  ▪ Source code available for customization
  ▪ Good for modest or even large data sizes

• Challenges:
  ▪ Performance, Scalability
  ▪ Maintenance
Open Source Search Engines: Examples

• **Lucene**
  - A full-text search library with core indexing and search services
  - Competitive in engine performance, relevancy, and code maintenance

• **Solr**
  - based on the Lucene Java search library with XML/HTTP APIs
  - caching, replication, and a web administration interface.

• **Lemur/Indri**
  - C++ search engine from U. Mass/CMU
# A Comparison of Open Source Search Engines

- **Middleton/Baeza-Yates** 2010 (Modern Information Retrieval. Text book)

<table>
<thead>
<tr>
<th>Search Engine</th>
<th>Storage</th>
<th>Incremental Index</th>
<th>Results Excerpt</th>
<th>Results Template</th>
<th>Stop Words</th>
<th>Filetype</th>
<th>Stemming</th>
<th>Fuzzy Search</th>
<th>Sort (d)</th>
<th>Ranking</th>
<th>Search Type (c)</th>
<th>Indexer Lang (b)</th>
<th>License (c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datapark</td>
<td>2</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3</td>
<td>■</td>
<td>■</td>
<td>1, 2</td>
<td>■</td>
<td>2</td>
<td>1, 2, 3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>ht://Dig</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1</td>
<td>2, 1, 2</td>
<td>2, 1, 2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Indri</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3, 4</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3</td>
<td>2, 3</td>
<td>1, 2, 4</td>
<td>2, 3, 8</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>IXE</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3</td>
<td>2, 8</td>
<td>1, 2, 3</td>
<td>2, 8</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Lucene</td>
<td>1</td>
<td>■, ■</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3</td>
<td>■</td>
<td>■</td>
<td>2, 1, 2</td>
<td>3, 1</td>
<td>1, 2, 3</td>
<td>3, 6</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>MG4J</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3</td>
<td>1</td>
<td>2, 1, 2</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>mnoGoSearch</td>
<td>2</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>1</td>
<td>2, 1</td>
<td>1, 2, 3</td>
<td>1, 2, 4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Namazu</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3</td>
<td>1</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Omega</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1, 2, 4, 5</td>
<td>■</td>
<td>■</td>
<td>1</td>
<td>1, 2</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>OmniFind</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3, 4, 5</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>3, 5</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>OpenFTS</td>
<td>2</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>1</td>
<td>1, 2</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>SWISH-E</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3</td>
<td>1</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>SWISH++</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3</td>
<td>1</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Terrier</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3, 4, 5</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>WebGlimpse</td>
<td>1</td>
<td>■, ■</td>
<td>■</td>
<td>■</td>
<td>1, 2</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>8, 9</td>
<td>1</td>
</tr>
<tr>
<td>XMLSearch</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>1, 2</td>
<td>1</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Zettair</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>1</td>
<td>■</td>
<td>■</td>
<td>1, 2, 3</td>
<td>1</td>
<td>1, 2, 3</td>
<td>1, 2, 3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

(a) 1:Apache, 2:BSD, 3:CMU, 4:GPL, 5:IBM, 6:LGPL, 7:MPL, 8:Comm, 9:Free  
(b) 1:C, 2:C++, 3:Java, 4:Perl, 5:PHP, 6:Tcl  
(c) 1:phrase, 2:boolean, 3: wild card.  
(d) 1:ranking, 2:date, 3:none.  
(e) 1:HTML, 2:plain text, 3:XML, 4:PDF, 5:PS.  
(f) 1:file, 2:database.  
(g) Commercial version only.
A Comparison of Open Source Search Engines for 1.69M Pages

- Middleton/Baeza-Yates 2010 (Modern Information Retrieval)

<table>
<thead>
<tr>
<th>Search Engine</th>
<th>Indexing Time (h:m:s)</th>
<th>Index Size (%)</th>
<th>Searching Time (ms)</th>
<th>Answer Quality P@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>ht://Dig</td>
<td>(7) 0:28:30</td>
<td>(10) 104</td>
<td>(6) 32</td>
<td>-</td>
</tr>
<tr>
<td>Indri</td>
<td>(4) 0:15:45</td>
<td>(9) 63</td>
<td>(2) 19</td>
<td>(2) 0.2851</td>
</tr>
<tr>
<td>IXE</td>
<td>(8) 0:31:10</td>
<td>(4) 30</td>
<td>(2) 19</td>
<td>(5) 0.1429</td>
</tr>
<tr>
<td>Lucene</td>
<td>(10) 1:01:25</td>
<td>(2) 26</td>
<td>(4) 21</td>
<td>-</td>
</tr>
<tr>
<td>MG4J</td>
<td>(3) 0:12:00</td>
<td>(8) 60</td>
<td>(5) 22</td>
<td>(4) 0.2480</td>
</tr>
<tr>
<td>Swish-E</td>
<td>(5) 0:19:45</td>
<td>(5) 31</td>
<td>(8) 45</td>
<td>-</td>
</tr>
<tr>
<td>Swish++</td>
<td>(6) 0:22:15</td>
<td>(3) 29</td>
<td>(10) 51</td>
<td>-</td>
</tr>
<tr>
<td>Terrier</td>
<td>(9) 0:40:12</td>
<td>(7) 52</td>
<td>(9) 50</td>
<td>(3) 0.2800</td>
</tr>
<tr>
<td>XMLSearch</td>
<td>(2) 0:10:35</td>
<td>(1) 22</td>
<td>(1) 12</td>
<td>-</td>
</tr>
<tr>
<td>Zettair</td>
<td>(1) 0:04:44</td>
<td>(6) 33</td>
<td>(6) 32</td>
<td>(1) 0.3240</td>
</tr>
</tbody>
</table>

Table 6.1: Ranking of search engines, comparing their indexing time, index size, and the average searching time (for the 2.7GB collection), and the Answer Quality for the engines that parsed the WT10g. The number in parentheses corresponds to the relative position of the search engine.
## A Comparison of Open Source Search Engines


<table>
<thead>
<tr>
<th>Platform</th>
<th>License</th>
<th>Lang.</th>
<th>Docs</th>
<th>Ranking</th>
<th>Users</th>
<th>Support</th>
<th>Parallel</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lucene</td>
<td>Apache</td>
<td>Java</td>
<td>Many</td>
<td>Flexible</td>
<td>Amazon</td>
<td>5/5</td>
<td>Yes</td>
<td>TB</td>
</tr>
<tr>
<td>zettair</td>
<td>BSD like</td>
<td>C</td>
<td>HTML, TREC, TXT</td>
<td>Flexible</td>
<td>Research</td>
<td>1/5</td>
<td>No</td>
<td>TB</td>
</tr>
<tr>
<td>Indri</td>
<td>BSD like</td>
<td>C++</td>
<td>Many</td>
<td>Very Flexible</td>
<td>Research</td>
<td>1.5/5</td>
<td>Yes</td>
<td>TB</td>
</tr>
<tr>
<td>Sphinx</td>
<td>GPL</td>
<td>C++</td>
<td>Many</td>
<td>Flexible</td>
<td>craigslist</td>
<td>4/5</td>
<td>Yes</td>
<td>TB</td>
</tr>
<tr>
<td>RDBMS</td>
<td>BSD, GPL</td>
<td>C</td>
<td>SQL Text</td>
<td>Limited</td>
<td>-</td>
<td>3/5</td>
<td>Maybe</td>
<td>GB</td>
</tr>
<tr>
<td>Xapian</td>
<td>GPL</td>
<td>C++</td>
<td>Many</td>
<td>Flexible</td>
<td>gmane</td>
<td>3/5</td>
<td>Yes</td>
<td>TB</td>
</tr>
</tbody>
</table>
A Comparison of Open Source Search Engines

- Vik's blog(http://zooie.wordpress.com/2009/07/06/a-comparison-of-open-source-search-engines-and-indexing-twitter/)

TREC Filtering OHSUMED Data Set
63 Topics = Queries (“37 yr old man with sickle cell disease”); Avg. Len: 6.7; OR’ed
196,403 Medical Results (300MB Indexable Text)
Judgement Data: (Topic, Result, 2 or 1 or 0 Rating)
Relevancy: DCG 10

<table>
<thead>
<tr>
<th>Platform</th>
<th>Index Peak Memory</th>
<th>Index Time</th>
<th>Index Size</th>
<th>Search Peak Memory</th>
<th>Search Time</th>
<th>Relevancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lucene 2.4.1</td>
<td>37 MB</td>
<td>2m15s</td>
<td>91 MB</td>
<td>18 MB</td>
<td>0.02168s (1.366s)</td>
<td>1.0449</td>
</tr>
<tr>
<td>zettair 0.9.3</td>
<td>22 MB</td>
<td>0m29.34s</td>
<td>122 MB</td>
<td>9 MB</td>
<td>0.02609s (1.644s)</td>
<td>0.8299</td>
</tr>
<tr>
<td>sphinx 0.9.8.1</td>
<td>19 MB</td>
<td>0m42.35s</td>
<td>201 MB</td>
<td>16 MB</td>
<td>0.00803s (0.506s)</td>
<td>0.7690</td>
</tr>
<tr>
<td>sqlite 3.6.11</td>
<td>8 MB</td>
<td>1m54.91s</td>
<td>474 MB</td>
<td>7 MB</td>
<td>0.91451s (54.614s)</td>
<td>0.0166</td>
</tr>
<tr>
<td>Xapian 1.0.13</td>
<td>48 MB</td>
<td>6m38.17s</td>
<td>339 MB</td>
<td>1 MB</td>
<td>0.02286s (1.440s)</td>
<td>1.0162</td>
</tr>
</tbody>
</table>
Lucene

• **Developed by Doug Cutting initially**
  - Java-based. Created in 1999, Donated to Apache in 2001

• **Features**
  - No crawler, No document parsing, No “PageRank”

• **Powered by Lucene**
  - IBM Omnifind Y! Edition, Technorati
  - Wikipedia, Internet Archive, LinkedIn, monster.com

• **Add documents to an index via IndexWriter**
  - A document is a collection of fields
  - Flexible text analysis – tokenizers, filters

• **Search for documents via IndexSearcher**
  \[ \text{Hits} = \text{search}(\text{Query, Filter, Sort, topN}) \]

• **Ranking based on \( tf \times idf \) similarity with normalization**
Lucene’s input content for indexing

- **Logical structure**
  - Documents are a collection of fields
    - Stored – Stored verbatim for retrieval with results
    - Indexed – Tokenized and made searchable
  - Indexed terms stored in inverted index

- **Physical structure of inverted index**
  - Multiple documents stored in segments

- **IndexWriter** is interface object for entire index
Example of Inverted Indexing

- aardvark
- hood: 0, 1
- little: 0, 2
- red: 0
- riding: 0
- robin: 1
- women: 2
- zoo

Titles:
- Little Red Riding Hood
- Robin Hood
- Little Women
Faceted Search/Browsing Example

### DESKTOPS

You found 1045 items for System type: **Budget desktop system**

Too few results? Click a link above to remove that filter, or remove all filters.

<table>
<thead>
<tr>
<th>Find by price</th>
<th>Find by manufacturer</th>
<th>Find by processor manufacturer</th>
<th>Or find by</th>
</tr>
</thead>
<tbody>
<tr>
<td>‣ Less than $400 (76)</td>
<td>‣ Dell, Inc. (43)</td>
<td>‣ Intel (804)</td>
<td>‣ Clock speed</td>
</tr>
<tr>
<td>‣ $400 to $699 (337)</td>
<td>‣ Lenovo (490)</td>
<td>‣ AMD (122)</td>
<td>‣ Graphics processor</td>
</tr>
<tr>
<td>‣ $700 to $999 (468)</td>
<td>‣ HP (342)</td>
<td>‣ Clock speed</td>
<td>‣ RAM installed</td>
</tr>
<tr>
<td>‣ $1000 to $1299 (5)</td>
<td>‣ Acer America Corp. (28)</td>
<td>‣ Hard drive size</td>
<td>‣ Hard drive size</td>
</tr>
<tr>
<td></td>
<td>‣ Cyberpower Inc (22)</td>
<td>‣ OS provided</td>
<td></td>
</tr>
<tr>
<td></td>
<td>‣ See all manufacturers</td>
<td>‣ See all</td>
<td></td>
</tr>
</tbody>
</table>

Sort by: Product name | Lowest price | Editors' rating | Review date

---

**Dell Dimension B110 Desktop Computer for Home (Cel-D 2.53GHz/160GB/512MB)**

Dell's entry-level Dimension B110 series features aging technology and a dated design, but its members will suffice as second PCs for basic tasks.

**Specs:** Celeron D (2.53 GHz), 512 MB, 160 GB, 15 in, Microsoft Windows XP Home Edition

**$479**
at 1 store

**Add to my products** New! What is this?

**Dell Dimension B110 Desktop Computer for Home (Cel-D 2.53GHz/80GB/256MB)**

Dell's entry-level Dimension B110 series features aging technology and a dated design, but its members will suffice as second PCs for basic tasks.

**Specs:** Celeron D (2.53 GHz), 80 MB, 80 GB, 15 in, Microsoft Windows XP Home Edition

**$349**
at 1 store

**Add to my products** New! What is this?
Indexing Flow

LexCorp BFG-9000

WhitespaceTokenizer

LexCorp BFG-9000

WordDelimiterFilter catenateWords=1

Lex Corp BFG-9000

LowercaseFilter

lex corp bfg 9000 lexcorp
Analyzers specify how the text in a field is to be indexed

- Options in Lucene
  - **WhitespaceAnalyzer**
    - divides text at whitespace
  - **SimpleAnalyzer**
    - divides text at non-letters
    - convert to lower case
  - **StopAnalyzer**
    - SimpleAnalyzer
    - removes stop words
  - **StandardAnalyzer**
    - good for most European Languages
    - removes stop words
    - convert to lower case
  - **Create you own Analyzers**
**Lucene Index Files: Field infos file (.fnm)**

<table>
<thead>
<tr>
<th>Format:</th>
<th>FieldsCount, <code>&lt;FieldName, FieldBits&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldsCount</td>
<td>the number of fields in the index</td>
</tr>
<tr>
<td>FieldName</td>
<td>the name of the field in a string</td>
</tr>
<tr>
<td>FieldBits</td>
<td>a byte and an int where the lowest</td>
</tr>
<tr>
<td></td>
<td>bit of the byte shows whether the</td>
</tr>
<tr>
<td></td>
<td>field is indexed, and the int is the</td>
</tr>
<tr>
<td></td>
<td>id of the term</td>
</tr>
</tbody>
</table>

1, `<content, 0x01>`

[http://lucene.apache.org/core/3_6_2/fileformats.html](http://lucene.apache.org/core/3_6_2/fileformats.html)
<table>
<thead>
<tr>
<th>Format:</th>
<th>TermCount, TermInfos</th>
</tr>
</thead>
<tbody>
<tr>
<td>TermInfos</td>
<td>&lt;Term, DocFreq&gt;</td>
</tr>
<tr>
<td>Term</td>
<td>&lt;PrefixLength, Suffix, FieldNum&gt;</td>
</tr>
</tbody>
</table>

This file is sorted by Term. Terms are ordered first lexicographically by the term's field name, and within that lexicographically by the term's text.

<table>
<thead>
<tr>
<th>TermCount</th>
<th>the number of terms in the documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term</td>
<td>Term text prefixes are shared. The PrefixLength is the number of initial characters from the previous term which must be pre-pended to a term's suffix in order to form the term's text. Thus, if the previous term's text was &quot;bone&quot; and the term is &quot;boy&quot;, the PrefixLength is two and the suffix is &quot;y&quot;.</td>
</tr>
<tr>
<td>FieldNumber</td>
<td>the term's field, whose name is stored in the .fnm file</td>
</tr>
</tbody>
</table>

4,<<0,football,1>,2> <<0,penn,1>, 1> <<1,layers,1>,1> <<0,state,1>,2> Document Frequency can be obtained from this file.
## Lucene Index Files: Term Info Index (.tii)

<table>
<thead>
<tr>
<th>Format</th>
<th>IndexTermCount, IndexInterval, TermIndices</th>
</tr>
</thead>
<tbody>
<tr>
<td>TermIndices</td>
<td>&lt;TermInfo, IndexDelta&gt;</td>
</tr>
</tbody>
</table>

This contains every $\text{IndexInterval}^{th}$ entry from the .tis file, along with its location in the "tis" file. This is designed to be read entirely into memory and used to provide random access to the "tis" file.

- **IndexDelta** determines the position of this term's TermInfo within the .tis file. In particular, it is the difference between the position of this term's entry in that file and the position of the previous term's entry.

4, <football,1> <penn,3> <layers,2> <state,1>
### Lucene Index Files: Frequency file (.frq)

<table>
<thead>
<tr>
<th>Format:</th>
<th>&lt;TermFreqs&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>TermFreqs</td>
<td>TermFreq</td>
</tr>
<tr>
<td>TermFreq</td>
<td>DocDelta, Freq?</td>
</tr>
</tbody>
</table>

TermFreqs are ordered by term (the term is implicit, from the .tis file). TermFreq entries are ordered by increasing document number.

**DocDelta** determines both the document number and the frequency. In particular, DocDelta/2 is the difference between this document number and the previous document number (or zero when this is the first document in a TermFreqs). When DocDelta is odd, the frequency is one. When DocDelta is even, the frequency is read as the next Int.

For example, the TermFreqs for a term which occurs once in document seven and three times in document eleven would be the following sequence of Ints: 15, 8, 3

\[[7, 1] [11, 3] \rightarrow [DocIDDelta = 7, Freq = 1] [DocIDDelta = 4 \ (11-7) \ , \ Freq = 3] \rightarrow (7 << 1) | 1 = 15 \quad \text{and} \quad (4 << 1) | 0 = 8 \rightarrow [DocDelta = 15] [DocDelta = 8, Freq = 3]\]

http://hackerlabs.org/blog/2011/10/01/hacking-lucene-the-index-format/
### Lucene Index Files: Position file (.prx)

<table>
<thead>
<tr>
<th>Format:</th>
<th><code>&lt;TermPositions&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>TermPositions</td>
<td><code>&lt;Positions&gt;</code></td>
</tr>
<tr>
<td>Positions</td>
<td><code>&lt;PositionDelta&gt;</code></td>
</tr>
</tbody>
</table>

TermPositions are ordered by term (the term is implicit, from the .tis file). Positions entries are ordered by increasing document number (the document number is implicit from the .frq file).

PositionDelta is the difference between the position of the current occurrence in the document and the previous occurrence (or zero, if this is the first occurrence in this document).

For example, the TermPositions for a term which occurs as the fourth term in one document, and as the fifth and ninth term in a subsequent document, would be the following sequence of Ints: 4, 5, 4
Query Syntax and Examples

- **Terms with fields and phrases**
  - Title:right and text: go
  - Title:right and go (go appears in default field “text”)
  - Title: “the right way” and go

- **Proximity**
  - “quick fox”~4

- **Wildcard**
  - pla?e (plate or place or plane)
  - practic* (practice or practical or practically)

- **Fuzzy (edit distance as similarity)**
  - planting~0.75 (granting or planning)
  - roam~ (default is 0.5)
Query Syntax and Examples

- **Range**
  - `date:[05072007 TO 05232007]` (inclusive)
  - `author: {king TO mason}` (exclusive)

- **Ranking weight boosting** ^
  - `title:“Bell” author:“Hemmingway”^3.0`
  - Default boost value 1. May be <1 (e.g 0.2)

- **Boolean operators: AND, "+", OR, NOT and "-"**
  - “Linux OS” AND system
  - Linux OR system, Linux system
  - +Linux system
  - +Linux –system

- **Grouping**
  - Title: (+linux +”operating system”)

Searching: Example

- Document analysis
  - LexCorp BFG-9000
    - WhitespaceTokenizer
      - LexCorp BFG-9000
        - WordDelimiterFilter catenateWords=1
          - Lex Corp BFG 9000
            - LowercaseFilter
              - lex corp bfg 9000

- Query analysis
  - Lex corp bfg9000
    - WhitespaceTokenizer
      - Lex Corp bfg9000
        - WordDelimiterFilter catenateWords=0
          - Lex corp bfg 9000
            - LowercaseFilter
              - lex corp bfg 9000

A Match!
Searching

- **Concurrent search query handling:**
  - Multiple searchers at once
  - Thread safe
- **Additions or deletions to index are not reflected in already open searchers**
  - Must be closed and reopened
- **Use commit or optimize on indexWriter**
Query Processing

- Query
  - Term Dictionary (Random file access)
  - Term Info Index (in Memory)
    - Field info (in Memory)
    - Frequency File (Random file access)
    - Position File (Random file access)

All interactions are constant time.
Factors involved in Lucene's scoring

- **tf** = term frequency in document = measure of how often a term appears in the document
- **idf** = inverse document frequency = measure of how often the term appears across the index
- **coord** = number of terms in the query that were found in the document
- **lengthNorm** = measure of the importance of a term according to the total number of terms in the field
- **queryNorm** = normalization factor so that queries can be compared
- **boost (index)** = boost of the field at index-time
- **boost (query)** = boost of the field at query-time

[http://lucene.apache.org/core/3_6_2/scoring.html](http://lucene.apache.org/core/3_6_2/scoring.html)
Scoring Function is specified in schema.xml

- **Similarity**
  
  $$\text{score}(Q,D) = \text{coord}(Q,D) \cdot \text{queryNorm}(Q) \cdot \sum_{t \in Q} (\text{tf}(t \in D) \cdot \text{idf}(t)^2 \cdot \text{t.getBoost()} \cdot \text{norm}(D))$$

- **term-based factors**
  
  - $\text{tf}(t \in D)$: term frequency of term $t$ in document $d$
    - default $\sqrt{\text{raw term frequency}}$
  
  - $\text{idf}(t)$: inverse document frequency of term $t$ in the entire corpus
    - default $\ln[N_{Docs}/(\text{docFreq} - 1)] + 1$
Default Scoring Functions for query Q in matching document D

- \( \text{coord}(Q,D) = \text{overlap between } Q \text{ and } D \div \text{maximum overlap} \)
  Maximum overlap is the maximum possible length of overlap between Q and D

- \( \text{queryNorm}(Q) = 1/\text{sum of square weight}^{\frac{1}{2}} \)
  \( \text{sum of square weight} = q.\text{getBoost()}^2 \cdot \sum_{t \in Q} (\text{idf}(t) \cdot t.\text{getBoost()})^2 \)
  If \( t.\text{getBoost() } = 1, \text{ and } q.\text{getBoost()} = 1 \)
  Then, \( \text{sum of square weight} = \sum_{t \in Q} (\text{idf}(t))^2 \)
  thus, \( \text{queryNorm}(Q) = 1/(\sum_{t \in Q} (\text{idf}(t))^2)^{\frac{1}{2}} \)

- \( \text{norm}(D) = 1/\text{number of terms}^{\frac{1}{2}} \) (This is the normalization by the total number of terms in a document. Number of terms is the total number of terms appeared in a document \( D \).)
Example:

\[
\text{score}(Q,D) = \text{coord}(Q,D) \cdot \text{queryNorm}(Q) \cdot \sum_{t \in Q}(\text{tf}(t \in D) \cdot \text{idf}(t)^2 \cdot t.\text{getBoost()} \cdot \text{norm}(D))
\]

- D1: hello, please say hello to him.
- D2: say goodbye
- Q: you say hello

- \(\text{coord}(Q, D) = \text{overlap between } Q \text{ and } D / \text{maximum overlap}\)
  - \(\text{coord}(Q, D1) = \frac{2}{3}, \text{coord}(Q, D2) = \frac{1}{2}\),

- \(\text{queryNorm}(Q) = \frac{1}{\text{sum of square weight}^{\frac{1}{2}}}\)
  - \(\text{sum of square weight} = q.\text{getBoost()}^2 \cdot \sum_{t \in Q}(\text{idf}(t) \cdot t.\text{getBoost()}^2)\)
  - \(t.\text{getBoost()} = 1, q.\text{getBoost()} = 1\)
  - \(\text{sum of square weight} = \sum_{t \in Q}(\text{idf}(t)^2)\)
  - \(\text{queryNorm}(Q) = \frac{1}{(0.5945^2 + 1^2)^{\frac{1}{2}}} = 0.8596\)

- \(\text{tf}(t \in d) = \text{frequency}^{\frac{1}{2}}\)
  - \(\text{tf}(\text{you},D1) = 0, \text{tf}(\text{say},D1) = 1, \text{tf}(\text{hello},D1) = 2^{\frac{1}{2}} = 1.4142\)
  - \(\text{tf}(\text{you},D2) = 0, \text{tf}(\text{say},D2) = 1, \text{tf}(\text{hello},D2) = 0\)

- \(\text{idf}(t) = \ln \left(\frac{N}{(n_j+1)}\right) + 1\)
  - \(\text{idf}(\text{you}) = 0, \text{idf}(\text{say}) = \ln(2/(2+1)) + 1 = 0.5945, \text{idf}(\text{hello}) = \ln(2/(1+1)) + 1 = 1\)

- \(\text{norm}(D) = \frac{1}{\text{number of terms}^{\frac{1}{2}}}\)
  - \(\text{norm}(D1) = \frac{1}{6^{\frac{1}{2}}} = 0.4082, \text{norm}(D2) = \frac{1}{2^{\frac{1}{2}}} = 0.7071\)

- Score(Q, D1) = \(2/3 \times 0.8596 \times (1 \times 0.5945^2 + 1.4142 \times 1^2) \times 0.4082 = 0.4135\)
- Score(Q, D2) = \(1/2 \times 0.8596 \times (1 \times 0.5945^2) \times 0.7071 = 0.1074\)
Lucene Sub-projects or Related

- **Nutch**
  - Web crawler with document parsing
- **Hadoop**
  - Distributed file systems and data processing
  - Implements MapReduce
- **Solr**
- **Zookeeper**
  - Centralized service (directory) with distributed synchronization
Solr

- Developed by Yonik Seeley at CNET. Donated to Apache in 2006

**Features**
- Servlet, Web Administration Interface
- XML/HTTP, JSON Interfaces
- Faceting, Schema to define types and fields
- Highlighting, Caching, Index Replication (Master / Slaves)
- Pluggable. Java

**Powered by Solr**
- Netflix, CNET, Smithsonian, GameSpot, AOL:sports and music
- Drupal module
Architecture of Solr

Admin Interface
- Standard Request Handler
- Disjunction Max Request Handler
- Custom Request Handler
- XML Response Writer

HTTP Request Servlet

Update Servlet
- XML Update Interface

Solr Core
- Config
- Schema
- Caching
- Concurrency

Analysis

Lucene

Replication

Update Handler
Application usage of Solr: YouSeer search [PennState]

Crawling (Heritrix) → Parsing → Indexing/Searching (Solr)

File System → FS Crawler → PDF HTML DOC TXT...

WWW → Crawl (Heritrix) → PDF parser → PDF parser → HTML parser

YouSeer

Soli Documents

Stop Analyzer
Standard Analyzer
Your Analyzer

Index

Indexer

Searcher

Searching
Adding Documents in Solr

HTTP POST to /update

<add><doc boost="2">
  <field name="article">05991</field>
  <field name="title">Apache Solr</field>
  <field name="subject">An intro…</field>
  <field name="category">search</field>
  <field name="category">lucene</field>
  <field name="body">Solr is a full…</field>
</doc></add>
Updating/Deleting Documents

• Inserting a document with already present uniqueKey will erase the original

• Delete by uniqueKey field (e.g. Id)

<delete><id>05591</id></delete>

• Delete by Query (multiple documents)

<delete>
  <query>manufacturer:microsoft</query>
</delete>
Commit

• `<commit/>` makes changes visible
  ▪ closes IndexWriter
  ▪ removes duplicates
  ▪ opens new IndexSearcher
    – newSearcher/firstSearcher events
    – cache warming
    – “register” the new IndexSearcher

• `<optimize/>` same as commit, merges all index segments.
Default Query Syntax

Lucene Query Syntax

1. mission impossible; releaseDate desc
2. +mission +impossible –actor:cruise
3. “mission impossible” –actor:cruise
4. title:spiderman^10 description:spiderman
5. description:“spiderman movie”~10
6. +HDTV +weight:[0 TO 100]
7. Wildcard queries: te?t, te*t, test*
# Default Parameters

## Query Arguments for HTTP GET/POST to /select

<table>
<thead>
<tr>
<th>param</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td></td>
<td>The query</td>
</tr>
<tr>
<td>start</td>
<td>0</td>
<td>Offset into the list of matches</td>
</tr>
<tr>
<td>rows</td>
<td>10</td>
<td>Number of documents to return</td>
</tr>
<tr>
<td>fl</td>
<td>*</td>
<td>Stored fields to return</td>
</tr>
<tr>
<td>qt</td>
<td>standard</td>
<td>Query type; maps to query handler</td>
</tr>
<tr>
<td>df</td>
<td>(schema)</td>
<td>Default field to search</td>
</tr>
</tbody>
</table>
Search Results

http://localhost:8983/solr/select?q=video&start=0&rows=2&fl=name,price

<response><responseHeader><status>0</status><QTime>1</QTime></responseHeader><result numFound="16173" start="0">
  <doc>
    <str name="name">Apple 60 GB iPod with Video</str>
    <float name="price">399.0</float>
  </doc>
  <doc>
    <str name="name">ASUS Extreme N7800GTX/2DHTV</str>
    <float name="price">479.95</float>
  </doc>
</result></response>
Schema

- Lucene has no notion of a schema
  - Sorting - string vs. numeric
  - Ranges - val:42 included in val:[1 TO 5]?
  - Lucene QueryParser has date-range support, but must guess.
- Defines fields, their types, properties
- Defines unique key field, default search field, Similarity implementation
Field Definitions

• **Field Attributes:** name, type, indexed, stored, multiValued, omitNorms

```xml
<field name="id" type="string" indexed="true" stored="true"/>
<field name="sku" type="textTight" indexed="true" stored="true"/>
<field name="name" type="text" indexed="true" stored="true"/>
<field name="reviews" type="text" indexed="true" stored="false"/>
<field name="category" type="text_ws" indexed="true" stored="true" multiValued="true"/>
```

Stored means retrievable during search

• **Dynamic Fields, in the spirit of Lucene!**

```xml
<dynamicField name="*_i" type="sint" indexed="true" stored="true"/>
<dynamicField name="*_s" type="string" indexed="true" stored="true"/>
<dynamicField name="*_t" type="text" indexed="true" stored="true"/>
```
Schema: Analyzers

```xml
<fieldtype name="nametext" class="solr.TextField">
  <analyzer class="org.apache.lucene.analysis.WhitespaceAnalyzer"/>
</fieldtype>

<fieldtype name="text" class="solr.TextField">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StandardFilterFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory"/>
    <filter class="solr.PorterStemFilterFactory"/>
  </analyzer>
</fieldtype>

<fieldtype name="myfieldtype" class="solr.TextField">
  <analyzer>
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.SnowballPorterFilterFactory" language="German"/>
  </analyzer>
</fieldtype>
```
<fieldtype name="text" class="solr.TextField">
  <analyzer>
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.SynonymFilterFactory"
      synonyms="synonyms.txt"/>
    <filter class="solr.StopFilterFactory"
      words="stopwords.txt"/>
    <filter class="solr.EnglishPorterFilterFactory"
      protected="protwords.txt"/>
  </analyzer>
</fieldtype>
Search Relevancy

Document Analysis

PowerShot SD 500

WhitespaceTokenizer

PowerShot  SD  500

WordDelimiterFilter catenateWords=1

Power  Shot  SD  500

PowerShot

LowercaseFilter

power  shot  sd  500

powershot

A Match!

Query Analysis

power-shot sd500

WhitespaceTokenizer

power-shot  sd500

WordDelimiterFilter catenateWords=0

power  shot  sd  500

powershot

LowercaseFilter

power  shot  sd  500

A Match!
copyField

• Copies one field to another at index time
• Use case: Analyze same field different ways
  ▪ copy into a field with a different analyzer
  ▪ boost exact-case, exact-punctuation matches
  ▪ language translations, thesaurus, soundex

    <field name="title" type="text"/>
    <field name="title_exact" type="text_exact" stored="false"/>
    <copyField source="title" dest="title_exact"/>

• Use case: Index multiple fields into single searchable field
Faceted Search/Browsing Example

You found 1045 items for System type: Budget desktop system
Too few results? Click a link above to remove that filter, or remove all filters.

Find by price
- Less than $400 (76)
- $400 to $699 (337)
- $700 to $999 (468)
- $1000 to $1299 (5)

Find by manufacturer
- Dell, Inc. (43)
- Lenovo (490)
- HP (342)
- Acer America Corp. (28)
- Cyberpower Inc (22)
- See all manufacturers

Find by processor manufacturer
- Intel (804)
- AMD (122)
- Motorola (1)

Or find by
- Clock speed
- Graphics processor
- RAM installed
- Hard drive size
- OS provided
- See all

Sort by: Product name | Lowest price | Editors' rating | Review date

Dell Dimension B110 Desktop Computer for Home (Cel-D 2.53GHz/160GB/512MB)
Dell's entry-level Dimension B110 series features aging technology and a dated design, but its members will suffice as second PCs for basic tasks.
Specs: Celeron D (2.53 GHz), 512 MB, 160 GB, 15 in, Microsoft Windows XP Home Edition

Price: $479
at 1 store

Add to my products
New! What is this?

Dell Dimension B110 Desktop Computer for Home (Cel-D 2.53GHz/80GB/256MB)
Dell's entry-level Dimension B110 series features aging technology and a dated design, but its members will suffice as second PCs for basic tasks.
Specs: Celeron D (2.53 GHz), 256 MB, 80 GB, 15 in, Microsoft Windows XP Home Edition

Price: $349
at 1 store
Faceted Search/Browsing

\begin{itemize}
  \item \textbf{Search} (Query, Filter[], Sort, offset, n)
  \item \textbf{DocList}
  \item \textbf{DocSet}
  \item \textbf{Query Response}
\end{itemize}
High Availability

Load Balancer

Appservers

Solr Searchers

Solr Master

Index Replication

admin queries

updates

admin terminal

Updater

DB

Dynamic HTML Generation

HTTP search requests

Updates

46
Distribution + Replication

Distributed + Replication

Shard 1 Master

Shard 2 Master

Shard 3 Master

slave1

slave2

slave1

slave1

slave2

slave2
Caching

IndexSearcher’s view of an index is fixed

- Aggressive caching possible
- Consistency for multi-query requests

- **filterCache** – unordered set of document ids matching a query. key=Query, val=DocSet
- **resultCache** – ordered subset of document ids matching a query. key=(Query,Sort,Filter), val=DocList
- **documentCache** – the stored fields of documents. key=docid, val=Document
- **userCaches** – application specific, custom query handlers. key=Object, val=Object
Warming for Speed

• **Lucene IndexReader warming**
  - field norms, FieldCache, tii – the term index

• **Static Cache warming**
  - Configurable static requests to warm new Searchers

• **Smart Cache Warming (autowarming)**
  - Using MRU items in the current cache to pre-populate the new cache

• **Warming in parallel with live requests**
Smart Cache Warming

On-Deck Solr IndexSearcher

User Cache

Filter Cache

Result Cache

Doc Cache

Request Handler

Warming Requests

Live Requests

Registered Solr IndexSearcher

User Cache

Filter Cache

Result Cache

Doc Cache

Regenerator

Autowarming

Autowarming – warm n MRU cache keys w/ new Searcher
Web Admin Interface

• Show Config, Schema, Distribution info
• Query Interface
• Statistics
  ▪ Caches: lookups, hits, hitratio, inserts, evictions, size
  ▪ RequestHandlers: requests, errors
  ▪ UpdateHandler: adds, deletes, commits, optimizes
  ▪ IndexReader, open-time, index-version, numDocs, maxDocs,
• Analysis Debugger
  ▪ Shows tokens after each Analyzer stage
  ▪ Shows token matches for query vs index
Solr Admin (example)

SEELEYXP.cnet.cnwk:8983
cwd=f:\code\solr\example SolrHome=solr/

Solr
   [SCHEMA] [CONFIG] [ANALYSIS]
   [STATISTICS] [INFO] [DISTRIBUTION] [PING] [LOGGING]

App server:
   [JAVA PROPERTIES] [THREAD DUMP]

Make a Query
   [FULL INTERFACE]

StyleSheet:
Query:
solr

Search

Assistance
   [DOCUMENTATION] [ISSUE TRACKER] [SEND EMAIL]
   [LUCENE QUERY SYNTAX]

Current Time: Mon Jun 05 15:38:08 EDT 2006
Server Start At: Mon Jun 05 15:37:59 EDT 2006

Done
References

- [http://lucene.apache.org/]()
- [http://lucene.apache.org/core/3_6_2/gettingstarted.html](http://lucene.apache.org/core/3_6_2/gettingstarted.html)
- [http://people.apache.org/~yonik/presentations/](http://people.apache.org/~yonik/presentations/)