Brief Contents

1 Boolean retrieval 1
2 The term vocabulary and postings lists 19
3 Dictionaries and tolerant retrieval 49
4 Index construction 67
5 Index compression 85
6 Scoring, term weighting and the vector space model 109
7 Computing scores in a complete search system 135
8 Evaluation in information retrieval 151
9 Relevance feedback and query expansion 177
10 XML retrieval 195
11 Probabilistic information retrieval 219
12 Language models for information retrieval 237
13 Text classification and Naive Bayes 253
14 Vector space classification 289
15 Support vector machines and machine learning on documents 319
16 Flat clustering 349
17 Hierarchical clustering 377
18 Matrix decompositions and latent semantic indexing 403
19 Web search basics 421
20 Web crawling and indexes 443
21 Link analysis 461
Contents

List of Tables xv
List of Figures xix
Table of Notation xxvii
Preface xxxi

1 Boolean retrieval 1
   1.1 An example information retrieval problem 3
   1.2 A first take at building an inverted index 6
   1.3 Processing Boolean queries 10
   1.4 The extended Boolean model versus ranked retrieval 14
   1.5 References and further reading 17

2 The term vocabulary and postings lists 19
   2.1 Document delineation and character sequence decoding 19
      2.1.1 Obtaining the character sequence in a document 19
      2.1.2 Choosing a document unit 20
   2.2 Determining the vocabulary of terms 22
      2.2.1 Tokenization 22
      2.2.2 Dropping common terms: stop words 27
      2.2.3 Normalization (equivalence classing of terms) 28
      2.2.4 Stemming and lemmatization 32
   2.3 Faster postings list intersection via skip pointers 36
   2.4 Positional postings and phrase queries 39
      2.4.1 Biword indexes 39
      2.4.2 Positional indexes 41
      2.4.3 Combination schemes 43
   2.5 References and further reading 45
3 Dictionaries and tolerant retrieval 49
  3.1 Search structures for dictionaries 49
  3.2 Wildcard queries 51
    3.2.1 General wildcard queries 53
    3.2.2 k-gram indexes for wildcard queries 54
  3.3 Spelling correction 56
    3.3.1 Implementing spelling correction 57
    3.3.2 Forms of spelling correction 57
    3.3.3 Edit distance 58
    3.3.4 k-gram indexes for spelling correction 60
    3.3.5 Context sensitive spelling correction 62
  3.4 Phonetic correction 63
  3.5 References and further reading 65

4 Index construction 67
  4.1 Hardware basics 68
  4.2 Blocked sort-based indexing 69
  4.3 Single-pass in-memory indexing 73
  4.4 Distributed indexing 74
  4.5 Dynamic indexing 78
  4.6 Other types of indexes 80
  4.7 References and further reading 83

5 Index compression 85
  5.1 Statistical properties of terms in information retrieval 86
    5.1.1 Heaps' law: Estimating the number of terms 88
    5.1.2 Zipf's law: Modeling the distribution of terms 89
  5.2 Dictionary compression 90
    5.2.1 Dictionary as a string 91
    5.2.2 Blocked storage 92
  5.3 Postings file compression 95
    5.3.1 Variable byte codes 96
    5.3.2 \( \gamma \) codes 98
  5.4 References and further reading 105

6 Scoring, term weighting and the vector space model 109
  6.1 Parametric and zone indexes 110
    6.1.1 Weighted zone scoring 112
    6.1.2 Learning weights 113
    6.1.3 The optimal weight \( g \) 115
  6.2 Term frequency and weighting 117
    6.2.1 Inverse document frequency 117
    6.2.2 Tf-idf weighting 118
## Contents

6.3 The vector space model for scoring 120
  6.3.1 Dot products 120
  6.3.2 Queries as vectors 123
  6.3.3 Computing vector scores 124

6.4 Variant tf-idf functions 126
  6.4.1 Sublinear tf scaling 126
  6.4.2 Maximum tf normalization 127
  6.4.3 Document and query weighting schemes 128
  6.4.4 Pivoted normalized document length 129

6.5 References and further reading 133

7 Computing scores in a complete search system 135
  7.1 Efficient scoring and ranking 135
    7.1.1 Inexact top K document retrieval 137
    7.1.2 Index elimination 137
    7.1.3 Champion lists 138
    7.1.4 Static quality scores and ordering 138
    7.1.5 Impact ordering 140
    7.1.6 Cluster pruning 141
  7.2 Components of an information retrieval system 143
    7.2.1 Tiered indexes 143
    7.2.2 Query-term proximity 144
    7.2.3 Designing parsing and scoring functions 145
    7.2.4 Putting it all together 146
  7.3 Vector space scoring and query operator interaction 147
  7.4 References and further reading 149

8 Evaluation in information retrieval 151
  8.1 Information retrieval system evaluation 152
  8.2 Standard test collections 153
  8.3 Evaluation of unranked retrieval sets 154
  8.4 Evaluation of ranked retrieval results 158
  8.5 Assessing relevance 164
    8.5.1 Critiques and justifications of the concept of relevance 166
  8.6 A broader perspective: System quality and user utility 168
    8.6.1 System issues 168
    8.6.2 User utility 169
    8.6.3 Refining a deployed system 170
  8.7 Results snippets 170
  8.8 References and further reading 173

9 Relevance feedback and query expansion 177
9.1 Relevance feedback and pseudo relevance feedback 178
  9.1.1 The Rocchio algorithm for relevance feedback 178
  9.1.2 Probabilistic relevance feedback 183
  9.1.3 When does relevance feedback work? 183
  9.1.4 Relevance feedback on the web 185
  9.1.5 Evaluation of relevance feedback strategies 186
  9.1.6 Pseudo relevance feedback 187
  9.1.7 Indirect relevance feedback 187
  9.1.8 Summary 188
9.2 Global methods for query reformulation 189
  9.2.1 Vocabulary tools for query reformulation 189
  9.2.2 Query expansion 189
  9.2.3 Automatic thesaurus generation 192
9.3 References and further reading 193

10 XML retrieval 195
  10.1 Basic XML concepts 197
  10.2 Challenges in XML retrieval 201
  10.3 A vector space model for XML retrieval 206
  10.4 Evaluation of XML retrieval 210
  10.5 Text-centric vs. data-centric XML retrieval 214
  10.6 References and further reading 216
  10.7 Exercises 217

11 Probabilistic information retrieval 219
  11.1 Review of basic probability theory 220
  11.2 The Probability Ranking Principle 221
    11.2.1 The 1/0 loss case 221
    11.2.2 The PRP with retrieval costs 222
  11.3 The Binary Independence Model 222
    11.3.1 Deriving a ranking function for query terms 224
    11.3.2 Probability estimates in theory 226
    11.3.3 Probability estimates in practice 227
    11.3.4 Probabilistic approaches to relevance feedback 228
  11.4 An appraisal and some extensions 230
    11.4.1 An appraisal of probabilistic models 230
    11.4.2 Tree-structured dependencies between terms 231
    11.4.3 Okapi BM25: a non-binary model 232
    11.4.4 Bayesian network approaches to IR 234
  11.5 References and further reading 235

12 Language models for information retrieval 237
  12.1 Language models 237
19 Web search basics

19.1 Background and history 421
19.2 Web characteristics 423
  19.2.1 The web graph 425
  19.2.2 Spam 427
19.3 Advertising as the economic model 429
19.4 The search user experience 432
  19.4.1 User query needs 432
19.5 Index size and estimation 433
19.6 Near-duplicates and shingling 437
19.7 References and further reading 441

20 Web crawling and indexes

20.1 Overview 443
  20.1.1 Features a crawler must provide 443
  20.1.2 Features a crawler should provide 444
20.2 Crawling 444
  20.2.1 Crawler architecture 445
  20.2.2 DNS resolution 449
  20.2.3 The URL frontier 451
20.3 Distributing indexes 454
20.4 Connectivity servers 455
20.5 References and further reading 458

21 Link analysis

21.1 The Web as a graph 462
  21.1.1 Anchor text and the web graph 462
21.2 PageRank 464
  21.2.1 Markov chains 465
  21.2.2 The PageRank computation 468
  21.2.3 Topic-specific PageRank 471
21.3 Hubs and Authorities 474
  21.3.1 Choosing the subset of the Web 477
21.4 References and further reading 480

Bibliography 483
Author Index 521
Index 537
List of Tables

4.1 Typical system parameters in 2007. The seek time is the time needed to position the disk head in a new position. The transfer time per byte is the rate of transfer from disk to memory when the head is in the right position.

4.2 Collection statistics for Reuters-RCV1. Values are rounded for the computations in this book. The unrounded values are: 806,791 documents, 222 tokens per document, 391,523 (distinct) terms, 6.04 bytes per token with spaces and punctuation, 4.5 bytes per token without spaces and punctuation, 7.5 bytes per term, and 96,969,056 tokens. The numbers in this table correspond to the third line ("case folding") in Table 5.1 (page 87).

4.3 The five steps in constructing an index for Reuters-RCV1 in blocked sort-based indexing. Line numbers refer to Figure 4.2.

4.4 Collection statistics for a large collection.

5.1 The effect of preprocessing on the number of terms, nonpositional postings, and tokens for Reuters-RCV1. "Δ%" indicates the reduction in size from the previous line, except that "30 stop words" and "150 stop words" both use "case folding" as their reference line. "T%" is the cumulative ("total") reduction from unfiltered. We performed stemming with the Porter stemmer (Chapter 2, page 33).

5.2 Dictionary compression for Reuters-RCV1.

5.3 Encoding gaps instead of document IDs. For example, we store gaps 107, 5, 43, …, instead of docIDs 283154, 283159, 283202, … for computer. The first docID is left unchanged (only shown for arachnocentric).

5.4 VB encoding.
5.5 Some examples of unary and $\gamma$ codes. Unary codes are only shown for the smaller numbers. Commas in $\gamma$ codes are for readability only and are not part of the actual codes.

5.6 Index and dictionary compression for Reuters-RCV1. The compression ratio depends on the proportion of actual text in the collection. Reuters-RCV1 contains a large amount of XML markup. Using the two best compression schemes, $\gamma$ encoding and blocking with front coding, the ratio compressed index to collection size is therefore especially small for Reuters-RCV1: $(101 + 5.9)/3600 \approx 0.03$.

5.7 Two gap sequences to be merged in blocked sort-based indexing

6.1 Cosine computation for Exercise 6.19.

8.1 Calculation of 11-point Interpolated Average Precision.

8.2 Calculating the kappa statistic.

10.1 RDB (relational database) search, unstructured information retrieval and structured information retrieval.

10.2 INEX 2002 collection statistics.

10.3 INEX 2002 results of the vector space model in Section 10.3 for content-and-structure (CAS) queries and the quantization function $Q$.


13.1 Data for parameter estimation examples.

13.2 Training and test times for NB.

13.3 Multinomial versus Bernoulli model.

13.4 Correct estimation implies accurate prediction, but accurate prediction does not imply correct estimation.

13.5 A set of documents for which the NB independence assumptions are problematic.

13.6 Critical values of the $\chi^2$ distribution with one degree of freedom. For example, if the two events are independent, then $P(X^2 > 6.63) < 0.01$. So for $X^2 > 6.63$ the assumption of independence can be rejected with 99% confidence.

13.7 The ten largest classes in the Reuters-21578 collection with number of documents in training and test sets.
13.8 Macro- and microaveraging. “Truth” is the true class and “call” the decision of the classifier. In this example, macroaveraged precision is
$\frac{10}{(10 + 10)} + \frac{90}{(10 + 90)} \div 2 = (0.5 + 0.9) \div 2 = 0.7$.
Microaveraged precision is $\frac{100}{(100 + 20)} \approx 0.83$. 282
13.10 Data for parameter estimation exercise. 284
14.1 Vectors and class centroids for the data in Table 13.1. 294
14.2 Training and test times for Rocchio classification. 296
14.3 Training and test times for kNN classification. 299
14.4 A linear classifier. 303
14.5 A confusion matrix for Reuters-21578. 308
15.1 Training and testing complexity of various classifiers including SVMs. 329
15.2 SVM classifier break-even $F_1$ from (Joachims 2002a, p. 114). 334
15.3 Training examples for machine-learned scoring. 342
16.1 Some applications of clustering in information retrieval. 351
16.2 The four external evaluation measures applied to the clustering in Figure 16.4. 357
16.3 The EM clustering algorithm. 371
17.1 Comparison of HAC algorithms. 395
17.2 Automatically computed cluster labels. 397
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A term-document incidence matrix.</td>
<td>4</td>
</tr>
<tr>
<td>1.2</td>
<td>Results from Shakespeare for the query Brutus AND Caesar AND NOT Calpurnia.</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td>The two parts of an inverted index.</td>
<td>7</td>
</tr>
<tr>
<td>1.4</td>
<td>Building an index by sorting and grouping.</td>
<td>8</td>
</tr>
<tr>
<td>1.5</td>
<td>Intersecting the postings lists for Brutus and Calpurnia from Figure 1.3.</td>
<td>10</td>
</tr>
<tr>
<td>1.6</td>
<td>Algorithm for the intersection of two postings lists $p_1$ and $p_2$.</td>
<td>11</td>
</tr>
<tr>
<td>1.7</td>
<td>Algorithm for conjunctive queries that returns the set of documents containing each term in the input list of terms.</td>
<td>12</td>
</tr>
<tr>
<td>2.1</td>
<td>An example of a vocalized Modern Standard Arabic word.</td>
<td>21</td>
</tr>
<tr>
<td>2.2</td>
<td>The conceptual linear order of characters is not necessarily the order that you see on the page.</td>
<td>21</td>
</tr>
<tr>
<td>2.3</td>
<td>The standard unsegmented form of Chinese text using the simplified characters of mainland China.</td>
<td>26</td>
</tr>
<tr>
<td>2.4</td>
<td>Ambiguities in Chinese word segmentation.</td>
<td>26</td>
</tr>
<tr>
<td>2.5</td>
<td>A stop list of 25 semantically non-selective words which are common in Reuters-RCV1.</td>
<td>26</td>
</tr>
<tr>
<td>2.6</td>
<td>An example of how asymmetric expansion of query terms can usefully model users’ expectations.</td>
<td>28</td>
</tr>
<tr>
<td>2.7</td>
<td>Japanese makes use of multiple intermingled writing systems and, like Chinese, does not segment words.</td>
<td>31</td>
</tr>
<tr>
<td>2.8</td>
<td>A comparison of three stemming algorithms on a sample text.</td>
<td>34</td>
</tr>
<tr>
<td>2.9</td>
<td>Postings lists with skip pointers.</td>
<td>36</td>
</tr>
<tr>
<td>2.10</td>
<td>Postings lists intersection with skip pointers.</td>
<td>37</td>
</tr>
<tr>
<td>2.11</td>
<td>Positional index example.</td>
<td>41</td>
</tr>
<tr>
<td>2.12</td>
<td>An algorithm for proximity intersection of postings lists $p_1$ and $p_2$.</td>
<td>42</td>
</tr>
</tbody>
</table>
List of Figures

3.1 A binary search tree. ........................................... 51
3.2 A B-tree. ..................................................... 52
3.3 A portion of a permuterm index. .............................. 54
3.4 Example of a postings list in a 3-gram index. ............... 55
3.5 Dynamic programming algorithm for computing the edit distance between strings \( s_1 \) and \( s_2 \). .......................... 59
3.6 Example Levenshtein distance computation. ................ 59
3.7 Matching at least two of the three 2-grams in the query bord. 61
4.1 Document from the Reuters newswire. ....................... 70
4.2 Blocked sort-based indexing. ................................ 71
4.3 Merging in blocked sort-based indexing. .................... 72
4.4 Inversion of a block in single-pass in-memory indexing ...... 73
4.5 An example of distributed indexing with MapReduce. .... 76
4.6 Map and reduce functions in MapReduce. .................. 77
4.7 Logarithmic merging. Each token (termID,docID) is initially added to in-memory index \( Z_0 \) by LMERGEOADDTOKEN. LOGARITHMICMERGE initializes \( Z_0 \) and indexes. .......... 79
4.8 A user-document matrix for access control lists. Element \((i, j)\) is 1 if user \( i \) has access to document \( j \) and 0 otherwise. During query processing, a user’s access postings list is intersected with the results list returned by the text part of the index. 81
5.1 Heaps’ law. ................................................... 88
5.2 Zipf’s law for Reuters-RCV1. .............................. 90
5.3 Storing the dictionary as an array of fixed-width entries. 91
5.4 Dictionary-as-a-string storage. ............................. 92
5.5 Blocked storage with four terms per block. ................. 93
5.6 Search of the uncompressed dictionary \( a \) and a dictionary compressed by blocking with \( k = 4 \) \((b)\). .................. 94
5.7 Front coding. ................................................. 94
5.8 VB encoding and decoding. ................................ 97
5.9 Entropy \( H(P) \) as a function of \( P(x_1) \) for a sample space with two outcomes \( x_1 \) and \( x_2 \). .......................... 100
5.10 Stratification of terms for estimating the size of a \( \gamma \) encoded inverted index. .......................... 102
6.1 Parametric search. .......................................... 111
6.2 Basic zone index ............................................. 111
6.3 Zone index in which the zone is encoded in the postings rather than the dictionary. 111
List of Figures

6.4 Algorithm for computing the weighted zone score from two postings lists. 113
6.5 An illustration of training examples. 115
6.6 The four possible combinations of $s_T$ and $s_B$. 115
6.7 Collection frequency (cf) and document frequency (df) behave differently, as in this example from the Reuters collection. 118
6.8 Example of idf values. 119
6.9 Table of tf values for Exercise 6.10. 120
6.10 Cosine similarity illustrated. 121
6.11 Euclidean normalized tf values for documents in Figure 6.9. 122
6.12 Term frequencies in three novels. 122
6.13 Term vectors for the three novels of Figure 6.12. 123
6.14 The basic algorithm for computing vector space scores. 125
6.15 SMART notation for tf-idf variants. 128
6.16 Pivoted document length normalization. 130
6.17 Implementing pivoted document length normalization by linear scaling. 131

7.1 A faster algorithm for vector space scores. 136
7.2 A static quality-ordered index. 139
7.3 Cluster pruning. 142
7.4 Tiered indexes. 144
7.5 A complete search system. 147

8.1 Graph comparing the harmonic mean to other means. 157
8.2 Precision/recall graph. 158
8.3 Averaged 11-point precision/recall graph across 50 queries for a representative TREC system. 160
8.4 The ROC curve corresponding to the precision-recall curve in Figure 8.2. 162
8.5 An example of selecting text for a dynamic snippet. 172

9.1 Relevance feedback searching over images. 179
9.2 Example of relevance feedback on a text collection. 180
9.3 The Rocchio optimal query for separating relevant and nonrelevant documents. 181
9.4 An application of Rocchio’s algorithm. 182
9.5 Results showing pseudo relevance feedback greatly improving performance. 187
9.6 An example of query expansion in the interface of the Yahoo! web search engine in 2006. 190
9.7 Examples of query expansion via the PubMed thesaurus. 191
9.8 An example of an automatically generated thesaurus. 192
List of Figures

10.1 An XML document. 198
10.2 The XML document in Figure 10.1 as a simplified DOM object. 198
10.3 An XML query in NEXI format and its partial representation as a tree. 199
10.4 Tree representation of XML documents and queries. 200
10.5 Partitioning an XML document into non-overlapping indexing units. 202
10.6 Schema heterogeneity: intervening nodes and mismatched names. 204
10.7 A structural mismatch between two queries and a document. 206
10.8 A mapping of an XML document (left) to a set of lexicalized subtrees (right). 207
10.9 The algorithm for scoring documents with SimNoMerge. 209
10.10 Scoring of a query with one structural term in SimNoMerge. 209
10.11 Simplified schema of the documents in the INEX collection. 211
11.1 A tree of dependencies between terms. 232
12.1 A simple finite automaton and some of the strings in the language it generates. 238
12.2 A one-state finite automaton that acts as a unigram language model. 238
12.3 Partial specification of two unigram language models. 239
12.4 Results of a comparison of tf-idf with language modeling (LM) term weighting by Ponte and Croft (1998). 247
12.5 Three ways of developing the language modeling approach: (a) query likelihood, (b) document likelihood, and (c) model comparison. 250
13.1 Classes, training set, and test set in text classification. 257
13.2 Naive Bayes algorithm (multinomial model): Training and testing. 260
13.3 NB algorithm (Bernoulli model): Training and testing. 263
13.4 The multinomial NB model. 266
13.5 The Bernoulli NB model. 267
13.6 Basic feature selection algorithm for selecting the k best features. 271
13.7 Features with high mutual information scores for six Reuters-RCV1 classes. 274
13.8 Effect of feature set size on accuracy for multinomial and Bernoulli models. 275
13.9 A sample document from the Reuters-21578 collection. 281
14.1 Vector space classification into three classes. 290
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.2</td>
<td>Projections of small areas of the unit sphere preserve distances</td>
<td>291</td>
</tr>
<tr>
<td>14.3</td>
<td>Rocchio classification</td>
<td>293</td>
</tr>
<tr>
<td>14.4</td>
<td>Rocchio classification: Training and testing</td>
<td>295</td>
</tr>
<tr>
<td>14.5</td>
<td>The multimodal class “a” consists of two different clusters (small upper circles centered on X’s)</td>
<td>295</td>
</tr>
<tr>
<td>14.6</td>
<td>Voronoi tessellation and decision boundaries (double lines) in 1NN classification</td>
<td>297</td>
</tr>
<tr>
<td>14.7</td>
<td>kNN training (with preprocessing) and testing</td>
<td>298</td>
</tr>
<tr>
<td>14.8</td>
<td>There are an infinite number of hyperplanes that separate two linearly separable classes</td>
<td>301</td>
</tr>
<tr>
<td>14.9</td>
<td>Linear classification algorithm</td>
<td>302</td>
</tr>
<tr>
<td>14.10</td>
<td>A linear problem with noise</td>
<td>304</td>
</tr>
<tr>
<td>14.11</td>
<td>A nonlinear problem</td>
<td>305</td>
</tr>
<tr>
<td>14.12</td>
<td>$J$ hyperplanes do not divide space into $J$ disjoint regions</td>
<td>307</td>
</tr>
<tr>
<td>14.13</td>
<td>Arithmetic transformations for the bias-variance decomposition</td>
<td>310</td>
</tr>
<tr>
<td>14.14</td>
<td>Example for differences between Euclidean distance, dot product similarity and cosine similarity</td>
<td>316</td>
</tr>
<tr>
<td>14.15</td>
<td>A simple non-separable set of points</td>
<td>317</td>
</tr>
<tr>
<td>15.1</td>
<td>The support vectors are the 5 points right up against the margin of the classifier</td>
<td>320</td>
</tr>
<tr>
<td>15.2</td>
<td>An intuition for large-margin classification</td>
<td>321</td>
</tr>
<tr>
<td>15.3</td>
<td>The geometric margin of a point ($r$) and a decision boundary ($\rho$)</td>
<td>323</td>
</tr>
<tr>
<td>15.4</td>
<td>A tiny 3 data point training set for an SVM</td>
<td>325</td>
</tr>
<tr>
<td>15.5</td>
<td>Large margin classification with slack variables</td>
<td>327</td>
</tr>
<tr>
<td>15.6</td>
<td>Projecting data that is not linearly separable into a higher dimensional space can make it linearly separable</td>
<td>331</td>
</tr>
<tr>
<td>15.7</td>
<td>A collection of training examples</td>
<td>343</td>
</tr>
<tr>
<td>16.1</td>
<td>An example of a data set with a clear cluster structure</td>
<td>349</td>
</tr>
<tr>
<td>16.2</td>
<td>Clustering of search results to improve recall</td>
<td>352</td>
</tr>
<tr>
<td>16.3</td>
<td>An example of a user session in Scatter-Gather</td>
<td>353</td>
</tr>
<tr>
<td>16.4</td>
<td>Purity as an external evaluation criterion for cluster quality</td>
<td>357</td>
</tr>
<tr>
<td>16.5</td>
<td>The K-means algorithm</td>
<td>361</td>
</tr>
<tr>
<td>16.6</td>
<td>A K-means example for $K = 2$ in $\mathbb{R}^2$</td>
<td>362</td>
</tr>
<tr>
<td>16.7</td>
<td>The outcome of clustering in K-means depends on the initial seeds</td>
<td>364</td>
</tr>
<tr>
<td>16.8</td>
<td>Estimated minimal residual sum of squares as a function of the number of clusters in K-means</td>
<td>366</td>
</tr>
<tr>
<td>17.1</td>
<td>A dendrogram of a single-link clustering of 30 documents from Reuters-RCV1</td>
<td>379</td>
</tr>
<tr>
<td>17.2</td>
<td>A simple, but inefficient HAC algorithm</td>
<td>381</td>
</tr>
</tbody>
</table>
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.3</td>
<td>The different notions of cluster similarity used by the four HAC algorithms.</td>
<td>381</td>
</tr>
<tr>
<td>17.4</td>
<td>A single-link (left) and complete-link (right) clustering of eight documents.</td>
<td>382</td>
</tr>
<tr>
<td>17.5</td>
<td>A dendrogram of a complete-link clustering.</td>
<td>383</td>
</tr>
<tr>
<td>17.6</td>
<td>Chaining in single-link clustering.</td>
<td>384</td>
</tr>
<tr>
<td>17.7</td>
<td>Outliers in complete-link clustering.</td>
<td>385</td>
</tr>
<tr>
<td>17.8</td>
<td>The priority-queue algorithm for HAC.</td>
<td>386</td>
</tr>
<tr>
<td>17.9</td>
<td>Single-link clustering algorithm using an NBM array.</td>
<td>387</td>
</tr>
<tr>
<td>17.10</td>
<td>Complete-link clustering is not best-merge persistent.</td>
<td>388</td>
</tr>
<tr>
<td>17.11</td>
<td>Three iterations of centroid clustering.</td>
<td>391</td>
</tr>
<tr>
<td>17.12</td>
<td>Centroid clustering is not monotonic.</td>
<td>392</td>
</tr>
<tr>
<td>18.1</td>
<td>Illustration of the singular-value decomposition.</td>
<td>409</td>
</tr>
<tr>
<td>18.2</td>
<td>Illustration of low rank approximation using the singular-value decomposition.</td>
<td>411</td>
</tr>
<tr>
<td>18.3</td>
<td>The documents of Example 18.4 reduced to two dimensions in $(V')$.</td>
<td>416</td>
</tr>
<tr>
<td>18.4</td>
<td>Documents for Exercise 18.11.</td>
<td>418</td>
</tr>
<tr>
<td>18.5</td>
<td>Glossary for Exercise 18.11.</td>
<td>418</td>
</tr>
<tr>
<td>19.1</td>
<td>A dynamically generated web page.</td>
<td>425</td>
</tr>
<tr>
<td>19.2</td>
<td>Two nodes of the web graph joined by a link.</td>
<td>425</td>
</tr>
<tr>
<td>19.3</td>
<td>A sample small web graph.</td>
<td>426</td>
</tr>
<tr>
<td>19.4</td>
<td>The bowtie structure of the Web.</td>
<td>427</td>
</tr>
<tr>
<td>19.5</td>
<td>Cloaking as used by spammers.</td>
<td>428</td>
</tr>
<tr>
<td>19.6</td>
<td>Search advertising triggered by query keywords.</td>
<td>431</td>
</tr>
<tr>
<td>19.7</td>
<td>The various components of a web search engine.</td>
<td>434</td>
</tr>
<tr>
<td>19.8</td>
<td>Illustration of shingle sketches.</td>
<td>439</td>
</tr>
<tr>
<td>19.9</td>
<td>Two sets $S_{j_1}$ and $S_{j_2}$; their Jaccard coefficient is 2/5.</td>
<td>440</td>
</tr>
<tr>
<td>20.1</td>
<td>The basic crawler architecture.</td>
<td>446</td>
</tr>
<tr>
<td>20.2</td>
<td>Distributing the basic crawl architecture.</td>
<td>449</td>
</tr>
<tr>
<td>20.3</td>
<td>The URL frontier.</td>
<td>452</td>
</tr>
<tr>
<td>20.4</td>
<td>Example of an auxiliary hosts-to-back queues table.</td>
<td>453</td>
</tr>
<tr>
<td>20.5</td>
<td>A lexicographically ordered set of URLs.</td>
<td>456</td>
</tr>
<tr>
<td>20.6</td>
<td>A four-row segment of the table of links.</td>
<td>457</td>
</tr>
<tr>
<td>21.1</td>
<td>The random surfer at node A proceeds with probability 1/3 to each of B, C and D.</td>
<td>464</td>
</tr>
<tr>
<td>21.2</td>
<td>A simple Markov chain with three states; the numbers on the links indicate the transition probabilities.</td>
<td>466</td>
</tr>
<tr>
<td>21.3</td>
<td>The sequence of probability vectors.</td>
<td>469</td>
</tr>
</tbody>
</table>
List of Figures

21.4 A small web graph. 470
21.5 Topic-specific PageRank. 472
21.6 A sample run of HITS on the query japan elementary schools. 479
21.7 Web graph for Exercise 21.22. 480
### Table of Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma$</td>
<td>p. 98</td>
<td>$\gamma$ code</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>p. 256</td>
<td>Classification or clustering function: $\gamma(d)$ is $d$’s class or cluster</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>p. 256</td>
<td>Supervised learning method in Chapters 13 and 14: $\Gamma(D)$ is the classification function $\gamma$ learned from training set $D$</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>p. 404</td>
<td>Eigenvalue</td>
</tr>
<tr>
<td>$\bar{\mu}(\cdot)$</td>
<td>p. 292</td>
<td>Centroid of a class (in Rocchio classification) or a cluster (in $K$-means and centroid clustering)</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>p. 114</td>
<td>Training example</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>p. 408</td>
<td>Singular value</td>
</tr>
<tr>
<td>$\Theta(\cdot)$</td>
<td>p. 11</td>
<td>A tight bound on the complexity of an algorithm</td>
</tr>
<tr>
<td>$\omega, \omega_k$</td>
<td>p. 357</td>
<td>Cluster in clustering</td>
</tr>
<tr>
<td>$\Omega$</td>
<td>p. 357</td>
<td>Clustering or set of clusters ${\omega_1, \ldots, \omega_K}$</td>
</tr>
<tr>
<td>$\arg\max_x f(x)$</td>
<td>p. 181</td>
<td>The value of $x$ for which $f$ reaches its maximum</td>
</tr>
<tr>
<td>$\arg\min_x f(x)$</td>
<td>p. 181</td>
<td>The value of $x$ for which $f$ reaches its minimum</td>
</tr>
<tr>
<td>$c, c_j$</td>
<td>p. 256</td>
<td>Class or category in classification</td>
</tr>
<tr>
<td>$cf_t$</td>
<td>p. 89</td>
<td>The collection frequency of term $t$ (the total number of times the term appears in the document collection)</td>
</tr>
<tr>
<td>$C$</td>
<td>p. 256</td>
<td>Set ${c_1, \ldots, c_J}$ of all classes</td>
</tr>
<tr>
<td>$\mathbb{C}$</td>
<td>p. 268</td>
<td>A random variable that takes as values members of $C$</td>
</tr>
</tbody>
</table>
Table of Notation

- C: Term-document matrix
- d: Index of the $d^{th}$ document in the collection $D$
- $d$: A document
- $\bar{d}$, $\bar{q}$: Document vector, query vector
- $D$: Set $\{d_1, \ldots, d_N\}$ of all documents
- $D_c$: Set of documents that is in class $c$
- $\mathcal{D}$: Set $\{\langle d_1, c_1 \rangle, \ldots, \langle d_N, c_N \rangle\}$ of all labeled documents in Chapters 13–15
- $df_t$: The document frequency of term $t$ (the total number of documents in the collection the term appears in)
- $H$: Entropy
- $H_M$: $M^{th}$ harmonic number
- $I(X;Y)$: Mutual information of random variables $X$ and $Y$
- $idf_t$: Inverse document frequency of term $t$
- $J$: Number of classes
- $k$: Top $k$ items from a set, e.g., $k$ nearest neighbors in kNN, top $k$ retrieved documents, top $k$ selected features from the vocabulary $V$
- $k$: Sequence of $k$ characters
- $K$: Number of clusters
- $L_d$: Length of document $d$ (in tokens)
- $L_a$: Length of the test document (or application document) in tokens
- $L_{ave}$: Average length of a document (in tokens)
- $M$: Size of the vocabulary ($|V|$)
- $M_a$: Size of the vocabulary of the test document (or application document)
- $M_{ave}$: Average size of the vocabulary in a document in the collection
- $M_d$: Language model for document $d$
- $N$: Number of documents in the retrieval or training collection
- $N_c$: Number of documents in class $c$
- $N(\omega)$: Number of times the event $\omega$ occurred
<table>
<thead>
<tr>
<th>Notation</th>
<th>Page</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(\cdot)$</td>
<td>p. 11</td>
<td>A bound on the complexity of an algorithm</td>
</tr>
<tr>
<td>$O(\cdot)$</td>
<td>p. 221</td>
<td>The odds of an event</td>
</tr>
<tr>
<td>$P$</td>
<td>p. 155</td>
<td>Precision</td>
</tr>
<tr>
<td>$P(\cdot)$</td>
<td>p. 220</td>
<td>Probability</td>
</tr>
<tr>
<td>$P$</td>
<td>p. 465</td>
<td>Transition probability matrix</td>
</tr>
<tr>
<td>$q$</td>
<td>p. 59</td>
<td>A query</td>
</tr>
<tr>
<td>$R$</td>
<td>p. 155</td>
<td>Recall</td>
</tr>
<tr>
<td>$s_i$</td>
<td>p. 58</td>
<td>A string</td>
</tr>
<tr>
<td>$s_i$</td>
<td>p. 112</td>
<td>Boolean values for zone scoring</td>
</tr>
<tr>
<td>$\sim(d_1, d_2)$</td>
<td>p. 121</td>
<td>Similarity score for documents $d_1, d_2$</td>
</tr>
<tr>
<td>$T$</td>
<td>p. 43</td>
<td>Total number of tokens in the document collection</td>
</tr>
<tr>
<td>$T_{ct}$</td>
<td>p. 259</td>
<td>Number of occurrences of word $t$ in documents of class $c$</td>
</tr>
<tr>
<td>$t$</td>
<td>p. 4</td>
<td>Index of the $t^{th}$ term in the vocabulary $V$</td>
</tr>
<tr>
<td>$t$</td>
<td>p. 61</td>
<td>A term in the vocabulary</td>
</tr>
<tr>
<td>$tf_{t,d}$</td>
<td>p. 117</td>
<td>The term frequency of term $t$ in document $d$ (the total number of occurrences of $t$ in $d$)</td>
</tr>
<tr>
<td>$U_t$</td>
<td>p. 266</td>
<td>Random variable taking values 0 (term $t$ is present) and 1 ($t$ is not present)</td>
</tr>
<tr>
<td>$V$</td>
<td>p. 208</td>
<td>Vocabulary of terms ${t_1, \ldots, t_M}$ in a collection (a.k.a. the lexicon)</td>
</tr>
<tr>
<td>$\vec{v}(d)$</td>
<td>p. 122</td>
<td>Length-normalized document vector</td>
</tr>
<tr>
<td>$\vec{V}(d)$</td>
<td>p. 120</td>
<td>Vector of document $d$, not length-normalized</td>
</tr>
<tr>
<td>$wt_{t,d}$</td>
<td>p. 125</td>
<td>Weight of term $t$ in document $d$</td>
</tr>
<tr>
<td>$w$</td>
<td>p. 112</td>
<td>A weight, for example for zones or terms</td>
</tr>
<tr>
<td>$\vec{o}^T \vec{x} = b$</td>
<td>p. 293</td>
<td>Hyperplane; $\vec{o}$ is the normal vector of the hyperplane and $w_i$ component $i$ of $\vec{o}$</td>
</tr>
<tr>
<td>$\vec{x}$</td>
<td>p. 222</td>
<td>Term incidence vector $\vec{x} = (x_1, \ldots, x_M)$; more generally: document feature representation</td>
</tr>
<tr>
<td>$X$</td>
<td>p. 266</td>
<td>Random variable taking values in $V$, the vocabulary (e.g., at a given position $k$ in a document)</td>
</tr>
<tr>
<td>$|X|$</td>
<td>p. 256</td>
<td>Document space in text classification</td>
</tr>
<tr>
<td>$</td>
<td>A</td>
<td>$</td>
</tr>
<tr>
<td>$</td>
<td>S</td>
<td>$</td>
</tr>
<tr>
<td>Notation</td>
<td>Page</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>$</td>
<td>s_i</td>
<td>$</td>
</tr>
<tr>
<td>$</td>
<td>\vec{x}</td>
<td>$</td>
</tr>
<tr>
<td>$</td>
<td>\vec{x} - \vec{y}</td>
<td>$</td>
</tr>
</tbody>
</table>
Preface

As recently as the 1990s, studies showed that most people preferred getting information from other people rather than from information retrieval systems. Of course, in that time period, most people also used human travel agents to book their travel. However, during the last decade, relentless optimization of information retrieval effectiveness has driven web search engines to new quality levels where most people are satisfied most of the time, and web search has become a standard and often preferred source of information finding. For example, the 2004 Pew Internet Survey (Fallows 2004) found that “92% of Internet users say the Internet is a good place to go for getting everyday information.” To the surprise of many, the field of information retrieval has moved from being a primarily academic discipline to being the basis underlying most people’s preferred means of information access. This book presents the scientific underpinnings of this field, at a level accessible to graduate students as well as advanced undergraduates.

Information retrieval did not begin with the Web. In response to various challenges of providing information access, the field of information retrieval evolved to give principled approaches to searching various forms of content. The field began with scientific publications and library records, but soon spread to other forms of content, particularly those of information professionals, such as journalists, lawyers, and doctors. Much of the scientific research on information retrieval has occurred in these contexts, and much of the continued practice of information retrieval deals with providing access to unstructured information in various corporate and governmental domains, and this work forms much of the foundation of our book.

Nevertheless, in recent years, a principal driver of innovation has been the World Wide Web, unleashing publication at the scale of tens of millions of content creators. This explosion of published information would be moot if the information could not be found, annotated and analyzed so that each user can quickly find information that is both relevant and comprehensive for their needs. By the late 1990s, many people felt that continuing to index
the whole Web would rapidly become impossible, due to the Web’s exponential growth in size. But major scientific innovations, superb engineering, the rapidly declining price of computer hardware, and the rise of a commercial underpinning for web search have all conspired to power today’s major search engines, which are able to provide high-quality results within subsecond response times for hundreds of millions of searches a day over billions of web pages.

Book organization and course development

This book is the result of a series of courses we have taught at Stanford University and at the University of Stuttgart, in a range of durations including a single quarter, one semester and two quarters. These courses were aimed at early-stage graduate students in computer science, but we have also had enrollment from upper-class computer science undergraduates, as well as students from law, medical informatics, statistics, linguistics and various engineering disciplines. The key design principle for this book, therefore, was to cover what we believe to be important in a one-term graduate course on information retrieval. An additional principle is to build each chapter around material that we believe can be covered in a single lecture of 75 to 90 minutes.

The first eight chapters of the book are devoted to the basics of information retrieval, and in particular the heart of search engines; we consider this material to be core to any course on information retrieval. Chapter 1 introduces inverted indexes, and shows how simple Boolean queries can be processed using such indexes. Chapter 2 builds on this introduction by detailing the manner in which documents are preprocessed before indexing and by discussing how inverted indexes are augmented in various ways for functionality and speed. Chapter 3 discusses search structures for dictionaries and how to process queries that have spelling errors and other imprecise matches to the vocabulary in the document collection being searched. Chapter 4 describes a number of algorithms for constructing the inverted index from a text collection with particular attention to highly scalable and distributed algorithms that can be applied to very large collections. Chapter 5 covers techniques for compressing dictionaries and inverted indexes. These techniques are critical for achieving subsecond response times to user queries in large search engines. The indexes and queries considered in Chapters 1–5 only deal with Boolean retrieval, in which a document either matches a query, or does not. A desire to measure the extent to which a document matches a query, or the score of a document for a query, motivates the development of term weighting and the computation of scores in Chapters 6 and 7, leading to the idea of a list of documents that are rank-ordered for a query. Chapter 8 focuses on the evaluation of an information retrieval system based on the
relevance of the documents it retrieves, allowing us to compare the relative performances of different systems on benchmark document collections and queries.

Chapters 9–21 build on the foundation of the first eight chapters to cover a variety of more advanced topics. Chapter 9 discusses methods by which retrieval can be enhanced through the use of techniques like relevance feedback and query expansion, which aim at increasing the likelihood of retrieving relevant documents. Chapter 10 considers information retrieval from documents that are structured with markup languages like XML and HTML. We treat structured retrieval by reducing it to the vector space scoring methods developed in Chapter 6. Chapters 11 and 12 invoke probability theory to compute scores for documents on queries. Chapter 11 develops traditional probabilistic information retrieval, which provides a framework for computing the probability of relevance of a document, given a set of query terms. This probability may then be used as a score in ranking. Chapter 12 illustrates an alternative, wherein for each document in a collection, we build a language model from which one can estimate a probability that the language model generates a given query. This probability is another quantity with which we can rank-order documents.

Chapters 13–17 give a treatment of various forms of machine learning and numerical methods in information retrieval. Chapters 13–15 treat the problem of classifying documents into a set of known categories, given a set of documents along with the classes they belong to. Chapter 13 motivates statistical classification as one of the key technologies needed for a successful search engine, introduces Naive Bayes, a conceptually simple and efficient text classification method, and outlines the standard methodology for evaluating text classifiers. Chapter 14 employs the vector space model from Chapter 6 and introduces two classification methods, Rocchio and kNN, that operate on document vectors. It also presents the bias-variance tradeoff as an important characterization of learning problems that provides criteria for selecting an appropriate method for a text classification problem. Chapter 15 introduces support vector machines, which many researchers currently view as the most effective text classification method. We also develop connections in this chapter between the problem of classification and seemingly disparate topics such as the induction of scoring functions from a set of training examples.

Chapters 16–18 consider the problem of inducing clusters of related documents from a collection. In Chapter 16, we first give an overview of a number of important applications of clustering in information retrieval. We then describe two flat clustering algorithms: the K-means algorithm, an efficient and widely used document clustering method; and the Expectation-Maximization algorithm, which is computationally more expensive, but also more flexible. Chapter 17 motivates the need for hierarchically structured
clusterings (instead of flat clusterings) in many applications in information retrieval and introduces a number of clustering algorithms that produce a hierarchy of clusters. The chapter also addresses the difficult problem of automatically computing labels for clusters. Chapter 18 develops methods from linear algebra that constitute an extension of clustering, and also offer intriguing prospects for algebraic methods in information retrieval, which have been pursued in the approach of latent semantic indexing.

Chapters 19–21 treat the problem of web search. We give in Chapter 19 a summary of the basic challenges in web search, together with a set of techniques that are pervasive in web information retrieval. Next, Chapter 20 describes the architecture and requirements of a basic web crawler. Finally, Chapter 21 considers the power of link analysis in web search, using in the process several methods from linear algebra and advanced probability theory.

This book is not comprehensive in covering all topics related to information retrieval. We have put aside a number of topics, which we deemed outside the scope of what we wished to cover in an introduction to information retrieval class. Nevertheless, for people interested in these topics, we provide a few pointers to mainly textbook coverage here.

**Cross-language IR** (Grossman and Frieder 2004, ch. 4) and (Oard and Dorr 1996).


**Speech retrieval** (Coden et al. 2002).

**Music Retrieval** (Downie 2006) and [http://www.ismir.net/](http://www.ismir.net/).

**User interfaces for IR** (Baeza-Yates and Ribeiro-Neto 1999, ch. 10).

**Parallel and Peer-to-Peer IR** (Grossman and Frieder 2004, ch. 7), (Baeza-Yates and Ribeiro-Neto 1999, ch. 9), and (Aberer 2001).

**Digital libraries** (Baeza-Yates and Ribeiro-Neto 1999, ch. 15) and (Lesk 2004).

**Information science perspective** (Korfhage 1997), (Meadow et al. 1999), and (Ingwersen and Järvelin 2005).

**Logic-based approaches to IR** (van Rijsbergen 1989).

**Natural Language Processing techniques** (Manning and Schütze 1999), (Jurafsky and Martin 2008), and (Lewis and Jones 1996).
Prerequisites

Introductory courses in data structures and algorithms, in linear algebra and in probability theory suffice as prerequisites for all 21 chapters. We now give more detail for the benefit of readers and instructors who wish to tailor their reading to some of the chapters.

Chapters 1–5 assume as prerequisite a basic course in algorithms and data structures. Chapters 6 and 7 require, in addition, a knowledge of basic linear algebra including vectors and dot products. No additional prerequisites are assumed until Chapter 11, where a basic course in probability theory is required; Section 11.1 gives a quick review of the concepts necessary in Chapters 11–13. Chapter 15 assumes that the reader is familiar with the notion of nonlinear optimization, although the chapter may be read without detailed knowledge of algorithms for nonlinear optimization. Chapter 18 demands a first course in linear algebra including familiarity with the notions of matrix rank and eigenvectors; a brief review is given in Section 18.1. The knowledge of eigenvalues and eigenvectors is also necessary in Chapter 21.

Book layout

Worked examples in the text appear with a pencil sign next to them in the left margin. Advanced or difficult material appears in sections or subsections indicated with scissors in the margin. Exercises are marked in the margin with a question mark. The level of difficulty of exercises is indicated as easy (⋆), medium (⋆⋆), or difficult (⋆ ⋆ ⋆).

Acknowledgments

We would like to thank Cambridge University Press for allowing us to make the draft book available online, which facilitated much of the feedback we have received while writing the book. We also thank Lauren Cowles, who has been an outstanding editor, providing several rounds of comments on each chapter, on matters of style, organization, and coverage, as well as detailed comments on the subject matter of the book. To the extent that we have achieved our goals in writing this book, she deserves an important part of the credit.

We are very grateful to the many people who have given us comments, suggestions, and corrections based on draft versions of this book. We thank for providing various corrections and comments: Cheryl Aasheim, Josh Attenberg, Daniel Beck, Luc Bélanger, Georg Buscher, Tom Breuel, Daniel Burkhardt, Fazli Can, Dinquan Chen, Stephen Clark, Ernest Davis, Pedro Domingos, Rodrigo Panchiniak Fernandes, Paolo Ferragina, Alex Fraser, Norbert
Many people gave us detailed feedback on individual chapters, either at our request or through their own initiative. For this, we’re particularly grateful to: James Allan, Omar Alonso, Ismail Sengor Altingovde, Vo Ngoc Anh, Roi Blanco, Eric Breck, Eric Brown, Mark Carman, Carlos Castillo, Junghoo Cho, Aron Culotta, Doug Cutting, Meghana Deodhar, Susan Dumais, Johannes Fürnkranz, Andreas Heß, Djoerd Hiemstra, David Hull, Thorsten Joachims, Siddharth Jonathan J. B., Jaap Kamps, Mounia Lalmas, Amy Langville, Nicholas Lester, Dave Lewis, Daniel Lowd, Yosi Mass, Jeff Michels, Alessandro Moschitti, Amir Najmi, Marc Najork, Giorgio Maria Di Nunzio, Paul Ogilvie, Priyank Patel, Jan Pedersen, Kathryn Pedings, Vassilis Plachouras, Daniel Ramage, Ghulam Raza, Stefan Riezler, Michael Schiehlen, Helmut Schmid, Falk Nicolas Scholer, Sabine Schulte im Walde, Fabrizio Sebastiani, Sarabjeet Singh, Valentin Spitkovsky, Alexander Strehl, John Tait, Shivakumar Vaithyanathan, Ellen Voorhees, Gerhard Weikum, Dawid Weiss, Yiming Yang, Yisong Yue, Jian Zhang, and Justin Zobel.

And finally there were a few reviewers who absolutely stood out in terms of the quality and quantity of comments that they provided. We thank them for their significant impact on the content and structure of the book. We express our gratitude to Pavel Berkhin, Stefan Büttcher, Jamie Callan, Byron Dom, Torsten Suel, and Andrew Trotman.

Parts of the initial drafts of Chapters 13–15 were based on slides that were generously provided by Ray Mooney. While the material has gone through extensive revisions, we gratefully acknowledge Ray’s contribution to the three chapters in general and to the description of the time complexities of text classification algorithms in particular.

The above is unfortunately an incomplete list: we are still in the process of incorporating feedback we have received. And, like all opinionated authors, we did not always heed the advice that was so freely given. The published versions of the chapters remain solely the responsibility of the authors.

The authors thank Stanford University and the University of Stuttgart for providing a stimulating academic environment for discussing ideas and the opportunity to teach courses from which this book arose and in which its
contents were refined. CM thanks his family for the many hours they’ve let him spend working on this book, and hopes he’ll have a bit more free time on weekends next year. PR thanks his family for their patient support through the writing of this book and is also grateful to Yahoo! Inc. for providing a fertile environment in which to work on this book. HS would like to thank his parents, family, and friends for their support while writing this book.

**Web and contact information**

This book has a companion website at http://informationretrieval.org. As well as links to some more general resources, it is our intent to maintain on this website a set of slides for each chapter which may be used for the corresponding lecture. We gladly welcome further feedback, corrections, and suggestions on the book, which may be sent to all the authors at informationretrieval (at) yahoogroups (dot) com.