

classification

Christopher D. Manning

search

Prabhakar Raghavan

Hinrich Schütze

precision

crawler

links

spam

Introduction to

recall

Information Retrieval

query

svm

clustering

index

web

xml

language model

CAMBRIDGE

ranking

DivX Exhibit 2010
Page 2010 - 1
IPR2020-00558

Introduction to Information Retrieval

Christopher D. Manning
Stanford University

Prabhakar Raghavan
Yahoo! Research

Hinrich Schütze
University of Stuttgart



CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org

Information on this title: www.cambridge.org/9780521865715

© Cambridge University Press 2008

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2008

4th printing 2018

Printed and bound in Great Britain by Clays Ltd, Elcograf S.p.A.

A catalog record for this publication is available from the British Library.

Library of Congress Cataloging in Publication Data

Manning, Christopher D.

Introduction to information retrieval / Christopher D. Manning, Prabhakar

Raghavan, Hinrich Schütze.

p. cm.

Includes bibliographical references and index.

ISBN 978-0-521-86571-5 (hardback)

1. Text processing (Computer science) 2. Information retrieval. 3. Document clustering. 4. Semantic Web. I. Raghavan, Prabhakar. II. Schütze, Hinrich.

III. Title.

QA76.9.T48M26 2008

025.04 – dc22 2008001257

ISBN 978-0-521-86571-5 hardback

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party Internet Web sites referred to in this publication and does not guarantee that any content on such Web sites is, or will remain, accurate or appropriate. Information regarding prices, travel timetables, and other factual information given in this work are correct at the time of first printing, but Cambridge University Press does not guarantee the accuracy of such information thereafter.

1 *Boolean retrieval*

The meaning of the term *information retrieval* (IR) can be very broad. Just getting a credit card out of your wallet so that you can type in the card number is a form of information retrieval. However, as an academic field of study, *information retrieval* might be defined thus:

INFORMATION
RETRIEVAL

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

As defined in this way, information retrieval used to be an activity that only a few people engaged in: reference librarians, paralegals, and similar professional searchers. Now the world has changed, and hundreds of millions of people engage in information retrieval every day when they use a web search engine or search their email.¹ Information retrieval is fast becoming the dominant form of information access, overtaking traditional database-style searching (the sort that is going on when a clerk says to you: "I'm sorry, I can only look up your order if you can give me your order ID").

Information retrieval can also cover other kinds of data and information problems beyond that specified in the core definition above. The term "unstructured data" refers to data that does not have clear, semantically overt, easy-for-a-computer structure. It is the opposite of structured data, the canonical example of which is a relational database, of the sort companies usually use to maintain product inventories and personnel records. In reality, almost no data are truly "unstructured." This is definitely true of all text data if you count the latent linguistic structure of human languages. But even accepting that the intended notion of structure is overt structure, most text has structure, such as headings, paragraphs, and footnotes, which is commonly represented in documents by explicit markup (such as the coding underlying web pages). Information retrieval is also used to facilitate "semistructured"

¹ In modern parlance, the word "search" has tended to replace "(information) retrieval"; the term "search" is quite ambiguous, but in context we use the two synonymously.

search such as finding a document where the title contains Java and the body contains threading.

The field of IR also covers supporting users in browsing or filtering document collections or further processing a set of retrieved documents. Given a set of documents, clustering is the task of coming up with a good grouping of the documents based on their contents. It is similar to arranging books on a bookshelf according to their topic. Given a set of topics, standing information needs, or other categories (such as suitability of texts for different age groups), classification is the task of deciding which class(es), if any, each of a set of documents belongs to. It is often approached by first manually classifying some documents and then hoping to be able to classify new documents automatically.

Information retrieval systems can also be distinguished by the scale at which they operate, and it is useful to distinguish three prominent scales. In *web search*, the system has to provide search over billions of documents stored on millions of computers. Distinctive issues are needing to gather documents for indexing, being able to build systems that work efficiently at this enormous scale, and handling particular aspects of the web, such as the exploitation of hypertext and not being fooled by site providers manipulating page content in an attempt to boost their search engine rankings, given the commercial importance of the web. We focus on all these issues in Chapters 19–21. At the other extreme is *personal information retrieval*. In the last few years, consumer operating systems have integrated information retrieval (such as Apple's Mac OS X Spotlight or Windows Vista's Instant Search). Email programs usually not only provide search but also text classification: they at least provide a spam (junk mail) filter, and commonly also provide either manual or automatic means for classifying mail so that it can be placed directly into particular folders. Distinctive issues here include handling the broad range of document types on a typical personal computer, and making the search system maintenance free and sufficiently lightweight in terms of startup, processing, and disk space usage that it can run on one machine without annoying its owner. In between is the space of *enterprise, institutional, and domain-specific search*, where retrieval might be provided for collections such as a corporation's internal documents, a database of patents, or research articles on biochemistry. In this case, the documents are typically stored on centralized file systems and one or a handful of dedicated machines provide search over the collection. This book contains techniques of value over this whole spectrum, but our coverage of some aspects of parallel and distributed search in web-scale search systems is comparatively light owing to the relatively small published literature on the details of such systems. However, outside of a handful of web search companies, a software developer is most likely to encounter the personal search and enterprise scenarios.

In this chapter, we begin with a very simple example of an IR problem, and introduce the idea of a term-document matrix (Section 1.1) and the

central inverted index data structure (Section 1.2). We then examine the Boolean retrieval model and how Boolean queries are processed (Sections 1.3 and 1.4).

1.1 An example information retrieval problem

A fat book that many people own is *Shakespeare's Collected Works*. Suppose you wanted to determine which plays of Shakespeare contain the words Brutus AND Caesar AND NOT Calpurnia. One way to do that is to start at the beginning and to read through all the text, noting for each play whether it contains Brutus and Caesar and excluding it from consideration if it contains Calpurnia. The simplest form of document retrieval is for a computer to do this sort of linear scan through documents. This process is commonly referred to as *grepping* through text, after the Unix command `grep`, which performs this process. Grepping through text can be a very effective process, especially given the speed of modern computers, and often allows useful possibilities for wildcard pattern matching through the use of regular expressions. With modern computers, for simple querying of modest collections (the size of *Shakespeare's Collected Works* is a bit under one million words of text in total), you really need nothing more.

But for many purposes, you do need more:

1. To process large document collections quickly. The amount of online data has grown at least as quickly as the speed of computers, and we would now like to be able to search collections that total in the order of billions to trillions of words.
2. To allow more flexible matching operations. For example, it is impractical to perform the query Romans NEAR countrymen with `grep`, where NEAR might be defined as "within 5 words" or "within the same sentence."
3. To allow ranked retrieval. In many cases, you want the best answer to an information need among many documents that contain certain words.

INDEX The way to avoid linearly scanning the texts for each query is to *index* the documents in advance. Let us stick with *Shakespeare's Collected Works*, and use it to introduce the basics of the Boolean retrieval model. Suppose we record for each document – here a play of Shakespeare's – whether it contains each word out of all the words Shakespeare used (Shakespeare used about 32,000 different words). The result is a binary term-document *incidence matrix*, as in Figure 1.1. *Terms* are the indexed units (further discussed in Section 2.2); they are usually words, and for the moment you can think of them as words, but the information retrieval literature normally speaks of terms because some of them, such as perhaps I-9 or Hong Kong are not usually thought of as words. Now, depending on whether we look at the matrix rows or columns, we can

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
...	*						

Figure 1.1 A term-document incidence matrix. Matrix element (t, d) is 1 if the play in column d contains the word in row t , and is 0 otherwise.

have a vector for each term, which shows the documents it appears in, or a vector for each document, showing the terms that occur in it.²

To answer the query *Brutus AND Caesar AND NOT Calpurnia*, we take the vectors for Brutus, Caesar and Calpurnia, complement the last, and then do a bitwise AND:

$$110100 \text{ AND } 110111 \text{ AND } 101111 = 100100$$

The answers for this query are thus *Antony and Cleopatra* and *Hamlet* (Figure 1.2).

The *Boolean retrieval model* is a model for information retrieval in which we can pose any query which is in the form of a Boolean expression of terms, that is, in which terms are combined with the operators AND, OR, and NOT. The model views each document as just a set of words.

Let us now consider a more realistic scenario, simultaneously using the opportunity to introduce some terminology and notation. Suppose we have $N = 1$ million documents. By *documents* we mean whatever units we have decided to build a retrieval system over. They might be individual memos or chapters of a book (see Section 2.1.2 (page 20) for further discussion). We refer to the group of documents over which we perform retrieval as the (document) *collection*. It is sometimes also referred to as a *corpus* (a *body* of texts).

Suppose each document is about 1,000 words long (2–3 book pages). If we assume an average of 6 bytes per word including spaces and punctuation, then this is a document collection about 6 gigabytes (GB) in size. Typically, there might be about $M = 500,000$ distinct terms in these documents. There is nothing special about the numbers we have chosen, and they might vary by an order of magnitude or more, but they give us some idea of the dimensions of the kinds of problems we need to handle. We will discuss and model these size assumptions in Section 5.1 (page 79).

Our goal is to develop a system to address the *ad hoc retrieval* task. This is the most standard IR task. In it, a system aims to provide documents from

² Formally, we take the transpose of the matrix to be able to get the terms as column vectors.