International Journal of Information Technology and Management

Vol. VI, Issue No. II, May-2014, ISSN 2249-4510

AN ANALYSIS ON THE ANATOMY OF A LARGE SCALE HYPERTEXTUAL WEB SEARCH ENGINE AND A WEB CRAWLER APPLICATION

AN INTERNATIONALLY INDEXED PEER REVIEWED & REFEREED JOURNAL

www.ignited.in

International Journal of Information Technology and Management
Vol. VI, Issue No. II, May-2014, ISSN 2249-4510

An Analysis on the Anatomy of a Large Scale Hypertextual Web Search Engine and a Web Crawler Application

Ms. Rubina Khan

Asst. Prof. Immersive Institute of Technology

Abstract – In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/ To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and web proliferation, creating a web search engine today is very different from three years ago. This paper provides an in-depth description of our large-scale web search engine – the first such detailed public description we know of to date. Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results. This paper addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext. Also we look at the problem of how to effectively deal with uncontrolled hypertext collections where anyone can publish anything they want.

To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of Web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the Web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and Web proliferation, creating a Web search engine today is very different from three years ago. This paper provides an in-depth description of our large-scale Web search engine — the first such detailed public description we know of to date.

Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results.

----------♦----------

INTRODUCTION

The ability to browse is generally regarded as one of the most important reasons for using hypertext, while searching facilities should also be supported in modern hypertext environments. The World Wide Web is such a hypertext environment. Because of its huge scale and arbitrary structure, it creates many challenges for the development of its searching capabilities.

In the current Web, most links are not typed, and there is no link-based composition mechanism. Thus the Web lacks explicit structural meta information, and the search engines on it are typically keyword-based. With such engines, people usually get a large amount of pages that they cannot process, or even more, many of the pages are totally irrelevant to their information needs, especially when they search for information on specific topics.

To improve the ability of expressing structures and semantics on the Web, several new standards, mainly XML (Extensible Markup Language) and RDF (Resource Description Framework), are developed or under development. These standards open new opportunities to improve the information access on the Web. However, it is an open question how to make use of the structural and semantic information that can be represented with the standards efficiently for search purposes. This paper describes a part of our effort to answer this question.

In this work, we focus on making use of hypertext contexts, one of the main highlevel hypermedia structures that can be represented with the new Web

standards for searching on the Web. We view a hypertext context as a mechanism to specify the scope of the information space to be examined in a search and argue that this would help to improve the search response time and the quality of the results of searches concerning a specific topic or subject domain.

The web creates new challenges for information retrieval. The amount of information on the web is growing rapidly, as well as the number of new users inexperienced in the art of web research. People are likely to surf the web using its link graph, often starting with high quality human maintained indices such as Yahoo! or with search engines. Human maintained lists cover popular topics effectively but are subjective, expensive to build and maintain, slow to improve, and cannot cover all esoteric topics.

Automated search engines that rely on keyword matching usually return too many low quality matches. To make matters worse, some advertisers attempt to gain people's attention by taking measures meant to mislead automated search engines. We have built a large-scale search engine which addresses many of the problems of existing systems. It makes especially heavy use of the additional structure present in hypertext to provide much higher quality search results. We chose our system name, Google, because it is a common spelling of googol, or 10100 and fits well with our goal of building very large-scale search.

Engineering a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been conducted on them.

Furthermore, due to rapid advance in technology and web proliferation, creating a web search engine today is very different from three years ago. A search engine finds information for its database by accepting listings sent in by authors who want exposure, or by getting the information from their "web crawlers," "spiders," or "robots," programs that roam the Internet storing links to and information about each page they visit. A web crawler is a program that downloads and stores Web pages, often for a Web search engine. Roughly, a crawler starts off by placing an initial set of URLs, *S*0, in a queue, where all URLs to be retrieved are kept and prioritized. From this queue, the crawler gets a URL (in some order), downloads the page, extracts any URLs in the downloaded page, and puts the new URLs in the queue. This process is repeated until the crawler decides to stop. Collected pages are later used for other applications, such as a Web search engine or a Web cache.

The most important measure for a search engine is the search performance, quality of the results and ability to crawl, and index the web efficiently. The primary goal is to provide high quality search results over a rapidly growing World Wide Web. Some of the efficient and recommended search engines are Google, Yahoo and Teoma, which share some common features and are standardized to some extent.

THE WORLD BEFORE GOOGLE

With the birth of the World Wide Web (WWW), the usage of the Internet has grown dramatically. One of the _rst search engines, the World Wide Web Worm, had an index of around 110,000 web pages and it received about 1500 queries per day by 1994 - only 4 years after WWW's creation. By November 1997, Altavista, one of the more popular web search engines during that time, claimed to handle roughly 20 million queries per day and indexed over tens of millions of web pages. (As of January 2005, the number of web pages grew to over 11.5 billion and Google claims to process nearly 90 million queries per day.)

Although the number of the users accessing the web has been growing rapidly, the way they access it remained pretty much the same. People tend to surf the web using a graph of links, often starting with search engines or popular web pages like Yahoo.com and continue navigating using their lists of topics or other links.

The automated searches used plain keyword matching or other simple techniques which returned too many low quality results or were very easy to manipulate. For example, some search engines used only the font size of the text to determine the importance of the search result. Hence, advertisers exploited this weakness by creating extra-large redundant text messages so that their web page would always appear in the front of search result. For example to gain attention of possible computer buyers, they would insert every possible word connected to computers in a large font like "Dell", "Vaio", "Acer", etc to their web page. In this case if a person searches for Sony Vaio laptops, for instance, he will see the advertiser's web site in the front of the search result instead of Sony Vaio's official web site or other more prominent shopping web sites. Some advertisers would simply pay money to the search engines, so that their web sites could appear on the top of search results.

The human maintained lists, on the other hand, have an advantage of efficiently listing the popular topics but this approach is expensive to build and maintain, slow and difficult to improve and in many cases it is not objective, as a person has to decide which topics to include in the list. These human maintained lists are also not able to scale with the growth of the web, as a person is limited in his/her ability to look to the enormous amount of documents.

GOOGLE: SCALING WITH THE WEB

Creating a search engine which scales even to today's web presents many challenges. Fast crawling technology is needed to gather the web documents and keep them up to date. Storage space must be used efficiently to store indices and, optionally, the documents themselves. The indexing system must process hundreds of gigabytes of data efficiently. Queries must be handled quickly, at a rate of hundreds to thousands per second.

These tasks are becoming increasingly difficult as the Web grows. However, hardware performance and cost have improved dramatically to partially offset the difficulty. There are, however, several notable exceptions to this progress such as disk seek time and operating system robustness. In designing Google, we have considered both the rate of growth of the Web and technological changes. Google is designed to scale well to extremely large data sets. It makes efficient use of storage space to store the index. Its data structures are optimized for fast and efficient access (see section 4.2). Further, we expect that the cost to index and store text or HTML will eventually decline relative to the amount that will be available. This will result in favorable scaling properties for centralized systems like Google.

WEB CRAWLER WORKING

Web crawlers are an essential component to search engines; running a web crawler is a challenging task. There are tricky performance and reliability issues and even more importantly, there are social issues. Crawling is the most fragile application since it involves interacting with hundreds of thousands of web servers and various name servers, which are all beyond the control of the system. Web crawling speed is governed not only by the speed of one's own Internet connection, but also by the speed of the sites that are to be crawled. Especially if one is a crawling site from multiple servers, the total crawling time can be significantly reduced, if many downloads are done in parallel.

Despite the numerous applications for Web crawlers, at the core they are all fundamentally the same. Following is the process by which Web crawlers work:

1. Download the Web page.

2. Parse through the downloaded page and retrieve all the links.

3. For each link retrieved, repeat the process.

The Web crawler can be used for crawling through a whole site on the Inter-/Intranet. You specify a start-URL and the Crawler follows all links found in that HTML page. This usually leads to more links, which will be followed again, and so on. A site can be seen as a tree-structure, the root is the start-URL; all links in that root-HTML-page are direct sons of the root. Subsequent links are then sons of the previous sons.

A single URL Server serves lists of URLs to a number of crawlers. Web crawler starts by parsing a specified web page, noting any hypertext links on that page that point to other web pages. They then parse those pages for new links, and so on, recursively. Webcrawler software doesn't actually move around to different computers on the Internet, as viruses or intelligent agents do. Each crawler keeps roughly 300 connections open at once. This is necessary to retrieve web pages at a fast enough pace. A crawler resides on a single machine. The crawler simply sends HTTP requests for documents to other machines on the Internet, just as a web browser does when the user clicks on links. All the crawler really does is to automate the process of following links.

Web crawling can be regarded as processing items in a queue. When the crawler visits a web page, it extracts links to other web pages. So the crawler puts these URLs at the end of a queue, and continues crawling to a URL that it removes from the front of the queue.

GOOGLE'S NEW THOUGHTS

Use of Proximity - Google uses several features to assure high quality search results. For example it uses location information for all hits to make extensive use of proximity in search and it uses some visual presentation details like font size, as words in a larger or bolder font tend to be more important and hence are weighted higher than other words. The proximity of the search keywords is also important because for instance when a person searches for \computer science", he/she is not interested in web sites about "computers" or "sciences" or any other person's personal web page containing a sentence like "Yesterday my computer broke down... I watched a science _ction movie".

Page Rank - A page with many inlinks can be considered to be important because many people point to this page. Counting only inlinks, however, can be easily manipulated. For example one can create several small web pages and make a mesh of links, connecting each to one another. So it is important to distinguish the characteristic of the linking web page. For example, a link from Yahoo.com is not the same as a link from an average Joe's home page.

As you can see this notion is similar to academic papers' citations: The more citations a paper has or the more distinguished authors' papers cite it, the more importance or quality the paper probably has.

Google uses this idea of counting inlinks but with the extension of (a) not counting links from all pages equally, and by (b) normalizing by the number of links on a page, to determine the importance of the web page, which is called PageRank.

Page rank can also be seen as a model of user behavior. The probability that a random user, given a random web page as a starting point, and by either clicking on links or by restarting at another random web page, reaches that certain web page, de_nes the "Page rank" of that page. So clearly, if many web sites point to that web page, the probability that it will get hit will be higher. Also if the page is pointed from a well-known/popular web site it will also have a higher probability to get hit.

Anchor Text - The designers of Google paid special attention to anchor texts, i.e. the texts of the links. It is possible to get more accurate results because anchor text often have better descriptions of the web pages that they point to. For example, someone would never make a link to michaeljordon.com with an anchor text sumo wrestler or Chevrolet. They are more likely to name the link "air jordan", "#23", or "the greatest basketball player ever", etc.

In addition, anchor texts provide an opportunity to find results which cannot be found by plain text search, like media files, programs, etc. To continue our previous example, if someone posted a video clip of Michael Jordan's best dunk shots, the only way we can find this file is if the link to the file had an anchor text like "MJ's dunk shots" etc.

This idea of propagating anchor text to the page it refers to was first implemented in the World Wide Web Worm. They used it to help search non-text documents and expand the search coverage. Google extended their idea to get better quality results.

GOOGLE ARCHITECTURE

In this section, we will give a high level overview of how the whole system works. Further sections will discuss the applications and data structures not mentioned in this section. Most of Google is implemented in C or C++ for efficiency and can run in either Solaris or Linux.

In Google, the Web crawling (downloading of Web pages) is done by several distributed crawlers. There is a URLserver that sends lists of URLs to be fetched to the crawlers. The Web pages that are fetched are then sent to the storeserver. The store- server then compresses and stores the Web pages into a repository. Every Web page has an associated ID number called a doclD which is assigned whenever a new URL is parsed out of a Web page. The indexing function is performed by the indexer and the sorter. The indexer performs a number of functions. It reads the repository, uncompresses the documents, and parses them. Each document is converted into a set of word occurrences called hits. The hits record the word, position in document, an approximation of font size, and capitalization. The indexer distributes these hits into a set of "barrels", creating a partially sorted forward index. The indexer performs another important function. It parses out all the links in every Web page and stores important information about them in an anchors tile. This file contains enough information to determine where each link points from and to. and the text of the link.

The URL resolvcr reads the anchors tile and converts relative URLs into absolute URLs and in turn into doclDs. It puts the anchor text into the forward index, associated with the doclD that the anchor points to. It also generates a database of links which are pairs of doc IDs. The links database is used to compute PageRanks for all the documents.

CRAWLING TECHNIQUES

Focused Crawling - A general purpose Web crawler gathers as many pages as it can from a particular set of URL's. Whereas a focused crawler is designed to only gather documents on a specific topic, thus reducing the amount of network traffic and downloads. The goal of the

focused crawler is to selectively seek out pages that are relevant to a pre-defined set of *topics*. The topics are specified not using keywords, but using exemplary documents.

Rather than collecting and indexing all accessible web documents to be able to answer all possible ad-hoc queries, a focused crawler analyzes its crawl boundary to find the links that are likely to be most relevant for the crawl, and avoids irrelevant regions of the web.

This leads to significant savings in hardware and network resources, and helps keep the crawl more up-to-date. The focused crawler has three main components: a classifier, which makes relevance judgments on pages crawled to decide on link expansion, a distiller which determines a measure of centrality of crawled pages to determine visit priorities, and a crawler with dynamically reconfigurable priority controls which is governed by the classifier and distiller.

The most crucial evaluation of focused crawling is to measure the harvest ratio, which is rate at which relevant pages are acquired and irrelevant pages are effectively filtered off from the crawl. This harvest ratio must be high, otherwise the focused crawler would spend a lot of time merely eliminating irrelevant pages, and it may be better to use an ordinary crawler instead (Baldi, 2003).

Distributed Crawling - Indexing the web is a challenge due to its growing and dynamic nature. As the size of the Web is growing it has become

imperative to parallelize the crawling process in order to finish downloading the pages in a reasonable amount of time. A single crawling process even if multithreading is used will be insufficient for large – scale engines that need to fetch large amounts of data rapidly. When a single centralized crawler is used all the fetched data passes through a single physical link. Distributing the crawling activity via multiple processes can help build a scalable, easily configurable system, which is fault tolerant system. Splitting the load decreases hardware requirements and at the same time increases the overall download speed and reliability. Each task is performed in a fully distributed fashion, that is, no central coordinator exists (Baldi, 2003).

CONCLUSION

Google is designed to be a scalable search engine. The primary goal is to provide high quality search results over a rapidly growing World Wide Web. Google employs a number of techniques to improve search quality including page rank, anchor text, and proximity information. Furthermore, Google is a complete architecture for gathering web pages, indexing them, and performing search queries over them.

A large-scale web search engine is a complex system and much remains to be done. Our immediate goals are to improve search efficiency and to scale to approximately 100 million web pages. Some simple improvements to efficiency include query caching, smart disk allocation, and subindices. Another area which requires much research is updates. We must have smart algorithms to decide what old web pages should be recrawled and what new ones should be crawled. Work toward this goal has been done in [Cho 98]. One promising area of research is using proxy caches to build search databases, since they are demand driven. We are planning to add simple features supported by commercial search engines like boolean operators, negation, and stemming. However, other features are just starting to be explored such as relevance feedback and clustering (Google currently supports a simple hostname based clustering). We also plan to support user context (like the user's location), and result summarization. We are also working to extend the use of link structure and link text. Simple experiments indicate PageRank can be personalized by increasing the weight of a user's home page or bookmarks. As for link text, we are experimenting with using text surrounding links in addition to the link text itself. A Web search engine is a very rich environment for research ideas.

REFERENCES

- Brian Pinkerton, *Finding What People Want: Experiences with the WebCrawler.* The Second International WWW Conference Chicago, USA, October 17-20, 1994.
- Casanova, M. A., Tucherman, L., "The nested context model for hyperdocuments," *Proc. of Hypertext'91*, pp. 193-201.
- Chakrabarti, Soumen. Mining the Web: Analysis of Hypertext and Semi Structured Data, 2003
- Franklin, Curt. How Internet Search Engines Work, 2002.
- Grossan, B. "Search Engines: What they are, how they work, and practical suggestions for getting the most out of them," February1997.
- Huck, G., Fankhauser, P., Aberer, K., Neuhold, E., "Jedi: Extracting and synthesizing information from the Web," *Proc. 3rd IFCIS International Conference on Cooperative Information Systems (CoopIS'98)*, New York City, August 1998, pp. 32-43.
- Junghoo Cho, Hector Garcia-Molina, Lawrence Page. *Efficient Crawling Through URL Ordering.* Seventh International Web Conference (WWW 98). Brisbane, Australia, April 14-18, 1998.
- Luis Gravano, Hector Garcia-Molina, and A. Tomasic. *The Effectiveness of GlOSS for the Text-Database Discovery Problem.* Proc. of the 1994 ACM SIGMOD International Conference On Management Of Data, 1994.
- Mauldin, Michael L. Lycos Design Choices in an Internet Search Service, IEEE Expert Interview http://www.computer.org/pubs/expert/1997/trends/x1008/mauldin.htm
- Oliver A. McBryan. GENVL and *WWWW: Tools for Taming the Web. First International Conference on the World Wide Web.* CERN, Geneva (Switzerland), May 25-26-27 1994.
- Schwartz, M., Delisle, N., "Contexts – A partitioning concept for hypertext," *ACM Transactions on Office Information Systems*, 5(2), April 1987, pp. 168-186.
- Trigg, R. H., "Hypermedia as integration: Recollections, reflections and exhortations," Keynote Address in *Hypertext'96 Conference.* Xerox Palo Alto Research Center.