A PRINCETON UNIVERSITY PRESS E-BOOK

Google's PageRank and Beyond:

The Science of Search Engine Rankings

Amy N. Langville and Carl D. Meyer

PRINCETON

Google's PageRank and Beyond: The Science of Search Engine Rankings

Amy N. Langville and Carl D. Meyer

PRINCETON UNIVERSITY PRESS
PRINCETON AND OXFORD

Contents

This page intentionally left blank

Preface

Purpose

As teachers of linear algebra, we wanted to write a book to help students and the general public appreciate and understand one of the most exciting applications of linear algebra today—the use of link analysis by web search engines. This topic is inherently interesting, timely, and familiar. For instance, the book answers such curious questions as: How do search engines work? Why is Google so good? What's a Google bomb? How can I improve the ranking of my homepage in Teoma?

We also wanted this book to be a single source for material on web search engine rankings. A great deal has been written on this topic, but it's currently spread across numerous technical reports, preprints, conference proceedings, articles, and talks. Here we have summarized, clarified, condensed, and categorized the state of the art in web ranking.

Our Audience

We wrote this book with two diverse audiences in mind: the general science reader and the technical science reader. The title echoes the technical content of the book, but in addition to being informative on a technical level, we have also tried to provide some entertaining features and lighter material concerning search engines and how they work.

The Mathematics

Our goal in writing this book was to reach a challenging audience consisting of the general scientific public as well as the technical scientific public. Of course, a complete understanding of link analysis requires an acquaintance with many mathematical ideas. Nevertheless, we have tried to make the majority of the book accessible to the general scientific public. For instance, each chapter builds progressively in mathematical knowledge, technicality, and prerequisites. As a result, Chapters 1-4, which introduce web search and link analysis, are aimed at the general science reader. Chapters 6, 9, and 10 are particularly mathematical. The last chapter, Chapter 15, "The Mathematics Guide," is a condensed but complete reference for every mathematical concept used in the earlier chapters. Throughout the book, key mathematical concepts are highlighted in shaded boxes. By postponing the mathematical definitions and formulas until Chapter 15 (rather than interspersing them throughout the text), we were able to create a book that our mathematically sophisticated readers will also enjoy. We feel this approach is a compromise that allows us to serve both audiences: the general and technical scientific public.

Asides

An enjoyable feature of this book is the use of Asides. Asides contain entertaining news stories, practical search tips, amusing quotes, and racy lawsuits. Every chapter, even the particularly technical ones, contains several asides. Often times a light aside provides the perfect break after a stretch of serious mathematical thinking. Brief asides appear in shaded boxes while longer asides that stretch across multiple pages are offset by horizontal bars and italicized font. We hope you enjoy these breaks—we found ourselves looking forward to writing them.

Computing and Code

Truly mastering a subject requires experimenting with the ideas. Consequently, we have incorporated Matlab code to encourage and jump-start the experimentation process. While any programming language is appropriate, we chose Matlab for three reasons: (1) its matrix storage architecture and built-in commands are particularly suited to the large sparse link analysis matrices of this text, (2) among colleges and universities, Matlab is a market leader in mathematical software, and (3) it's very user-friendly. The Matlab programs in this book are intended to be instruction, not production, code. We hope that, by playing with these programs, readers will be inspired to create new models and algorithms.

Acknowledgments

We thank Princeton University Press for supporting this book. We especially enjoyed working with Vickie Kearn, the Senior Editor at PUP. Vickie, thank you for displaying just the right combination of patience and gentle pressure. For a book with such timely material, you showed amazing faith in us. We thank all those who reviewed our manuscripts and made this a better book. Of course, we also thank our families and friends for their encouragement. Your pride in us is a powerful driving force.

Dedication

We dedicate this book to mentors and mentees worldwide. The energy, inspiration, and support that is sparked through such relationships can inspire great products. For us, it produced this book, but more importantly, a wonderful synergistic friendship.

Chapter One

Introduction to Web Search Engines

1.1 A SHORT HISTORY OF INFORMATION RETRIEVAL

Today we have museums for everything—the museum of baseball, of baseball players, of crazed fans of baseball players, museums for world wars, national battles, legal fights, and family feuds. While there's no shortage of museums, we have yet to find a museum dedicated to this book's field, a museum of information retrieval and its history. Of course, there are related museums, such as the Library Museum in Boras, Sweden, but none concentrating on information retrieval. **Information retrieval**[1] is the process of searching within a document collection for a particular information need (called a **query**). Although dominated by recent events following the invention of the computer, information retrieval actually has a long and glorious tradition. To honor that tradition, we propose the creation of a museum dedicated to its history. Like all museums, our museum of information retrieval contains some very interesting artifacts. Join us for a brief tour.

The earliest document collections were recorded on the painted walls of caves. A cave dweller interested in searching a collection of cave paintings to answer a particular information query had to travel by foot, and stand, staring in front of each painting. Unfortunately, it's hard to collect an artifact without being gruesome, so let's fast forward a bit.

Before the invention of paper, ancient Romans and Greeks recorded information on papyrus rolls. Some papyrus artifacts from ancient Rome had tags attached to the rolls. These tags were an ancient form of today's Post-it Note, and make an excellent addition to our museum. A tag contained a short summary of the rolled document, and was attached in order to save readers from unnecessarily unraveling a long irrelevant document. These abstracts also appeared in oral form. At the start of Greek plays in the fifth century B.C., the chorus recited an abstract of the ensuing action. While no actual classification scheme has survived from the artifacts of Greek and Roman libraries, we do know that another elementary information retrieval tool, the table of contents, first appeared in Greek scrolls from the second century B.C. Books were not invented until centuries later, when necessity required an alternative writing material. As the story goes, the Library of Pergamum (in what is now Turkey) threatened to overtake the celebrated Library of Alexandria as the best library in the world, claiming the largest collection of papyrus rolls. As a result, the Egyptians ceased the supply of papyrus to Pergamum, so the Pergamenians invented an alternative writing material, parchment, which is made from thin layers of animal skin. (In fact, the root of the word *parchment* comes from the word *Pergamum*.) Unlike papyrus,

[1] The boldface terms that appear throughout the book are also listed and defined in the Glossary, which begins on page 201.

parchment did not roll easily, so scribes folded several sheets of parchment and sewed them into books. These books outlasted scrolls and were easier to use. Parchment books soon replaced the papyrus rolls.

The heights of writing, knowledge, and documentation of the Greek and Roman periods were contrasted with their lack during the Dark and Middle Ages. Precious few documents were produced during this time. Instead, most information was recorded orally. Document collections were recorded in the memory of a village's best storyteller. Oral traditions carried in poems, songs, and prayers were passed from one generation to the next. One of the most legendary and lengthy tales is *Beowulf*, an epic about the adventures of a sixth-century Scandinavian warrior. The tale is believed to have originated in the seventh century and been passed from generation to generation through song. Minstrels often took poetic license, altering and adding verses as the centuries passed. An inquisitive child wishing to hear stories about the monster Grendel waited patiently while the master storyteller searched his memory to find just the right part of the story. Thus, the result of the child's search for information was biased by the wisdom and judgement of the intermediary storyteller. Fortunately, the invention of paper, the best writing medium yet, superior to even parchment, brought renewed acceleration to the written record of information and collections of documents. In fact, Beowulf passed from oral to written form around A.D. 1000, a date over which scholars still debate. Later, monks, the possessors of treasured reading and writing skills, sat in scriptoriums working as scribes from sunrise to sunset. The scribes' works were placed in medieval libraries, which initially were so small that they had no need for classification systems. Eventually the collections grew, and it became common practice to divide the holdings into three groups: theological works, classical authors of antiquity, and contemporary authors on the seven arts. Lists of holdings and tables of contents from classical books make nice museum artifacts from the medieval period.

Other document collections sprung up in a variety of fields. This dramatically accelerated with the re-invention of the printing press by Johann Gutenberg in 1450. The wealthy proudly boasted of their private libraries, and public libraries were instituted in America in the 1700s at the prompting of Benjamin Franklin. As library collections grew and became publicly accessible, the desire for focused search became more acute. Hierarchical classification systems were used to group documents on like subjects together. The first use of a hierarchical organization system is attributed to the Roman author Valerius Maximus, who used it in A.D. 30 to organize the topics in his book, *Factorum ac dictorum memorabilium libri IX* (Nine Books of Memorable Deeds and Sayings). Despite these rudimentary organization systems, word of mouth and the advice of a librarian were the best means of obtaining accurate quality information for a search. Of course, document collections and their organization expanded beyond the limits of even the best librarian's memory. More orderly ways of maintaining records of a collection's holdings were devised. Notable artifacts that belong in our information retrieval museum are a few lists of individual library holdings, sorted by title and also author, as well as examples of the Dewey decimal system (1872), the card catalog (early 1900s), microfilm (1930s), and the MARC (MAchine Readable Cataloging) system (1960s).

These inventions were progress, yet still search was not completely in the hands of the information seeker. It took the invention of the digital computer (1940s and 1950s) and the subsequent inventions of computerized search systems to move toward that goal. The first computerized search systems used special syntax to automatically retrieve book and article information related to a user's query. Unfortunately, the cumbersome syntax kept search largely in the domain of librarians trained on the systems. An early representative of computerized search such as the Cornell SMART system (1960s) [146] deserves a place in our museum of information retrieval.

In 1989 the storage, access, and searching of document collections was revolutionized by an invention named the World Wide Web by its founder Tim Berners-Lee [79]. Of course, our museum must include artifacts from this revolution such as a webpage, some HTML, and a hyperlink or two. The invention of linked document collections was truly original at this time, despite the fact that Vannevar Bush, once Director of the Office of Scientific Research and Development, foreshadowed its coming in his famous 1945 essay, "As We May Think" [43]. In that essay, he describes the memex, a futuristic machine (with shocking similarity to today's PC and Web) that mirrors the cognitive processes of humans by leaving "trails of association" throughout document collections. Four decades of progress later, remnants of Bush's memex formed the skeleton of Berners-Lee's Web. A drawing of the memex (Figure 1.1) by a graphic artist and approved by Bush was included in *LIFE* magazine's 1945 publishing of Bush's prophetic article.

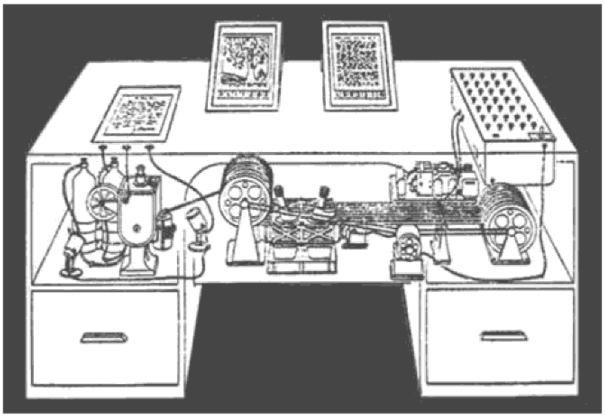

Figure 1.1 Drawing of Vannevar Bush's memex appearing in *LIFE*. Original caption read: "Memex in the form of a desk would instantly bring files and material on any subject to the operator's fingertips. Slanting translucent screens supermicrofilm filed by code numbers. At left is a mechanism which automatically photographs longhand notes, pictures, and letters, then files them in the desk for future reference."

The World Wide Web became the ultimate signal of the dominance of the Information Age and the death of the Industrial Age. Yet despite the revolution in information storage and access ushered in by the Web, users initiating web searches found themselves floundering. They were looking for the proverbial needle in an enormous, ever-growing information haystack. In fact, users felt much like the men in Jorge Luis Borges' 1941 short story [35], "The Library of Babel", which describes an imaginary, infinite library.

> When it was proclaimed that the Library contained all books, the first impression was one of extravagant happiness. All men felt themselves to be the masters of an intact and secret treasure. There was no personal or world problem whose eloquent solution did not exist in some hexagon.

. . . As was natural, this inordinate hope was followed by an excessive depression. The certitude that some shelf in some hexagon held precious books and that these precious books were inaccessible seemed almost intolerable.

Much of the information in the Library of the Web, like that in the fictitious Library of Babel, remained inaccessible. In fact, early web search engines did little to ease user frustration; search could be conducted by sorting through hierarchies of topics on Yahoo, or by sifting through the many (often thousands of) webpages returned by the search engine, clicking on pages to personally determine which were most relevant to the query. Some users resorted to the earliest search techniques used by ancient queriers—word of mouth and expert advice. They learned about valuable websites from friends and linked to sites recommended by colleagues who had already put in hours of search effort.

All this changed in 1998 when **link analysis** hit the information retrieval scene [40, 106]. The most successful search engines began using link analysis, a technique that exploited the additional information inherent in the hyperlink structure of the Web, to improve the quality of search results. Web search improved dramatically, and web searchers religiously used and promoted their favorite engines like Google and AltaVista. In fact, in 2004 many web surfers freely admit their obsession with, dependence on, and addiction to today's search engines. Below we include the comments [117] of a few Google fans to convey the joy caused by the increased accessibility of the Library of the Web made possible by the link analysis engines. Incidentally, in May 2004 Google held the largest share of the search market with 37% of searchers using Google, followed by 27% using the Yahoo conglomerate, which includes AltaVista, AlltheWeb, and Overture.[2]

- "It's not my homepage, but it might as well be. I use it to ego-surf. I use it to read the news. Anytime I want to find out anything, I use it."—Matt Groening, creator and executive producer, *The Simpsons*
- "I can't imagine life without Google News. Thousands of sources from around the world ensure anyone with an Internet connection can stay informed. The diversity of viewpoints available is staggering."—Michael Powell, chair, Federal Communications Commission
- "Google is my rapid-response research assistant. On the run-up to a deadline, I may use it to check the spelling of a foreign name, to acquire an image of a particular piece of military hardware, to find the exact quote of a public figure, check a stat, translate a phrase, or research the background of a particular corporation. It's the Swiss Army knife of information retrieval."—Garry Trudeau, cartoonist and creator, *Doonesbury*

Nearly all major search engines now combine link analysis scores, similar to those used by Google, with more traditional information retrieval scores. In this book, we record the history of one aspect of *web* information retrieval. That aspect is the link analysis or *ranking* algorithms underlying several of today's most popular and successful search

[2]These market share statistics were compiled by comScore, a company that counted the number of searches done by U.S. surfers in May 2004 using the major search engines. See the article at `http://searchenginewatch.com/reports/article.php/2156431`.

engines, including Google and Teoma. Incidentally, we'll add the **PageRank** link analysis algorithm [40] used by Google (see Chapters 4-10) and the **HITS** algorithm [106] used by Teoma (see Chapter 11) to our museum of information retrieval.

1.2 AN OVERVIEW OF TRADITIONAL INFORMATION RETRIEVAL

To set the stage for the exciting developments in link analysis to come in later chapters, we begin our story by distinguishing **web information retrieval** from **traditional information retrieval**. Web information retrieval is search within the world's largest and linked document collection, whereas traditional information retrieval is search within smaller, more controlled, nonlinked collections. The traditional nonlinked collections existed before the birth of the Web and still exist today. Searching within a university library's collection of books or within a professor's reserve of slides for an art history course—these are examples of traditional information retrieval.

These document collections are nonlinked, mostly static, and are organized and categorized by specialists such as librarians and journal editors. These documents are stored in physical form as books, journals, and artwork as well as electronically on microfiche, CDs, and webpages. However, the mechanisms for searching for items in the collections are now almost all computerized. These computerized mechanisms are referred to as search engines, virtual machines created by software that enables them to sort through virtual file folders to find relevant documents. There are three basic computer-aided techniques for searching traditional information retrieval collections: Boolean models, vector space models, and probabilistic models [14]. These search models, which were developed in the 1960s, have had decades to grow, mesh, and morph into new search models. In fact, as of June 2000, there were at least 3,500 different search engines (including the newer web engines) [37], which means that there are possibly 3,500 different search techniques. Nevertheless, since most search engines rely on one or more of the three basic models, we describe these in turn.

1.2.1 Boolean Search Engines

The **Boolean model** of information retrieval, one of the earliest and simplest retrieval methods, uses the notion of exact matching to match documents to a user query. Its more refined descendents are still used by most libraries. The adjective Boolean refers to the use of Boolean algebra, whereby words are logically combined with the Boolean operators AND, OR, and NOT. For example, the Boolean AND of two logical statements x and y means that both x AND y must be satisfied, while the Boolean OR of these two statements means that at least one of these statements must be satisfied. Any number of logical statements can be combined using the three Boolean operators. The Boolean model of information retrieval operates by considering which keywords are present or absent in a document. Thus, a document is judged as relevant or irrelevant; there is no concept of a partial match between documents and queries. This can lead to poor performance [14]. More advanced fuzzy set theoretic techniques try to remedy this black-white Boolean logic by introducing shades of gray. For example, a title search for `car` AND `maintenance` on a Boolean engine causes the virtual machine to return all documents that use both words in the title. A relevant document entitled "Automobile Maintenance" will not be returned. Fuzzy Boolean engines use fuzzy logic to categorize this document as somewhat relevant and return it to the user.

The car maintenance query example introduces the main drawbacks of Boolean search engines; they fall prey to two of the most common information retrieval problems, **synonymy** and **polysemy**. Synonymy refers to multiple words having the same meaning, such as car and automobile. A standard Boolean engine cannot return semantically related documents whose keywords were not included in the original query. Polysemy refers to words with multiple meanings. For example, when a user types `bank` as their query, does he or she mean a financial center, a slope on a hill, a shot in pool, or a collection of objects [24]? The problem of polysemy can cause many documents that are irrelevant to the user's actual intended query meaning to be retrieved. Many Boolean search engines also require that the user be familiar with Boolean operators and the engine's specialized syntax. For example, to find information about the phrase `iron curtain`, many engines require quotation marks around the phrase, which tell the search engine that the entire phrase should be searched as if it were just one keyword. A user who forgets this syntax requirement would be surprised to find retrieved documents about interior decorating and mining for iron ore.

Nevertheless, variants of the Boolean model do form the basis for many search engines. There are several reasons for their prevalence. First, creating and programming a Boolean engine is straightforward. Second, queries can be processed quickly; a quick scan through the keyword files for the documents can be executed in parallel. Third, Boolean models scale well to very large document collections. Accommodating a growing collection is easy. The programming remains simple; merely the storage and parallel processing capabilities need to grow. References [14, 75, 107] all contain chapters with excellent introductions to the Boolean model and its extensions.

1.2.2 Vector Space Model Search Engines

Another information retrieval technique uses the **vector space model** [147], developed by Gerard Salton in the early 1960s, to sidestep some of the information retrieval problems mentioned above. Vector space models transform textual data into *numeric vectors* and *matrices*, then employ *matrix analysis*[3] techniques to discover key features and connections in the document collection. Some advanced vector space models address the common text analysis problems of synonymy and polysemy. Advanced vector space models, such as LSI [64] (Latent Semantic Indexing), can access the hidden semantic structure in a document collection. For example, an LSI engine processing the query `car` will return documents whose keywords are related semantically (in meaning), e.g., `automobile`. This ability to reveal hidden semantic meanings makes vector space models, such as LSI, very powerful information retrieval tools.

Two additional advantages of the vector space model are **relevance scoring** and **relevance feedback**. The vector space model allows documents to partially match a query by assigning each document a number between 0 and 1, which can be interpreted as the likelihood of relevance to the query. The group of retrieved documents can then be sorted by degree of relevancy, a luxury not possible with the simple Boolean model. Thus, vector space models return documents in an ordered list, sorted according to a relevance score. The first document returned is judged to be most relevant to the user's query.

[3]Mathematical terms are defined in Chapter 15, the Mathematics Chapter, and are italicized throughout.

Some vector space search engines report the relevance score as a relevancy percentage. For example, a 97% next to a document means that the document is judged as 97% relevant to the user's query. (See the Federal Communications Commission's search engine, `http://www.fcc.gov/searchtools.html`, which is powered by *Inktomi*, once known to use the vector space model. Enter a query such as `taxes` and notice the relevancy score reported on the right side.) Relevance feedback, the other advantage of the vector space model, is an information retrieval tuning technique that is a natural addition to the vector space model. Relevance feedback allows the user to select a subset of the retrieved documents that are useful. The query is then resubmitted with this additional relevance feedback information, and a revised set of generally more useful documents is retrieved.

A drawback of the vector space model is its computational expense. At query time, distance measures (also known as similarity measures) must be computed between each document and the query. And advanced models, such as LSI, require an expensive singular value decomposition [82, 127] of a large matrix that numerically represents the entire document collection. As the collection grows, the expense of this matrix decomposition becomes prohibitive. This computational expense also exposes another drawback—vector space models do not scale well. Their success is limited to small document collections.

Understanding Search Engines

The informative little book by Michael Berry and Murray Browne, *Understanding Search Engines: Mathematical Modeling and Text Retrieval* [23], provides an excellent explanation of vector space models, especially LSI, and contains several examples and sample code. Our mathematical readers will enjoy this book and its application of linear algebra algorithms in the context of traditional information retrieval.

1.2.3 Probabilistic Model Search Engines

Probabilistic models attempt to estimate the probability that the user will find a particular document relevant. Retrieved documents are ranked by their odds of relevance (the ratio of the probability that the document is relevant to the query divided by the probability that the document is not relevant to the query). The probabilistic model operates recursively and requires that the underlying algorithm guess at initial parameters then iteratively tries to improve this initial guess to obtain a final ranking of relevancy probabilities.

Unfortunately, probabilistic models can be very hard to build and program. Their complexity grows quickly, deterring many researchers and limiting their scalability. Probabilistic models also require several unrealistic simplifying assumptions, such as independence between terms as well as documents. Of course, the independence assumption is restrictive in most cases. For instance, in this document the most likely word to follow `information` is the word `retrieval`, but the independence assumption judges each word as equally likely to follow the word `information`. On the other hand, the probabilistic framework can naturally accommodate a priori preferences, and thus, these models do offer promise of tailoring search results to the preferences of individual users. For example, a user's query history can be incorporated into the probabilistic model's initial guess, which generates better query results than a democratic guess.

1.2.4 Meta-search Engines

There's actually a fourth model for traditional search engines, meta-search engines, which combines the three classic models. **Meta-search engines** are based on the principle that while one search engine is good, two (or more) are better. One search engine may be great at a certain task, while a second search engine is better at another task. Thus, meta-search engines such as *Copernic* (`www.copernic.com`) and *SurfWax* (`www.surfwax.com`) were created to simultaneously exploit the best features of many individual search engines. Meta-search engines send the query to several search engines at once and return the results from all of the search engines in one long unified list. Some meta-search engines also include subject-specific search engines, which can be helpful when searching within one particular discipline. For example, *Monster* (`www.monster.com`) is an employment search engine.

1.2.5 Comparing Search Engines

Annual information retrieval conferences, such as TREC [3], SIGIR, CIR [22] (for traditional information retrieval), and WWW [4] (for web information retrieval), are used to compare the various information retrieval models underlying search engines and help the field progress toward better, more efficient search engines. The two most common ratings used to differentiate the various search techniques are precision and recall. **Precision** is the ratio of the number of relevant documents retrieved to the total number of documents retrieved. **Recall** is the ratio of the number of relevant documents retrieved to the total number of relevant documents in the collection. The higher the precision and recall, the better the search engine is. Of course, search engines are tested on document collections with known parameters. For example, the commonly used test collection Medlars [6], containing 5,831 keywords and 1,033 documents, has been examined so often that its properties are well known. For instance, there are exactly 24 documents relevant to the phrase `neoplasm immunology`. Thus, the denominator of the recall ratio for a user query on `neoplasm immunology` is 24. If only 10 documents were retrieved by a search engine for this query, then a recall of $10/24 = .41\overline{6}$ is reported. Recall and precision are information retrieval-specific performance measures, but, of course, when evaluating any computer system, time and space are always performance issues. All else held constant, quick, memory-efficient search engines are preferred to slower, memory-inefficient engines. A search engine with fabulous recall and precision is useless if it requires 30 minutes to perform one query or stores the data on 75 supercomputers. Some other performance measures take a user-centered viewpoint and are aimed at assessing user satisfaction and frustration with the information system. A book by Robert Korfhage, *Information Storage and Retrieval* [107], discusses these and several other measures for comparing search engines. Excellent texts for information retrieval are [14, 75, 163].

1.3 WEB INFORMATION RETRIEVAL

1.3.1 The Challenges of Web Search

Tim Berners-Lee and his World Wide Web entered the information retrieval world in 1989 [79]. This event caused a branch that focused specifically on search within this new document collection to break away from traditional information retrieval. This branch is called **web information retrieval**. Many web search engines are built on the techniques of traditional search engines, but they differ in many important ways. We list the properties that make the Web such a unique document collection. The Web is:

- huge,
- dynamic,
- self-organized, and
- hyperlinked.

The Web is indeed huge! In fact, it's so big that it's hard to get an accurate count of its size. By January 2004, it was estimated that the Web contained over 10 billion pages, with an average page size of 500KB [5]. With a world population of about 6.4 billion, that's almost 2 pages for each inhabitant. The early exponential growth of the Web has slowed recently, but it is still the largest document collection in existence. The Berkeley information project, "How Much Information," estimates that the amount of information on the Web is about 20 times the size of the entire Library of Congress print collection [5]. Bigger still, a company called BrightPlanet sells access to the so-called Deep Web, which they estimate to contain over 92,000TB of data spread over 550 billion pages [1]. BrightPlanet defines the Deep Web as the hundreds of thousands of publicly accessible databases that create a collection over 500 times larger than the Surface Web. Deep webpages can not be found by casual, routine surfing. Surfers must request information from a particular database, at which point, the relevant pages are served to the user dynamically within a matter of seconds. As a result, search engines cannot easily find these dynamic pages since they do not exist before or after the query. However, Yahoo appears to be the first search engine aiming to index parts of the Deep Web.

The Web is dynamic! Contrast this with traditional document collections which can be considered static in two senses. First, once a document is added to a traditional collection, it does not change. The books sitting on a bookshelf are well behaved. They don't change their content by themselves, but webpages do, very frequently. A study by Junghoo Cho and Hector Garcia-Molina [52] in 2000 reported that 40% of all webpages in their dataset changed within a week, and 23% of the .com pages changed daily. In a much more extensive and recent study, the results of Fetterly et al. [74] concur. About 35% of all webpages changed over the course of their study, and also pages that were larger in size changed more often and more extensively than their smaller counterparts. Second, for the most part, the size of a traditional document collection is relatively static. It is true that abstracts are added to MEDLINE each year, but how many? Hundreds, maybe thousands. These are minuscule additions by Web proportions. Billions of pages are added to the Web each year. The dynamics of the Web make it tough to compute relevancy scores for queries when the collection is a moving, evolving target.

do the data collection and categorization tasks on their own. As a result, all web search engines have a crawler module. This module contains the software that collects and categorizes the web's documents. The crawling software creates virtual robots, called **spiders**, that constantly scour the Web gathering new information and webpages and returning to store them in a central repository.

- **Page Repository.** The spiders return with new webpages, which are temporarily stored as full, complete webpages in the page repository. The new pages remain in the repository until they are sent to the indexing module, where their vital information is stripped to create a compressed version of the page. Popular pages that are repeatedly used to serve queries are stored here longer, perhaps indefinitely.

- **Indexing Module.** The indexing module takes each new uncompressed page and extracts only the vital descriptors, creating a compressed description of the page that is stored in various indexes. The indexing module is like a black box function that takes the uncompressed page as input and outputs a "Cliffnotes" version of the page. The uncompressed page is then tossed out or, if deemed popular, returned to the page repository.

- **Indexes.** The indexes hold the valuable compressed information for each webpage. This book describes three types of indexes. The first is called the **content index**. Here the content, such as keyword, title, and anchor text for each webpage, is stored in a compressed form using an **inverted file** structure. Chapter 2 describes the inverted file in detail. Further valuable information regarding the hyperlink structure of pages in the search engine's index is gleaned during the indexing phase. This link information is stored in compressed form in the **structure index**. The crawler module sometimes accesses the structure index to find uncrawled pages. **Special-purpose indexes** are the final type of index. For example, indexes such as the image index and pdf index hold information that is useful for particular query tasks.

The four modules above (crawler, page repository, indexers, indexes) and their corresponding data files exist and operate independent of users and their queries. Spiders are constantly crawling the Web, bringing back new and updated pages to be indexed and stored. In Figure 1.2 these modules are circled and labeled as **query-independent**. Unlike the preceding modules, the query module is **query-dependent** and is initiated when a user enters a query, to which the search engine must respond in **real-time**.

- **Query Module.** The query module converts a user's natural language query into a language that the search system can understand (usually numbers), and consults the various indexes in order to answer the query. For example, the query module consults the content index and its inverted file to find which pages use the query terms. These pages are called the relevant pages. Then the query module passes the set of relevant pages to the ranking module.

- **Ranking Module.** The ranking module takes the set of relevant pages and ranks them according to some criterion. The outcome is an ordered list of webpages such that the pages near the top of the list are most likely to be what the user desires. The ranking module is perhaps the most important component of the search process because the output of the query module often results in too many (thousands of) relevant pages that the user must sort through. The ordered list filters the less relevant pages to the bottom, making the list of pages more manageable for the user. (In contrast, the similarity measures of traditional information retrieval often do not filter out enough irrelevant pages.) Actually, this ranking which carries valuable, discriminatory power is arrived at by combining two scores, the **content score** and the **popularity score**. Many rules are used to give each relevant page a relevancy or content score. For example, many web engines give pages using the query word in the title or description a higher content score than pages using the query word in the body of the page [39]. The popularity score, which is the focus of this book, is determined from an analysis of the Web's hyperlink structure. The content score is combined with the popularity score to determine an **overall score** for each relevant page [30]. The set of relevant pages resulting from the query module is then presented to the user in order of their overall scores.

Chapter 2 gives an introduction to all components of the web search process, except the ranking component. The ranking component, specifically the popularity score, is the subject of this book. Chapters 3 through 12 provide a comprehensive treatment of the ranking problem and its suggested solutions. Each chapter progresses in depth and mathematical content.

This page intentionally left blank

Chapter Two

Crawling, Indexing, and Query Processing

Spiders are the building blocks of search engines. Decisions about the design of the crawler and the capabilities of its spiders affect the design of the other modules, such as the indexing and query processing modules.

So in this chapter, we begin our description of the basic components of a web search engine with the crawler and its spiders. We purposely exclude one component, the ranking component, since it is the focus of this book and is covered in the remaining chapters. The goals and challenges of web crawlers are introduced in section 2.1, and a simple program for crawling the Web is provided. Indexing a collection of documents as enormous as the Web creates special storage challenges (section 2.2), and also has search engines constantly increasing the size of their indexes (see the aside on page 20). The size of the Web makes the real-time processing of queries an astounding feat, and section 2.3 describes the structures and mechanisms that make this possible.

2.1 CRAWLING

The crawler module contains a short software program that instructs robots or spiders on how and which pages to retrieve. The crawling module gives a spider a root set of URLs to visit, instructing it to start there and follow links on those pages to find new pages. Every crawling program must address several issues. For example, which pages should the spiders crawl? Some search engines focus on specialized search, and as a result, conduct specialized crawls, through only `.gov` pages, or pages with images, or blog files, etc. For instance, Bernhard Seefeld's search engine, search.ch, crawls only Swiss webpages and stops at the geographical borders of Switzerland. Even the most comprehensive search engine indexes only a small portion of the entire Web. Thus, crawlers must carefully select which pages to visit.

How often should pages be crawled? Since the Web is dynamic, last month's crawled page may contain different content this month. Therefore, crawling is a never-ending process. Spiders return exhausted, carrying several new and many updated pages, only to be immediately given another root URL and told to start over. Theirs is an endless task like Sisyphus's uphill ball-rolling. However, some pages change more often than others, so a crawler must decide which pages to revisit and how often. Some engines make this decision democratically, while others refresh pages in proportion to their perceived freshness or importance levels. In fact, some researchers have proposed a crawling strategy that uses the PageRank measure of Chapters 3 and 4 to decide which pages to update [31].

How should pages be crawled ethically? When a spider visits a webpage, it consumes resources, such as bandwidth and hits quotas, belonging to the page's host and the

Internet at large. Like outdoor activists who try to "leave no trace," polite spiders try to minimize their impact. The Robots Exclusion Protocol was developed to define proper spidering activities and punish obnoxious, disrespectful spiders. In fact, website administrators can use a `robots.txt` file to block spiders from accessing parts of their sites.

How should multiple spiders coordinate their activities to avoid coverage overlap? One crawler can set several spiders loose on the Web, figuring parallel crawling can save time and effort. However, an optimal crawling policy is needed to insure websites are not visited multiple times, and thus significant overhead communication is required.

Regardless of the ways a crawling program addresses these issues, spiders return with URLs for new or refreshed pages that need to be added to or updated in the search engine's indexes. We discuss one index in particular, the content index, in the next section.

Submitting a Site to Search Engines

Like a castaway stranded on a tiny island, many webpage authors worry that a search engine spider might never find their webpage. This is certainly possible, especially if the page is about an obscure topic, and contains little content and few inlinks. Authors hosting a new page can check if spiders such as Googlebot have visited their site by viewing their web server's log files. Most search engines have mechanisms to calm the fears of castaway authors. For example, Google offers authors a submission feature. Every webpage author can submit his or her site through a web form (`http://www.google.com/addurl.html`), which adds the site to Google's list of to-be-crawled URLs. While Google offers no guarantees on if or when the site will be crawled, this service does help both site authors and the Google crawler. Almost all major search engines offer a "Submit Your Site" feature, although some require small fees in exchange for a listing, featured listing, or sponsored listing in their index.

Spidering Hacks

Readers interested in programming their own special purpose crawler will find the O'Reilly book, *Spidering Hacks* [93], useful. This book contains 100 tips and tools for training a spider to do just about anything. With these tricks, your spider will be able to do more than just sit, roll over, and play dead; he'll go find news stories about an actor, retrieve stock quotes, run an email discussion group, or find current topical trends on the Web.

```
      if i > 0
         disp(['     link ' int2str(i) ' ' url])
         L(i,j) = 1;
      end
   end
end

%------------------------
function h = hashfun(url)
% Almost unique numeric hash code for pages already visited.
h = length(url) + 1024*sum(url);
```

2.2 THE CONTENT INDEX

Each new or refreshed page that a spider brings back is sent to the indexing module, where software programs parse the page content and strip it of its valuable information, so that only the essential skeleton of the page is passed to the appropriate indexes. Valuable information is contained in title, description, and **anchor text** as well as in bolded terms, terms in large font, and hyperlinks. One important index is the content index, which stores the textual information for each page in compressed form. An inverted file, which is used to store this compressed information, is like the index in the back of a book. Next to each term is a list of all locations where the term appears. In the simplest case, the location is the page identifier. An inverted file might look like:

- term 1 (aardvark) - 3, 117, 3961

  $\vdots$

- term 10 (aztec) - 3, 15, 19, 101, 673, 1199
- term 11 (baby) - 3, 31, 56, 94, 673, 909, 11114, 253791

  $\vdots$

- term m (zymurgy) - 1159223

This means that term 1 is used in webpages 3, 117, and 3961. It is clear that an advantage of the inverted file is its use as a quick lookup table. Processing a query on term 11 begins by consulting the inverted list for term 11.

The simple inverted file, a staple in traditional information retrieval [147], does pose some challenges for web collections. Because multilingual terms, phrases, and proper names are used, the number of terms m, and thus the file size, is huge. Also, the number of webpages using popular broad terms such as *weather* or *sports* is large. Therefore, the number of page identifiers next to these terms is large and consumes storage. Further, page identifiers are usually not the only descriptors stored for each term. See section 2.3. Other descriptors such as the location of the term in the page (title, description, or body) and the appearance of the term (bolded, large font, or in anchor text) are stored next to each page identifier. Any number of descriptors can be used to aid the search engine in retrieving relevant documents. In addition, as pages change content, so must their compressed representation in the inverted file. Thus, an active area of research is the design of methods for efficiently updating indexes. Lastly, the enormous inverted file must be stored on a distributed architecture, which means strategies for optimal partitioning must be designed.

ASIDE: Indexing Wars

While having a larger index of webpages accessed does not necessarily make one search engine better than another, it does mean the "bigger" search engine has a better opportunity to return a longer list of relevant results, especially for unusual queries. As a result, search engines are constantly battling for the title of "The World's Largest Index." Reporters writing for The Search Engine Showdown *or* Search Engine Watch *enjoy charting the changing leaders in the indexing war. Figure 2.1 shows how self-reported search engine sizes have changed over the years.*

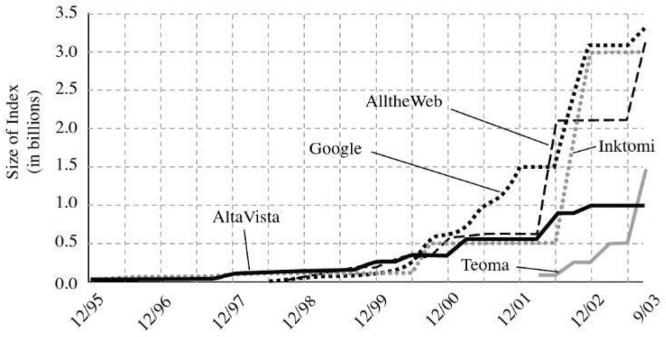

Figure 2.1 Growth of index for major search engines

Google, whose name is a play on googol, the word for the number 10^{100}, entered the search market in 1998 and immediately grew, dethroning AltaVista and claiming the title of the World's Largest Index. In 2002, AlltheWeb snatched the title from Google by declaring it had reached the two billion mark. Google soon regained the lead by indexing three billion pages. AlltheWeb and Inktomi quickly upped their sizes to hit this same mark. The search latecomer Teoma has been steadily growing its index since its debut in early 2002. Web search engines use elaborate schemes, structures, and machines to store their massive indices. In fact, in 2003, Google used a network of over 15,000 computers to store their index [19], which in November 2004 jumped from 4.3 billion to 8.1 billion webpages. The number of servers used today is at least an order of magnitude higher. Figure 2.2 shows part of the server system that is housed in the Googleplex Mountain View, California site. Google history buffs can see the dramatic evolution of Google's server system by viewing pictures of their original servers that used a Lego-constructed cabinet to house disk drives and cooling fans (`http://www-db.stanford.edu/pub/voy/museum/pictures/display/0-4-Google.htm`*).*

Figure 2.2 Google servers
© Timothy Archibald, 2006