What can you do with a Web in your Pocket?

Sergey Brin * Rajeev Motwani † Lawrence Page ‡ Terry Winograd §

Abstract

The amount of information available online has grown enormously over the past decade. Fortunately, computing power, disk capacity, and network bandwidth have also increased dramatically. It is currently possible for a university research project to store and process the entire World Wide Web. Since there is a limit on how much text humans can generate, it is plausible that within a few decades one will be able to store and process all the human-generated text on the Web in a shirt pocket.

The Web is a very rich and interesting data source. In this paper, we describe the Stanford WebBase, a local repository of a significant portion of the Web. Furthermore, we describe a number of recent experiments that leverage the size and the diversity of the WebBase. First, we have largely automated the process of extracting a sizable relation of books (title, author pairs) from hundreds of data sources spread across the World Wide Web using a technique we call Dual Iterative Pattern Relation Extraction. Second, we have developed a global ranking of Web pages called PageRank based on the link structure of the Web that has properties that are useful for search and navigation. Third, we have used PageRank to develop a novel search engine called Google, which also makes heavy use of anchor text. All of these experiments rely significantly on the size and diversity of the WebBase.

1 Introduction

The Web is a very diverse data source combining highly structured HTML, fully general natural language contained within the HTML, and embedded images. On top of this data is a reasonably well defined link structure, which is a directed graph over all the Web pages, with labels on the links (the text of the anchors).

We have found many applications for this data, some of which we will describe here.

- The extraction of structured data from many unstructured pieces spread throughout the Web

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Partially supported by the Community Management Staff's Massive Digital Data Systems Program, NSF grant IRI–96–31952, and grants of IBM and Hitachi Corp.

†Supported by an NSF Young Investigator Award CCR–9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

‡Supported by the Stanford Integrated Digital Library Project (see below).

§Supported by the Stanford Integrated Digital Library Project. This project is supported by NSF Cooperative Agreement IRI-9411306. Funding for this cooperative agreement is also provided by DARPA and NASA, and by Interval Research, and the industrial partners of the Stanford Digital Libraries Project.

If such a process is sufficiently accurate, it can provide a comprehensive and queryable source of information. We present a technique and some initial results for automatically extracting relations from the Web in Section 3.

- Enhanced information retrieval.

  This area has already been explored by a number of researchers and search engine companies. However, there is plenty of room for improvement in information retrieval on the Web, and we present some novel techniques in Sections 4 and 5.

In the work presented in this paper we take advantage of one central idea: the Web provides its own metadata through its link structure, anchor text, and partially redundant content. This is because a substantial portion *of* the Web is *about* the Web. To take full advantage of this data would require human intelligence or more. However, simple techniques that focus on a small subset of the potentially useful data can succeed due to the scale of the web. A technique that might capture only one percent of the available information might still be very useful. Because size is so crucial to these techniques, and the amount of publicly available information is likely to grow rapidly, it is important to maintain a large Web repository like our Stanford WebBase.

2 The Stanford WebBase

The size of the World Wide Web is an elastic number. There are many automatically generated infinite Web spaces, there is a large amount of duplication, and the Web changes every day. Nonetheless, a recent estimate has put its size at about 200 million web pages in November 1997 [BB98].

In 1994, one of the first web search engines, the World Wide Web Worm [McB94], had an index of 110,000 web pages and web accessible documents. As of March, 1998, the largest search engines claim to index from 2 to 110 million web documents [Sul]. It is foreseeable that by the year 2000, a comprehensive index of the Web will contain over a billion documents.

The Stanford WebBase is designed to store a significant subset of the text content of the Web for research purposes. It currently holds roughly 25 million web pages and will soon expand to approximately double that size. The repository itself is roughly 150 GB of HTML (stored compressed in 50 GB). Additionally, we keep an inverted index of the text, which includes word position and font information, occupying an additional 50 GB. Finally, various metadata including URL's and link structure occupy another 10 GB .

By traditional standards, the repository contains a huge amount of information. However, since the acquisition and indexing can be almost fully automated, the repository can be collected, processed, and maintained by a university research project. As disk space, network bandwidth, and computing power continue to improve and fall in price, eventually a school child will be able to store a comparable collection on a portable computer.

3 Extraction of Relations

The World Wide Web provides a vast resource for information. At the same time it is extremely distributed. A particular type of information such as restaurant lists may be scattered across thousands of independent information sources in many different formats. If these chunks of information could be extracted from the World Wide Web and integrated into a structured form, they would form an unprecedented source of data, such as a combined international directory of people, the largest and most diverse databases of products, and the broadest bibliography of academic works.

There has been considerable work on integrating multiple information sources using specially coded wrappers or filters [Tsi, MOS97]. However, these can be time-consuming to create and maintain and are usually used for tens, not thousands of sources. In this section, we address the problem of extracting a relation from the thousands of sources that may hold pieces of the relation on the World Wide Web. Our goal is to discover information sources and to extract the relevant information from them either entirely automatically, or with very minimal human intervention.

3.1 Dual Iterative Pattern Relation Expansion (DIPRE)

Our test problem is to extract a relation of books – (author,title) pairs from the Web. Intuitively, our solution works as follows. We begin with a small seed set of (author, title) pairs (in tests we used a set of just five pairs). Then we find all occurrences of those books on the Web (an occurrence of a book is the appearance of the title and author in close proximity on the same Web page). From these occurrences we recognize patterns for the citations of books. Then we search the Web for these patterns and find new books. We can then take these books, find all their occurrences, and from those generate more patterns. We can use these new patterns to find more books, and so forth. Eventually, we will obtain a large list of books and patterns for finding them. We call this technique DIPRE - Dual Iterative Pattern Relation Expansion. In the process of applying DIPRE, it is much more important to be precise (not generate too much garbage) than it is to be complete, because the effect of adding false patterns is a proliferation of false entries.

Thus far we have been vague about the concepts of an occurrence and a pattern. They can be formalized in a number of ways. We chose a very simple approach to demonstrate the general technique:

An occurrence of an *(author,title)* pair is represented as a seven-tuple: *(author, title, order, url, prefix, middle, suffix)*. The *order* is a boolean value that determines whether the author appears before the title or the title appears before the author. The *url* is the URL of the document they occurred on. The *prefix* consists of the m characters (in tests m was 10) preceding the author (or title if the title was first). The *middle* is the text between the author and title, and the *suffix* consists of the m characters following the title (or author).

A pattern is a five-tuple: *(order, urlprefix, prefix, middle, suffix)*. The $order$ is boolean and the other attributes are strings. If *order* is true, an *(author,title)* pair matches the pattern if there is a document in the collection (the WWW) with a URL that matches *urlprefix*$*$ and which contains text that matches the regular expression: $*$*prefix, <author>, middle, <title>, suffix*$*$. If order is false, then the author and title are reversed.

Let R be the relation we are extracting, P be the set of patterns we are extracting, and let O be a set of occurrences. DIPRE works as follows:

1. $R \leftarrow$ Sample
   Initialize R with a small seed of the relation we are trying to find. In our tests it was just a list of five author,title pairs.

2. $O \leftarrow$ FindOccurrencesA(R)
   Find all occurrences of tuples of R on the Web.

3. $P \leftarrow$ GenPatterns(O)
   Generate patterns from the set of occurrences.

4. $O \leftarrow$ FindOccurrencesB(P)
   Find all occurrences of the patterns on the Web.

5. $R \leftarrow R \cup$ Tuples(O)
   Add the newly found author,title pairs to R.

6. If R is large enough, terminate. Else, go to step 2.

There are three important operations above: FindOccurrencesA, FindOccurrencesB, and GenPatterns (Tuples is just a trivial project). FindOccurences A and B find occurrences of books and patterns respectively in the Web repository. These are fairly similar operations but can be challenging to compute efficiently, since there are tens

Isaac Asimov	The Robots of Dawn
David Brin[1]	Startide Rising
James Gleick	Chaos: Making a New Science
Charles Dickens	Great Expectations
William Shakespeare	The Comedy of Errors

Figure 1: Initial sample of books.

URL Pattern	Text Pattern
`www.sff.net/locus/c.*`	`<LI><B>`*title*`</B> by` *author* `(`
`dns.city-net.com/~lmann/awards/hugos/1984.html`	`<i>`*title*`</i> by` *author* `(`
`dolphin.upenn.edu/~dcummins/texts/sf-award.htm`	*author* `||` *title* `||` `(`

Figure 2: Patterns found in first iteration.

of thousands of books (or more) and hundreds of patterns. For the purposes of our experiment, the equivalent of a *grep* over the WebBase was used. Much of the experiment was limited to just portions of the WebBase because this operation is so time-consuming.

The third operation, GenPatterns, is the trickiest. The key is to avoid generating overly general patterns. We avoid such patterns by requiring that any patterns generated provide sufficiently long urlprefixes, prefixes, middles, and suffixes. We refer to [Bri] for the details of this process.

3.2 The Experiment

We started the experiment with just 5 books (see Figure 1). These produced 199 occurrences and generated 3 patterns (see Figure 2). Interestingly, only the first two of the five books produced the patterns because they were both science fiction books. A run of these patterns over matching URL's produced 4047 unique (author,title) pairs. They were mostly science fiction but there were some exceptions.

A search through roughly 5 million web pages found 3972 occurrences of these books. This number was something of a disappointment since it was not as large a blowup as had happened in the first iteration. However, it would have taken at least a couple of days to run over the entire repository so we did not attempt to generate more.

These occurrences produced 105 patterns, 24 of which had url prefixes that were not complete URLs. A pass over roughly 2 million URLs produced 9369 unique (author, title) pairs. Unfortunately, there were some bogus books among these. In particular, 242 of them were legitimate titles but had an author of “Conclusion”. We removed these from the list. This was the only manual intervention through the whole process. In future experiments, it would be interesting to see whether leaving these in would produce an unacceptable amount of junk.

For the final iteration, we chose to use the subset of the repository that contained the word “books.” This subset consisted of roughly 156,000 documents. Scanning for the 9127 remaining books produced 9938 occurrences. These in turn generated 346 patterns. Scanning over the same set of documents produced 15257 unique books with very little bogus data.

The quality of the results was surprisingly good. A random sample of the proposed books revealed that roughly 95% were legitimate books. The remainder were mostly articles and other media. Of these, roughly 25% were not found on Amazon (which incidentally is not a part of the WebBase since they are not accessible to robots) or several other major sites but were legitimate books with copyrights and publishers. Some were out of print, some were published electronically, and still others were missing from these online book stores for no apparent reason. Therefore, the union of hundreds of small sources of book citations of the Web is very useful even in the presence of the huge catalogs available online. For more details about this experiment see [Bri].

4 PageRank

4.1 Link Structure of the Web

While estimates vary, the current graph of the crawlable Web has roughly 300 million nodes (pages) and 3 billion edges (links). Every page has some number of forward links (outedges) and backlinks (inedges) (see Figure 3). We can never know whether we have found all the backlinks of a particular page, but if we have downloaded it, we know all of its forward links at that time.

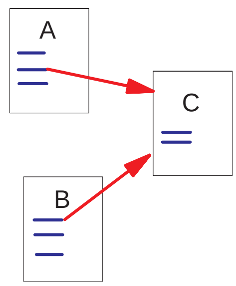

Figure 3: A and B are Backlinks of C

Web pages vary greatly in terms of the number of backlinks they have. For example, the Netscape home page has 62,804 backlinks in our current database while most pages have just a few backlinks. Generally, highly linked pages are more "important" than pages with few links. Simple citation counting has been used for many things including speculating on the future winners of the Nobel Prize [San95]. PageRank -[PBMW] provides a more sophisticated method for doing citation counting.

The reason that PageRank is interesting is that there are many cases where simple citation counting does not correspond to our common-sense notion of importance. For example, if a web page has a link from the Yahoo home page, it may be just one link but it is a very important one. This page should be ranked higher than other pages with more links but only from obscure places. PageRank is an attempt to see how good an approximation to "importance" can be obtained from just the link structure.

4.2 Propagation of Ranking Through Links

Based on the discussion above, we give the following intuitive description of PageRank: a page has high rank if the sum of the ranks of its backlinks is high. This covers both the case when a page has many backlinks and when a page has a few highly ranked backlinks.

4.3 Definition of PageRank

Let u be a web page. Then let F_u be the set of pages u points to and B_u be the set of pages that point to u. Let $N_u = |F_u|$ be the number of links from u and let c be a factor used for normalization (so that the total rank of all web pages is constant).

We begin by defining a simple ranking, R which is a slightly simplified version of PageRank:

$$R(u) = \frac{1}{c} \sum_{v \in B_u} \frac{R(v)}{N_v}$$

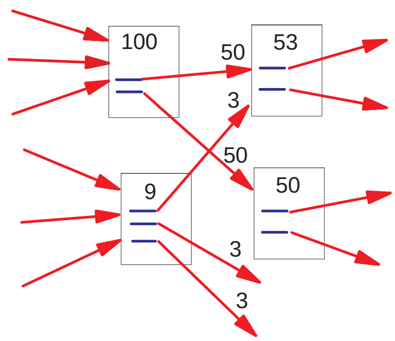

Figure 4: Simplified PageRank Calculation

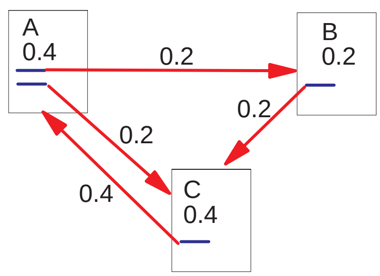

Figure 5: Steady State Solution to the PageRank Calculation

This equation formalizes the intuition in the previous section. Note that the rank of a page is divided among its forward links evenly to contribute to the ranks of the pages they point to. Note that $c < 1$ because there are a number of pages with no forward links and their weight is lost from the system. The equation is recursive but it may be computed by starting with any set of ranks and iterating the computation until it converges. Figure 4 demonstrates the propagation of rank from one pair of pages to another. Figure 5 shows a consistent steady state solution for a set of pages.

Stated another way, let A be a square matrix with the rows and columns corresponding to web pages. Let $A_{u,v} = 1/N_u$ if there is an edge from u to v and $A_{u,v} = 0$ if not. If we treat R as a vector over web pages, then we have $cR = AR$. So R is an eigenvector of A with eigenvalue c. In fact, we want the dominant eigenvector of A. It may be computed by repeatedly applying A to any nondegenerate initial rank vector.

There is a small problem with this simplified ranking function. Consider two web pages that point to each other but to no other page, and suppose there is some web page that points to one of them. This situation is shown in Figure 6. During iteration, this loop will accumulate rank but never distribute any rank (since there are no outedges). The loop forms a sort of trap which we call a rank sink.

To overcome this problem of rank sinks, we introduce a rank source:

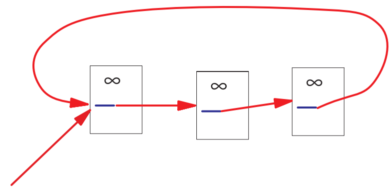

Figure 6: Loop that Acts as a Rank Sink

Definition 1: Let $E(u)$ be some vector over the Web pages that corresponds to a source of rank. Then, the PageRank of a set of Web pages is an assignment, R', to the Web pages which satisfies

$$R'(u) = \frac{1}{c}(\sum_{v \in B_u} \frac{R'(v)}{N_v} + E(u)) \tag{1}$$

such that c is maximized and $||R'||_1 = 1$ ($||R'||_1$ denotes the sum of the components of R').

$E(u)$ is some vector over the web pages that corresponds to a source of rank. Note that if E is all positive, c must be increased to balance the equation. Therefore, this technique corresponds to a decay factor. In matrix notation we have $cR' = AR' + E$. Since $||R'||_1 = 1$, we can rewrite this as $cR' = A + E \times \mathbf{1}R'$ where $\mathbf{1}$ is the vector consisting of all ones. So, R' is an eigenvector of $(A + E \times \mathbf{1})$.

4.4 Random Surfer Model

The definition of PageRank above has another intuitive basis in random walks on graphs. The simplified version corresponds to the standing probability distribution of a random walk on the graph of the Web. Intuitively, this can be thought of as modeling the behavior of a "random surfer". The "random surfer" simply keeps clicking on successive links at random. However, if a real Web surfer ever gets into a small loop of web pages, it is unlikely that the surfer will continue in the loop forever. Instead, the surfer will jump to some other page. The additional factor E can be viewed as a way of modeling this behavior: the surfer periodically "gets bored" and jumps to a random page chosen based on the distribution in E.

4.5 Personalization and Malicious Manipulation (Spam)

So far we have left E as a user defined parameter. In most tests we let E be uniform over all web pages with value α. However, E can be skewed to a particular user, say to weigh their bookmarks or homepage higher. We did some preliminary experiments with E set only to one particular Stanford professor's home page. This seemed to result in higher ranking for things relating loosely to that professor. Also, with this type of personalized E it is nearly impossible to mislead the ranking algorithm. There is no way to inflate the importance of a page, without convincing other pages to give you part of their importance. Even if I create a million web pages trying to mislead the system, these million pages will have the total importance only of the sum of the links into them. So if there is only one link from the outside web into my million pages, the million pages will have only the ranking from that one link to distribute amongst themselves. This type of resistance is very important in the commercial space, because there is a great deal of economic incentive to move pages to the top of popular search results.

4.6 Computing PageRank

The computation of PageRank is fairly straightforward if we ignore the issues of scale. Let S be almost any vector over Web pages (for example E). Then PageRank may be computed as follows:

$$\begin{array}{rcl} & R_0 & \leftarrow & S \\ \text{loop}: & & & \\ & R_{i+1} & \leftarrow & AR_i \\ & d & \leftarrow & ||R_i||_1 - ||R_{i+1}||_1 \\ & R_{i+1} & \leftarrow & R_{i+1} + dE \\ & \delta & \leftarrow & ||R_{i+1} - R_i||_1 \\ \text{while } \delta > \epsilon & & & \end{array}$$

Note that the d factor increases the rate of convergence and maintains $||R||_1$. An alternative normalization is to multiply R by the appropriate factor. The use of d may have a small impact on the influence of E.

We should also note that there are a large number of nodes that have no outgoing edges when we do not have a complete version of the web. Because of the presence of these nodes with unknown outedges, there is no completely satisfactory solution as to where to distribute their weight. If we do nothing, we have found that these nodes with no outgoing edges can cause the d factor to become larger than seems to work well. As a solution, we often remove nodes with no outgoing edges during the computation of PageRank, then add them back in after the weights have stabilized. This results in a slight boost to some nodes due to the change in normalization, but seems to be preferable to the alternatives.

5 The Google Search Engine

A major application of PageRank is searching. We have implemented two search engines which use PageRank. The first is a simple title-based search engine. The second is a full text search engine called Google [BPa] - [BP98]. Google utilizes a number of factors to rank search results, including standard IR measures, proximity, anchor text (text of links pointing to web pages), and PageRank. We encourage readers to try out Google at http://google.stanford.edu/ which allows searching the full text of 24 million web pages.

5.1 Use of PageRank

The benefits of PageRank are the greatest for underspecified queries. For example, on a conventional search engine a query for “Stanford University” may return any number of web pages which mention Stanford (such as publication lists. Using PageRank, the university home page is listed first.

An even more general query – “University” – is completely hopeless on conventional search engines, at best returning the home pages of random universities and at worst returning random pages at random universities. However, because of PageRank, the top ten results are the home pages of ten major universities – UIUC, Stanford, Michigan, and so forth.

5.2 Anchor Text

Let us consider the query “Stanford University” again. It so happens that there are nearly 6000 links to the Stanford home page, and for the overwhelming majority of them the text of the link reads “Stanford University”. When there are several thousand links that match the query exactly and point to the same Web page (like the

Stanford example), they should be a pretty good indication to a search engine that that page may be a good result to return for that query.

Even in cases when there are just one or several anchors pointing to a page that match a query, these anchors are very useful. First, anchors are often better descriptions of Web pages than the pages themselves. They frequently state precisely what is significant about the Web page. Second, they are often written by people other than the author of the Web page, so they are more resistant to malicious tampering to move a page to the top of the search results for commercial gain. In fact, Google distinguishes between on-site, off-site, and off-domain anchors to improve the resistance to malicious tampering. Finally, anchors allow a search engine to return a Web page even without crawling it. So even though the WebBase contains roughly 25 million Web pages, Google can return any of roughly 60 million URL's in search results, including images and email addresses.

5.3 Proximity

Another important component of Google is heavy reliance on word proximity. That is, if two query words occur close together in a document, then the document is weighted much more heavily than if they appear far apart. Use of proximity makes searching considerably more computationally expensive. However, in practice our unoptimized system can answer the vast majority of queries in a matter of seconds.

6 Other Experiments

The WebBase has also been used for several other experiments, which we mention only briefly here. These range from duplicate detection to data mining to queryless search.

The Web can be a very useful testbed for experimenting with systems that may be applied more broadly. SCAM is a project at Stanford to detect duplicated or nearly duplicated documents. The WebBase turns out to be a very good test set for that purpose, since it is a large corpus with a large amount of duplication with many variations in how texts are duplicated. The results of duplicate analysis ([SGM]) show that roughly 22% of the documents in the WebBase are exact duplicates, and even more have approximate duplicates.

Another application for which the Web makes a good test data set is market basket mining. We consider the Web pages to be the baskets and the words that appear on them are the items. This data set is particularly challenging for traditional algorithms. There are over 10 million distinct words of which tens of thousands still occur in significant frequency. A number of the words occur very often, and there are many extremely strong correlations. A recent project on dynamic data mining considers a market basket algorithm when it is not possible to exhaustively explore all possible rules [BPb].

A research direction for searching is what we call "background surfing" or "queryless search". Our contributions in Google thus far have made underspecified queries such as one or two word queries work well. However, now we consider the case of zero word queries. Our goal is to build a system that listens to conversation going on around it and produce relevant Web pages. We feel that such a system can have a very substantial impact on the way people work, because it makes them aware of things they did not even know to look for. A current very early prototype scans through email and retrieves relevant Web pages. The overall goal is a difficult task because it must include and combine high quality continuous speech recognition over a large vocabulary with high quality search where it is not feasible to display 10 results.

6.1 Summarization

An issue we wish to explore is the summarization of a collection of documents. We are currently developing the following process. Suppose that we wish to summarize the contents of the pages returned for a query to a web search engine. We fetch the pages whose URLs form the response to the query from a local repository, identify a family of important keywords (via proximity to search terms or using traditional IR techniques such as inter-document frequency), identify phrases containing these keywords, and finally output a set of phrases that occur most frequently across the documents (under some notion of approximate phrase equality). Our thesis is that this set of phrases will read like a summary of the web documents.

A critical component of this approach is the identification of interesting phrases in collections of documents. It is important to avoid generating phrases that are merely common language constructs or are sporadic juxtapositions of terms. We do not restrict ourselves to grammatical or English phrases; indeed, we would like to generate more general kinds of phrases including dates, email addresses, phone numbers, names, etc. It is our expectation, supported by some preliminary experimental results, that the resulting sets of phrases provide a good sense of the contents and the topic of a document collection. We have developed an approach for quickly identifying interesting phrases. For instance, we define a two-word phrase as a pair of words that have a strong affinity for each other and appear together in a particular order far more often than could be explained by language constructs or pure chance. Our technique scans text in linear time and produces a ranked list of phrases without prior knowledge of the content of a document or its relationship with the document collection. While performing experiments in phrase detection, we noticed that when our technique is applied to a collection of documents returned by a search engine on a query, the detected phrases are very descriptive of the quality of the pages returned by the search engine.

Another way to use this summarization is to visualize clusters of documents. The clusters may be formed out of the responses to a query or we may just have a clustering of some part of the web. In particular, suppose we have a hierarchical clustering of a collection of web pages. We can provide the user with a summary of the document sub-collection associated with any point in the hierarchy as well as the differences in the set of characterizing phrases between two adjacent points in the hierarchy. This summarization will enable the user to visualize the clustered documents while navigating or browsing the hierarchy.

7 Conclusion

A repository of Web pages such as the WebBase is an excellent research tool, enabling experiments that would otherwise be impossible to perform efficiently. And, of course, it can be crucial to the development of better search engines.

An important lesson we have learned from these experiments is that *size does matter*. The extraction experiment would likely have failed if the WebBase had been one third of its current size.

Furthermore, the hardware cost of a large WebBase is quite reasonable and trends in disk capacity and computing power make it very likely that many more applications involving a local Web repository will become practical in the near future.

In analyzing the Web, we have found that it is important to look beyond just the text. The extraction experiment made heavy use of formatting and URL's. PageRank takes advantage of the link structure. Google makes use of anchor text and font information. Much of the information on the Web is not in the plain text and many applications can achieve great gains by leveraging it.

8 Acknowledgements

Scott Hassan and Alan Steremberg have been critical to the development of Google. Their talented contributions are irreplaceable, and the authors owe them much gratitude. Finally we would like to recognize the generous support of our equipment donors IBM, Intel, and Sun and our funders.

References

[BB98] Krishna Bharat and Andrei Broder. A technique for measuring the relative size and overlap of public web search engines. In *Proceedings of the Seventh International World Wide Web Conference*, 1998.

[BPa] Sergey Brin and Larry Page. Google search engine. `http://google.stanford.edu`.

[BPb] Sergey Brin and Lawrence Page. Dynamic data mining: Exploring large rule spaces by sampling. `http://www-db.stanford.edu/~sergey/ddm.ps`.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International World Wide Web Conference*, 1998.

[Bri] Sergey Brin. Extracting patterns and relations from the world wide web. In *Proceedings of the WebDB Workshop at EDBT 1998*. `http://www-db.stanford.edu/~sergey/extract.ps`.

[McB94] Oliver A. McBryan. GENVL and WWWW: Tools for Taming the Web. In O. Nierstarsz, editor, *Proceedings of the first International World Wide Web Conference*, page 15, CERN, Geneva, May 1994.

[MOS97] Workshop on management of semistructured data. `http://www.research.att.com/~suciu/workshop-papers.html`, May 1997.

[PBMW] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. `http://google.stanford.edu/google\_papers.html`.

[San95] Neeraja Sankaran. Speculation in the biomedical community abounds over likely candidates for nobel. *The Scientist*, 9(19), Oct 1995.

[SGM] Narayanan Shivakumar and Hector Garcia-Molina. Finding near-replicas of documents on the web. In *Proceedings of the WebDB Workshop at EDBT 1998*.

[Sul] Danny Sullivan. Search engine watch. `http://www.searchenginewatch.com`.

[Tsi] Tsimmis home page. `http://www-db.stanford.edu/tsimmis/tsimmis.html`.