

Lucene in Action

Second Edition

MICHAEL MCCANDLESS
ERIK HATCHER
OTIS GOSPODNETIC



MANNING
Greenwich
(74° w. long.)

brief contents

PART 1	CORE LUCENE.....	1
	1 • Meet Lucene	3
	2 • Building a search index	31
	3 • Adding search to your application	74
	4 • Lucene's analysis process	110
	5 • Advanced search techniques	152
	6 • Extending search	204
PART 2	APPLIED LUCENE.....	233
	7 • Extracting text with Tika	235
	8 • Essential Lucene extensions	255
	9 • Further Lucene extensions	288
	10 • Using Lucene from other programming languages	325
	11 • Lucene administration and performance tuning	345
PART 3	CASE STUDIES.....	381
	12 • Case study LKrugle	383
	13 • Case study 2: SIREn	394
	14 • Case study 3: LinkedIn	409

contents

foreword xvii
preface xix
preface to the first edition xx
acknowledgments xxiii
about this book xxvi
JUnit primer xxxiv
about the authors xxxvii

PART 1 CORE LUCENE1

I *Meet Lucene 3*

1.1 Dealing with information explosion 4

1.2 What is Lucene? 6

What Lucene can do 7 • History of Lucene 7

1.3 Lucene and the components of a search application 9

Components for indexing 11 • Components for searching 14
*The rest of the search application 16 * Where Lucene fits into your application 18*

1.4 Lucene in action: a sample application 19

Creating an index 19 Searching an index 23*

1.5 Understanding the core indexing classes 25

IndexWriter 26 • Directory 26 • Analyzer 26
Document 27 • Field 27

- 1.6 Understanding the core searching classes 28
 - IndexSearcher* 28 • *Term* 28 • *Query* 29 * *TermQuery* 29
 - TopDocs* 29
- 1.7 Summary 29

Building a search index 31

- 2.1 How Lucene models content 32
 - Documents and fields* 32 • *Flexible schema* 33
 - Denormalization* 34
- 2.2 Understanding the indexing process 34
 - Extracting text and creating the document* 34
 - Analysis* 35 • *Adding to the index* 35
- 2.3 Basic index operations 36
 - Adding documents to an index* 37 * *Deleting documents from an index* 39 • *Updating documents in the index* 41
- 2.4 Field options 43
 - Field options for indexing* 43 • *Field options for storing fields* 44
 - Field options for term vectors* 44 • *Reader, TokenStream, and byte[]field values* 45 • *Field option combinations* 46 * *Field options for sorting* 46 • *Multivalued fields* 47
- 2.5 Boosting documents and fields 48
 - Boosting documents* 48 * *Boosting fields* 49 * *Norms* 50
- 2.6 Indexing numbers, dates, and times 51
 - Indexing numbers* 51 * *Indexing dates and times* 52
- 2.7 Field truncation 53
- 2.8 Near-real-time search 54
- 2.9 Optimizing an index 54
- 2.10 Other directory implementations 56
- 2.11 Concurrency, thread safety, and locking issues 58
 - Thread and multi-JVM safety* 58 • *Accessing an index over a remote file system* 59 * *Index locking* 61
- 2.12 Debugging indexing 63
- 2.13 Advanced indexing concepts 64
 - Deleting documents with IndexReader* 65 • *Reclaiming disk space used by deleted documents* 66 • *Buffering and flushing* 66
 - Index commits* 67 * *ACID transactions and index consistency* 69 * *Merging* 70
- 2.14 Summary 72

- ## 3 Adding search to your application 74
- 3.1 Implementing a simple search feature 76
 - Searching for a specific term 76 * Parsing a user-entered query expression: QueryParser 77*
 - 3.2 Using IndexSearcher 80
 - Creating an IndexSearcher 81 * Performing searches .82 Working with TopDocs 82 * Paging through results 84 Near-real-time search 84*
 - 3.3 Understanding Lucene scoring 86
 - How Lucene scores 86 * Using explain() to understand hit scoring 88*
 - 3.4 Lucene's diverse queries 90
 - Searching by term: TermQuery 90 * Searching within a term range: TermRangeQuery 91 * Searching within a numeric range: NumericRangeQuery 92 * Searching on a string: PrefixQuery 93 * Combining queries: BooleanQuery 94 Searching by phrase: PhraseQuery 96 * Searching by wildcard: WildcardQuery 99 « Searching for similar terms: Fuzzy Query 100* Matching all documents: MatchAllDocsQuery 101*
 - 3.5 Parsing query expressions: QueryParser 101
 - Query. toString 102 * TermQuery 103 * Term range searches 103* Numeric and date range searches 104 Prefix and wildcard queries 104* Boolean operators 105 Phrase queries 105 * Fuzzy queries 106 MatchAllDocsQuery 107 * Grouping 107 * Field selection 107* Setting the boost for a subquery 108 To QueryParse or not to QueryParse'? 108*
 - 3.6 Summary 109
- ## 4 Lucene's analysis process 110
- 4.1 Using analyzers 111
 - Indexing analysis 113 • QueryParser analysis 114 Parsing vs. analysis: when an analyzer isn 't appropriate 114*
 - 4.2 What's inside an analyzer? 115
 - What's in a token? 116* TokenStream uncensored 117 Visualizing analyzers 120* TokenFilter order can be significant 125*
 - 4.3 Using the built-in analyzers 127
 - StopAnalyzer 127 * StandardAnalyzer 128 * Which core analyzer should you use ? 128*

- 4.4 Sounds-like querying 129
- 4.5 Synonyms, aliases, and words that mean the same 131
 - Creating SynonymAnalyzer* 132 * *Visualizing token positions* 137
- 4.6 Stemming analysis 138
 - StopFilter leaves holes* 138 * *Combining stemming and stop-word removal* 139
- 4.7 Field variations 140
 - Analysis of multivalued fields* 140 * *Field-specific analysis* 140
 - Searching on unanalyzed fields* 141
- 4.8 Language analysis issues 144
 - Unicode and encodings* 144 * *Analyzing non-English languages* 145 * *Character normalization* 145 * *Analyzing Asian languages* 146 * *Zaijian* 148
- 4.9 Nutch analysis 149
- 4.10 Summary 151

5 *Advanced search techniques* 152

- 5.1 Lucene's field cache 153
 - Loading field values for all documents* 154 * *Per-segment readers* 155
- 5.2 Sorting search results 155
 - Sorting search results by field value* 156 * *Sorting by relevance* 158 * *Sorting by index order* 159 * *Sorting by multiple fields* 160 * *Reversing sort order* 161 * *Sorting by multiple fields* 161 * *Selecting a sorting field type* 163 * *Using a nondefault locale for sorting* 163
- 5.3 Using MultiPhraseQuery 163
- 5.4 Querying on multiple fields at once 166
- 5.5 Span queries 168
 - Building block of spanning, SpanTermQuery* 170 * *Finding spans at the beginning of a field* 172 * *Spans near one another* 173 * *Excluding span overlap from matches* 174
 - SpanOrQuery* 175 * *SpanQuery and QueryParser* 177
- 5.6 Filtering a search 177
 - TermRangeFilter* 178 * *NumericRangeFilter* 179
 - FieldCacheRangeFilter* 179 * *Filtering by specific terms* 180
 - Using QueryWrapperFilter* 180 * *Using SpanQueryFilter* 181

CONTENTS

- Security filters 181 * Using BooleanQuery for filtering 183*
- PrefixFilter 183 * Caching filter results 184 * Wrapping a filter as a query 184 * Filtering a filter 184 * Beyond the built-in filters 185*
- 5.7 Custom scoring using function queries 185
 - Function query classes 185 * Boosting recently modified documents using function queries 187*
- 5.8 Searching across multiple Lucene indexes 189
 - Using MultiSearcher 189* Multithreaded searching using ParallelMultiSearcher 191*
- 5.9 Leveraging term vectors 191
 - Books like this 192* What category? 195*
 - Term VectorMapper 198*
- 5.10 Loading fields with FieldSelector 200
- 5.11 Stopping a slow search 201
- 5.12 Summary 202
- Extending search 204*
 - 6.1 Using a custom sort method 205
 - Indexing documents for geographic sorting 205 * Implementing custom geographic sort 206 * Accessing values used in custom sorting 209*
 - 6.2 Developing a custom Collector 210
 - The Collector base class 211 * Custom collector: BookLinkCollector 212 • AliDocCollector 213*
 - 6.3 Extending QueryParser 214
 - Customizing QueryParser's behavior 214 * Prohibiting fuzzy and wildcard queries 215 * Handling numeric field-range queries 216 * Handling date ranges 218 * Allowing ordered phrase queries 220*
 - 6.4 Custom filters 221
 - Implementing a custom filter 221 * Using our custom filter during searching 223 * An alternative: FilteredQuery 224*
 - 6.5 Payloads 225
 - Producing pay loads during analysis 226 * Using payloads during searching 227 * Payloads and SpanQuery 230*
 - Retrieving payloads via TermPositions 230*
 - 6.6 Summary 231

PART 2 APPLIED LUCENE 233

7 *Extracting text with Tika* 235

- 7.1 What is Tika? 236
- 7.2 Tika's logical design and API 238
- 7.3 Installing Tika 240
- 7.4 Tika's built-in text extraction tool 240
- 7.5 Extracting text programmatically 242
 - Indexing a Lucene document* 242 * *The Tika utility class* 245
 - Customizing parser selection* 246
- 7.6 Tika's limitations 246
- 7.7 Indexing custom XML 247
 - Parsing using SAX* 248 * *Parsing and indexing using Apache Commons Digester* 250
- 7.8 Alternatives 253
- 7.9 Summary 254

O *Essential Lucene extensions* 255

- ^J** 8.1 Luke, the Lucene Index Toolbox 256
 - Overview: seeing the big picture* 257 * *Document browsing* 257
 - Using QueryParser to search* 260 * *Files and plugins view* 261
- 8.2 Analyzers, tokenizers, and TokenFilters 262
 - SnowballAnalyzer* 264 * *Ngram filters* 265 * *Shingle filters* 267 • *Obtaining the contrib analyzers* 267
- 8.3 Highlighting query terms 268
 - Highlighter components* 268 * *Standalone highlighter example* 271 * *Highlighting with CSS* 272 * *Highlighting search results* 273
- 8.4 FastVectorHighlighter 275
- 8.5 Spell checking 277
 - Generating a suggestions list* 278 * *Selecting the best suggestion* 280 * *Presenting the result to the user* 281
 - Some ideas to improve spell checking* 281
- 8.6 Fun and interesting Query extensions 283
 - MoreLikeThis* 283 • *FuzzyLikeThisQuery* 284
 - BoostingQuery* 284 * *TermsFilter* 284 * *DuplicateFilter* 285
 - RegexQuery* 285

- 8.7 Building contrib modules 286
 - Get the sources* 286 * *Ant in the contrib directory* 286
- 8.8 Summary 287

9 *Further Lucene extensions* 288

- 9.1 Chaining filters 289
- 9.2 Storing an index in Berkeley DB 292
- 9.3 Synonyms from WordNet 294
 - Building the synonym index* 295 * *Tying WordNet synonyms into an analyzer* 297
- 9.4 Fast memory-based indices 298
- 9.5 XML QueryParser: Beyond "one box" search interfaces 299
 - Using XmlQueryParser* 300 * *Extending the XML query syntax* 304
- 9.6 Surround query language 306
- 9.7 Spatial Lucene 308
 - Indexing spatial data* 308 * *Searching spatial data* 312
 - Performance characteristics of Spatial Lucene* 314
- 9.8 Searching multiple indexes remotely 316
- 9.9 Flexible QueryParser 320
- 9.10 Odds and ends 322
- 9.11 Summary 323

10 *Using Lucene from other programming languages* 325

- 10.1 Ports primer 326
 - Trade-offs* 327 * *Choosing the right port* 328
- 10.2 CLucene (C++) 328
 - Motivation* 329 * *API and index compatibility* 330
 - Supported platforms* 332 * *Current and future work* 332
- 10.3 Lucene.Net (C# and other .NET languages) 332
 - API compatibility* 334 • *Index compatibility* 335
- 10.4 KinoSearch and Lucy (Perl) 335
 - KinoSearch* 336 * *Lucy* 338 * *Other Perl options* 338
- 10.5 Ferret (Ruby) 338
- 10.6 PHP 340
 - Zend Framework* 340 * *PHP Bridge* 341

CONTENTS

- 10.7 PyLucene (Python) 341
 - API compatibility 342* Other Python options 343*
- 10.8 Solr (many programming languages) 343
- 10.9 Summary 344

- Lucene administration and performance tuning 345*
 - 11.1 Performance tuning 346
 - Simple performance-tuning steps 347 * Testing approach 348*
 - Tuning for index-to-search delay 349 • Tuning for indexing throughput 350 * Tuning for search latency and throughput 354*
 - 11.2 Threads and concurrency 356
 - Using threads for indexing 357 * Using threads for searching 361*
 - 11.3 Managing resource consumption 364
 - Disk space 364 * File descriptors 367 * Memory 371*
 - 11.4 Hot backups of the index 374
 - Creating the backup 374 * Restoring the index 376*
 - 11.5 Common errors 376
 - Index corruption 377 * Repairing an index 378*
 - 11.6 Summary 378

PART 3 CASE STUDIES.....381

Case study 1: Krugle

Krugle: Searching source code 383

- 12.1 Introducing Krugle 384
- 12.2 Appliance architecture 385
- 12.3 Search performance 386
- 12.4 Parsing source code 387
- 12.5 Substring searching 388
- 12.6 Query vs. search 391
- 12.7 Future improvements 391
 - FieldCache memory usage 392 * Combining indexes 392*
- 12.8 Summary 392

13

Case study 2: SIREn*Searching semistructured documents with SIREn 394*

13.1 Introducing SIREn 395

13.2 SIREn's benefits 396

*Searching across all fields 398 * A single efficient lexicon 398
Flexible fields 398 * Efficient handling of multivalued
fields 398*

13.3 Indexing entities with SIREn 399

*Data model 399 * Implementation issues 400 * Index
schema 400 * Data preparation before indexing 401*

13.4 Searching entities with SIREn 402

*Searching content 402 * Restricting search within a cell 403
Combining cells into tuples 404 * Querying an entity
description 404*

13.5 Integrating SIREn in Solr 405

13.6 Benchmark 405

13.7 Summary 407

? /if **Case study 3: LinkedIn**#" *Adding facets and real-time search with Bobo Browse and Zoie 409*

14.1 Faceted search with Bobo Browse 410

Bobo Browse design 410 Beyond simple faceting 415*

14.2 Real-time search with Zoie 416

*Zoie architecture 418 * Real-time vs. near-real-time 421
Documents and indexing requests 421 * Custom
IndexReaders 423 * Comparison with Lucene near-real-time
search 424 * Distributed search 425*

14.3 Summary 427

appendix a Installing Lucene 428

appendix b Lucene index format 433

appendix c Lucene/contrib benchmark 443

appendix d Resources 465

index 469