Mining and Knowledge Discovery from the Web

Kevin S. McCurley and Andrew Tomkins
IBM Almaden Research Center

Abstract

The World Wide Web presents an interesting opportunity for data mining and knowledge discovery, and this area is growing rapidly as both a research topic and a business activity. In this survey we describe some of the problems that are addressed, and elements of the WebFountain infrastructure that we have built for addressing them. Our focus here is on describing some of the lessons learned and some broad research areas that are involved.

1 Introduction

The World Wide Web has had an impact on nearly every aspect of society including commerce, science, politics and government, personal relationships, and health. In only one short decade it has evolved to become a global information space for almost every kind of information. As such, it presents an intriguing object of study. A group of us at IBM Almaden have been working since 1999 on an infrastructure that is designed to crawl a substantial portion of the web and extract information from it. Our original motivation was to build a research tool to assist us in understanding the underlying mathematical structure that was evolving in the web, but it has also evolved into a platform to serve data mining needs of customers. In this paper we shall describe the goals and architecture of the WebFountain project (`http://www.almaden.ibm.com/webfountain`), as well as outline some problems that have arisen from our attempts to understand the structure of information on the web.

There are several projects worldwide that have focused on information retrieval and data mining of a substantial fraction of the entire web, and some of them have grown into businesses with high visibility. Some notable examples are search engines such as Altavista and Google.[1] Each of these projects uses a distributed "crawler" program to retrieve documents from the web; storing them into a federated database for later processing and extracting links to new URLs as it progresses. In the case of a search engine, the major pre-processing consists of building an inverted keyword index that maps terms to document IDs, which reduces to a large sorting operation. Once this is done, a query interface is built to provide a user interface. The importance of search as an application cannot be underestimated, and in fact it is believed that a majority of web user sessions now begin by first consulting a search engine for an informational need.

The WebFountain system includes a search engine, but adds other features of hypertext analysis and knowledge discovery. The activity of knowledge discovery seeks to uncover potentially useful and understandable patterns in data, and in our case the underlying data is the structure and information present on the Web. This is integrated into business processes through a cyclic process. First we identify opportunities where data can provide value, after which we apply data mining to gain knowledge about the opportunity, apply this knowledge, and verify our results. At this point new insight may be gained to feed back into the cycle. This methodology has been applied with customer data for customer relationship management and predictive marketing, but when applied to web information the opportunities are quite broad.

For example, one of our activities has been semantic tagging of web data. In recent years there has been an emerging trend to create a machine-readable *semantic web* that augments the human-readable World Wide Web [2] and creates an infrastructure for higher-level functionality. In [7] we describe an application of the WebFountain infrastructure that generated semantic tagging information for over 434 million web pages, including the identification of *named entities* such as people, place names, product names, etc. Another example of semantic tagging applications was described in [16], in which geographic tagging of web pages and web sites was extracted from pages, and a navigational interface was provided to allow the user to find pages that were geographically similar to any given page. These represent example applications that the WebFountain project is designed to support.

One can view the semantic tagging process as creating a semi-structured database from the raw hypertext information on the web. Once we have built this database, there are numerous opportunities for discovery of associ-

[1] Altavista and Google are registered trademarks.

ation rules [1], clustering and classification of documents and/or entities, deviation detection, and application to business processes. In this case the semantic tags become the primary data source for data mining, but the original web information is still crucial for verification and inference.

The web has many substructures to it, including the high level knowledge that it represents, the linguistic structure of text, the character representation of the text, the visual structure of layout, and the hierarchical structure of DNS, file systems, and DOM [11]. Perhaps the most interesting one is the link structure that turns it into hypertext. This link structure contains a great deal of information about the relationships between the information entities that reside at URLs, and will be discussed further in section 6.

2 How big is the web?

In the early days of the web, this question was of great interest because it was clear that the web was growing so rapidly. Over time the web has not only grown, but its very nature has also changed. In the early days of the web it was natural to think of URLs as representing document identifiers, or perhaps mini-documents in a web of hypertext. One major theme that has emerged is that URLs represent true *resources* rather than documents, and the content that is fetched from a given URL is often expected to change over time (sometimes very rapidly!). Another trend that has emerged is the use of HTTP as a means to access databases (e.g., reviews on products, inventory, discussion lists, etc.). This is often indicated by the presence of a '?' character in the URL, which arises from a standard for encoding arguments into the HTTP protocol.

Given these changes, it is clear that the Web does not represent a static set of documents to count, but rather a rapidly evolving dynamic information space. Thus the question of "how big is the web" has several aspects to it:

- is there any practical way of counting the number of legitimate URLs that represent content? In theory the web has no upper bound on its size, and the only practical bound is that currently imposed by the most popular browser, which limits URLs to only 2083 characters.

- the URL `http://www.nytimes.com/` represents content that changes many times each day, in spite of the fact that it does not immediately appear to be a dynamically generated resource. How many resources are there of this type, and how rapidly do they change?

- as time passes, major new portions of the web appear, but others disappear. How can we characterize the rate at which information is removed from the web? Should it be archived, and if so - how?

- large segments of the web are protected off from retrieval by automated programs, either by the convention of a `robots.txt` file or by requirements for user authentication. Is there any way to estimate the amount of content that is viewable only by humans?

- the same content often can be found at multiple URLs, and sometimes content appears at many URLs with only subtle changes (e.g., new color scheme, new date, or different advertisements). Is there a reasonable methodology to characterize and estimate the duplication that appears on the web?

As a lower bound, Google now reports that they include over 4.2 billion URLs in their web index, though it is not clear how many of these have been crawled since it is possible (and indeed, advantageous [10]) to index documents without crawling them. In particular, the precision of search is often improved by using the "anchor text" that points to them, namely the highlighted text underlying the hypertext link to a document. As of this writing, the crawler at IBM Almaden has discovered links to approximately six billion URLs, and has fetched a large fraction of these, some of them many times. The Internet Archive (`www.archive.org`) reportedly holds approximately 30 billion web pages that they have collected over the life of the World Wide Web from various sources.

3 Crawling the Web

While it is tempting to go after as much of the web as possible, there is a law of diminishing returns from doing so. In particular, there is a tension between the desire to have as much coverage as possible vs. the desire to have only good quality content. The definition of what constitutes "quality" is of course dependent on the application the content is used for, and this greatly complicates the construction of a general purpose architecture for web data mining. Moreover, the widespread appeal of the web means that it represents the full diversity of social forces in the world, and what constitutes "quality" to one person may not be in agreement with that of another person or culture. As an example, there appears to be a huge amount of pornographic material on the web, and for some applications this is problematic. Another example is provided by the direct manipulation of search engine rankings through the construction of hypertext with specific patterns.

There is another tradeoff to be made between trying to cover as much content as possible and trying to maintain as fresh a copy as possible of any high quality dynamic content. Thus decisions must be made as to how to use the available processing and bandwidth in order to maintain a collection that balances quality, freshness, and coverage.

Ignoring data types other than HTML, the average size of a document in our crawl is approximately 12,500 bytes. Making reasonable assumptions about how much data can be pulled through a 100 Mbit connection to the Internet, it is reasonable to assume that we could download approximately 800 URLs per second. This is accomplished by using a cluster of machines to manage the crawling. The crawler application is run as a parallel application, with responsibility for crawling a host being assigned to one of the cluster machines, and all hyperlinks to URLs on that site being reported to that machine. The division of work by hostname is dictated by the need to balance several requirements, including:

politeness when fetching documents from the web, there is an accepted policy that automated crawlers should not consume the resources of a particular too heavily, by placing a delay between consecutive pages fetched.

DNS Before we can fetch a page from a site, we must resolve the DNS for the hostname. Since DNS is itself an unreliable distributed protocol, the latency for resolving hostnames must be masked from the fetching process. We maintain our own database for caching DNS information, as it is a rich source of information about the web.

`robots.txt` by convention, each site can declare that all or part of a site is off limits to automated crawlers. As URLs are discovered for a site, they must be checked against the current version of the server's `robots.txt` file to make sure that they can be retrieved.

Because the crawler is our interface to the outside world, running a crawler requires some attention to problems that can crop up. In particular, the politeness and robots.txt requirements are a constant source of confusion for the people who maintain web servers, and it often requires explanation or problem resolution. Our crawler is described in more detail in in [8].

3.1 Storage Requirements

In order to store the uncompressed contents of two billion HTML URLs, it takes about 25 terabytes of raw disk space. Estimates on the cost of disk storage vary widely, depending on the characteristics of the technology (e.g., reliability, bandwidth, latency, packaging). One thing is clear however: this cost has fallen rapidly over the last five years, and the trend is even more pronounced than Moore's law. It is currently relatively straightforward to build a system that can store a copy of the text on the web for under US$100,000, and this number has not changed substantially over the last five years (the web grows but storage gets cheaper).

By supercomputing standards this does not represent a very large data set any more, as the National Center for Atmospheric Research reported in 2003 that they stored over a petabyte of data for weather modeling, and NASA's Terra satellite reportedly generates 194 gigabytes of new raw data each day. Moreover, the web is certainly not the only large source of text that we could imagine storing and processing. By contrast, the largest comparable repository of textual material resides in the United States Library of Congress, which holds approximately 29 million books, with perhaps 10 billion pages (a printed page almost certainly contains more text than the average web page). The digitization of textual information that is currently stored on paper presents an intriguing area for the future of text analytics, and a number of projects have been undertaken in recent years to produce digital libraries with full search capabilities (e.g.,[17]) using only paper as the original data source. The typical approach for this is to scan the printed page, producing an image, and then process this image using optical character recognition to produce an approximate textual representation. A rule of thumb is that a scanned page requires about 50K of data for reasonable OCR quality, so the entire Library of Congress can be stored in a system that requires perhaps 500 terabytes, including both images and text (in this case the images are already compressed).

In passing we note that email represents another potentially large collection of text for data mining purposes (though the privacy implications are chilling, to say the least). It has also been estimated that the world sends approximately 20 billion email messages per day in 2003, which translates into perhaps 200 terabytes per day of email. Unfortunately in the case of email, a majority of it is currently unsolicited commercial email (spam), and the major problem faced at present is trying to classify the content. In the future we might expect some organizations to increase their efforts to apply knowledge discovery on email.

4 Mining the Web

While the data set is not all that large, the Web still presents some challenges for a storage and processing architecture. One of the crucial lessons in parallel applications is that data locality is a primary determining factor for performance and scalability. Most scientific applications use data that is inherently three-dimensional, and most scientific applications and simulations lend themselves to partitioning data so that communication from any given processor only involves a other few processors. The inherent low dimensionality of many problems is a fundamental reason why so many scientific applications have been amenable to parallel processing. By contrast, the web is often characterized as a high dimensional data set, since every distinct word potentially represents a dimension. If you look at all the data

elements (documents) that are related to a given document based on textual similarity, it quickly encompasses a major portion of the Web unless you are careful to constrain the problem. From another perspective, the "small world" nature of the link graph (see [18]) suggests that the information on the web is very closely intertwined.

In designing a distributed system to support large scale hypertext analytics, one of our first decisions is how to lay out the data. Consider for example the problem of constructing an inverted keyword index on the text corpus. Ignoring for a moment the problems of stemming, synonomy and polysemy, the problem of building an index comes down to tokenization and sorting the list of (docid,termid) pairs by termid. These sorted lists are called "postings lists", and they typically make up a data set that is itself 30% the size of the original corpus, though this figure varies according to strategy. Once an index has been built, we can perform keyword queries on the data by retrieving the postings list for the query terms, and performing boolean and other operations on them.

The decision of how to partition the data to best support text indexing is a non-trivial one [19]. One option is to partition the index by document, in which case the building of an index on the documents is trivially parallelizable. Query processing then becomes a highly parallel process however, as we have to send the query to every machine in order to discover all documents that satisfy the query. By contrast, we could decide to partition the postings lists by term, in which case we need only consult the machines that hold postings lists for terms contained in our query. In this case there are opportunities for parallelizing the query processing as batches of queries arrive with requests for different terms. Each of these approaches has certain disadvantages for reliability, performance, and scalability, and the choice of an architecture depends on the characterization of the likely queries. In the case of a public search engine like Google, the queries tend to be extremely short, with a heavy tail distribution for query term frequency but very bursty traffic in response to current events. Moreover, the customer expectation is for nearly instantaneous response. In our business model, we expect queries to be extremely complex, but the customer may be more patient in waiting for their results if it is part of a methodical business cycle. In our system we chose to partition the index by document, which means that the query runtime system has to involve all machines with postings lists.

4.1 Semantic Tagging

In some cases it makes relatively little difference how we partition the documents, because processing proceeds in a trivially parallelizable fashion, processing one document at a time. In the WebFountain project we employ a large number of programs called "miners" that extract metadata about the document and store the results with the document. Some example miners that have been built include:

encoding miner this miner uses various clues to deduce the encoding used for the text of the page,

Language miner this takes the raw content of the page along with the encoding from the encoding miner and produces a guess of the dominant human language used in the page.

UTF-8 based on the encoding used, this miner will produce a canonicalized UTF-8 encoding of the page.

porn miner a substantial fraction of the web is pornographic in nature, and this program determines with high probability whether a page should be considered pornographic.

detagger this miner takes the UTF-8 content of the page and produces a detagged version in which all markup and punctuation is removed, leaving only the text.

phone number miner this miner examines the UTF-8 content and recognizes phone numbers.

link miner this miner extracts hyperlinks and their associated metadata, including the tag type, the offset within the page, and the anchor text associated with the link.

name miner this miner recognizes the names of people in the content, and produces a canonicalized version of the name (e.g., George Bush vs. G. W. Bush). The data from this is used in [12] for experiments on social networks.

Each of these miners reads some input from the data store and produces some metadata that is written back to the store along with the page. All of the data for the page is stored in an XML-like structure [6]. In order to reduce the amount of disk I/O, these miners are run in a sequential chain on each page, lifting the necessary inputs for a page from disk, running the relevant pieces through each miner, producing tags along the way, and finally rewriting the record for the page, but now with the newly added tags.

5 Global Analysis and aggregation

Per-page mining requires very little coordination between the machines in the cluster, but some processing requires global coordination. In fact we have already seen two examples of such global processing, namely the crawling (which reports URLs to the relevant node that is responsible for the host of the URL) and the text indexer. At this

time it is appropriate to mention that in addition to indexing the terms in the document, we also index the tags that are associated with the documents. This makes it possible to construct queries such as "find all pages that are written in German, have a link to a page on `ibm.com`, mention a politician, and contain the word *schnell*."

Another problem that requires global computations is that of duplicate detection (or near duplicate detection). For this purpose we employ a technique in which hashes over moving windows of the page are computed [4], and these "shingles" are gathered together to an off-cluster location and processed to provide global information on exact and approximate duplicate pages.

Another form of processing that we perform outside the main store cluster is global ranking of documents [9]. This calculation entails assembling all of the links from all of the pages in order to compute the principal eigenvector of the incidence matrix for the hyperlink graph. For this purpose we run a parser to extract links, and we assemble them on a single machine, where we sort them by destination. We then prune the set of links to save only links that point the pages we have already crawled, saving the rest of the links to the uncrawled frontier for post-processing. This results in a directed graph that contains approximately 20 billion edges. In order to facilitate processing of this graph, we convert the URLs to sequential integer IDs, and construct a data representation for the graph that uses only the integer IDs. The actual calculation of global ranking uses an iterative linear algebra process to compute a principal eigenvector of the incidence matrix for the graph, and for this we do not need to hold the entire graph in memory. Instead, we scan through the edges sequentially, performing a matrix-vector multiplication as we go. The processing for this turns out to be formidable but doable on a single machine [5].

There are numerous other global tasks that one might imagine performing on the body of semantic tags and content. One example that uses the recognition of people's names is to find good connection paths between people, using colocation of names on a single page as an indication of connection between them [12].

6 Modeling the Web

The many substructures of the web are what provides us with the raw material for knowledge discovery, and part of the research activity underlying this is the construction of accurate data models that will explain and predict features of these substructures. Examples of this include the power-law distribution of inlinks [15], the bowtie structure of the web hyperlink graph [3], and the interaction between the hyperlink structure and the hierarchical structure of web sites [11].

One interesting feature of the web is trying to predict its rate of growth and the distribution of pages on different sites. In Figure 1 we show the distribution of the number of web pages per site. The linear appearance in a log-log plot illustrates a common theme in models of the web, namely that the distribution appears to have a tail that is a power law, namely

$$\Pr(\text{host size} > k) \approx k^{-alpha}$$

for large k and some constant $alpha > 1$. In this particular case, models for the size of a web site at time t have been hypothesized [13] to follow a multiplicative growth model, in which if $S(t)$ represents the expected number of URLs on a site at time t, then $S(t) = S(t-1)g(t)$ for some random variable $g(t)$ that represents the growth rate at time t. Note that this implies that

$$\log(S(t)) = log(S(0)) + \sum_{i=1}^{t} \log(g(i)).$$

If $\log(g(i))$ are identically and independently distributed with finite mean and variance, then the central limit theorem suggests that $S(t)$ should have a lognormal distribution in the limit as $t \to \infty$, which in fact seems to agree with data in Figure 1. One feature that is non-obvious is that Figure 1 actually shows the distribution of web site size observed at a single point in time, but in fact the different web sites are in various stages of their development, and most of the web sites are very small. Because they are in an early stage of their development, the asymptotics of the central limit theorem may not be valid for most sites.

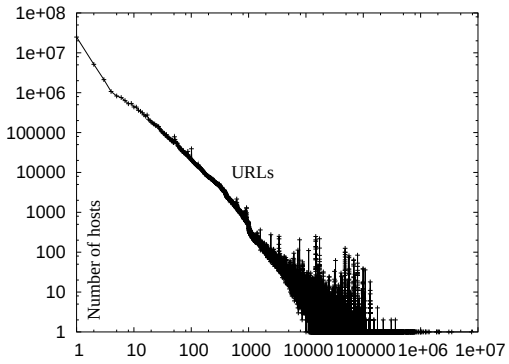

Figure 1. Distribution of the number of pages per web site. This data was taken when our database contained 48 million web sites.

7 Scalability

Our approach to knowledge discovery on the web has been done on a cluster of hundreds of machines, and is very

I/O intensive. It was noted in Section 3 that storage technology seems to have advanced at a rate that easily keeps up with the rate of growth of the textual web. At present there is a great deal that can be done by operating on a single URL at a time, extracting semi-structured data that can be processed globally with relatively little effort. In the future we might start to see much more aggressive approaches to the problem, in which large-scale supercomputers are applied to the problems and major portions of the web are operated upon in RAM. For example, modern supercomputers already exist that have fast interconnection networks and terabytes of RAM, but to our knowledge these have never been employed for any web knowledge discovery applications.

While we can devise growth models such as those described in Section 6, it is important to remember that the web is only a little over a decade old, and it is difficult to predict what the future of the web will hold. One of the biggest unknowns lies in mining activities surrounding the "deep web" that consists of HTTP interfaces to large databases. Moreover, an increasing amount of information (e.g., weblogs [14]) is being made available in semi-structured formats that lend themselves to knowledge discovery. The emergence of the Web as a global information space means that new forms of data and information will appear on the web, and there are likely to be significant challenges ahead.

8 Conclusions

We have attempted to describe some of the work that has been done by us and others on knowledge discovery and data mining of the Web. This burgeoning field offers many opportunities and challenges in distributed and parallel computation, and the rapid growth and evolution of the web promises to bring forth many new and interesting problems for future research.

Acknowledgments The WebFountain project represents the collective effort of many people over several years (in fact, too many to list here). Computer science researchers seldom get a chance to work on such large collaborative projects, and we owe a great deal to our colleagues that we have worked with over the years.

Thanks are due to Srinivasan Balasubramanian for his work on the crawler that generated the WebFountain data set described above. Special thanks also to Sridhar Rajagopalan, Daniel Gruhl, David Gibson, and Jason Zien for their contributions to many of the ideas discussed above.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499, 1994.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[3] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. L. Wiener. Graph structure in the web. In *Proc. 9th WWW*, pages 309–320, 2000.

[4] A. Z. Broder, S. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *WWW6/Computer Networks*, 29(8-13):1157–1166, 1997.

[5] Y.-Y. Chen, Q. Gan, and T. Suel. I/O efficient techniques for computing pagerank. In *CIKM 2002*, pages 549–557, McLean, Virgina, 2002.

[6] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K. S. McCurley, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. A case for automated larg-scale semantic annotation. *Journal of Web Semantics*, 1:115–132, 2003.

[7] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K. S. McCurley, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. A case for automated large-scale semantic annotation. *Journal of Web Semantics*, 1:115–132, 2003.

[8] J. Edwards, K. S. McCurley, and J. A. Tomlin. An adaptive model for optimizing performance of an incremental web crawler. In *Proc. of the 10th Int. World Wide Web Conf.*, pages 106–113, 2001.

[9] N. Eiron, K. McCurley, and J. Tomlin. Ranking the web frontier. In *Proc. WWW Conference*, 2004.

[10] N. Eiron and K. S. McCurley. Analysis of anchor text for web search. pages 459–460, Toronto, 2003.

[11] N. Eiron and K. S. McCurley. Link structure of hierarchical information networks, 2004. submitted.

[12] C. Faloutsos, K. S. McCurley, and A. Tomkins. Connection subgraphs in social networks, 2004. submitted.

[13] B. Huberman and L. Adamic. Growth dynamics of the world-wide web. *Nature*, 399:130, 1999.

[14] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. In *Proc. 10th Int. World Wide Web Conf.*, pages 568–576, Budapest, 2003.

[15] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proc. 41st FOCS*, pages 57–65, 2000.

[16] K. S. McCurley. Geospatial mapping and navigation of the web. In *Proc. 10th International World Wide Web Conference*, pages 221–229, Hong Kong, 2001.

[17] K. S. McCurley and H. D. Ziegler. *Advances in Cryptology: Electronic Proceedings of the Eurocrypt and Crypto Conferences 1981-1997*, volume 1440 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[18] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[19] A. Tomasic and H. Garcia-Molina. Performance of inverted indices in shared-nothing distributed text document information retrieval systems. In *Proc. PDIS '93*, 1993.