Proc. Int. Conference on Info-Tech & Info-Net ICII'2001, Beijing, China,
November 2001, IEEE Computer Society Press, pp. 1-6

A Method for Indexing Web Pages Using Web Bots

Boleslaw K. Szymanski and Ming-Shu Chung

Department of Computer Science
Rensselaer Polytechnic Institute, Troy, N.Y. 12180-3590, USA

Abstract

Exploring the content of web pages for automatic indexing is of fundamental importance for efficient e-commerce and other applications of the Web. It enables users, including customers and businesses, to locate the best sources for their use. Today's search engines use one of two approaches to indexing web pages. They either (i) analyze the frequency of the words (after filtering out common or meaningless words) appearing in the entire or a part (typically, a title, an abstract or the first 300 words) of the text of the target web page, or (ii) they use sophisticated algorithms to take into account associations of words in the indexed web page. In both cases only words appearing in the web page in question are used in analysis. Often, to increase relevance of the selected terms to the potential searches, the indexing is refined by human processing.

To identify so called "authority" or "expert" pages, some search engines use the structure of the links between pages to identify pages that are often referenced by other pages. Analyzing the density, direction and clustering of links, this method is capable of identifying the pages that are likely to contain valuable information. It is analogous to a well known citation analysis method developed in library sciences and used by such publications as the Science Citation Index. A slightly different approach is used in the Google Search Engine implementation which assigns to each page a score that depends on frequency with which this page is visited by web surfers.

The basic difference between the existing methods and the one discussed here is that these methods rely on a structure of web page linkages that lead from or to the indexed page. In contrast, our method uses the content of the pages linked to or from the indexed page for indexing. So our method uses a structure of words used by the linked pages, whereas the current methods use the structure of the connections between linked pages.

In this paper we propose and demonstrate usage of a new method based on bots which analyze content of the pages linked to or from the page of interest. We analyze the similarity of the word usage at the different link distance from the page of interest and demonstrate that a structure of words used by the linked pages enables more efficient indexing and search.

1 Introduction

With rapid growth of the Internet, the World Wide Web (WWW) has become one of the most important resources for obtaining information and one of the most important media for communications. The importance and popularity of the WWW reflect increasing sophistication of web sites and their growing interconnectivity. Increasingly, web pages represent collectively complex knowledge and information. No longer can web pages be considered isolated documents. As a result, it became practically impossible to thoroughly explore web pages manually by the users. Consequently, increasingly web bots are used to automate retrieving and collecting Web resource.

Exploring the content of web pages for automatic indexing is of fundamental importance for efficient e-commerce and other applications of the Web. It enables users, including customers and businesses, to locate the best sources for their needs. Majority of search engines today use one of two approaches to indexing web pages. In the first approach, the search engine selects the terms indexing a web page by analyzing the frequency of the words (after filtering out common or meaningless words) appearing in the entire or a part of the text of the target web page. Typically, only a title, an abstract or the first 300 words or so are analyzed. The second method relies on sophisticated algorithm that take into account associations of words in the indexed web page. In both cases only words appearing in the web page in question are used in analysis. Often, to increase relevance of the selected terms to the potential searches, the indexing is refined by human processing.

To identify so called "authority" or "expert" pages, some search engines (see the Clever Project at IBM [9, 1]) use the structure of the links between pages to identify pages that are often referenced by other pages. Analyzing the density, direction

and clustering of links, this method is capable of identifying the pages that are likely to contain valuable information. It is analogous to a well known citation analysis method developed in library sciences and used by such publications as the Science Citation Index. A slightly different approach is used in the Google Search Engine implemented at the University of Stanford [2]. This engine assigns to each indexed page a score that is defined by the frequency with which this page is visited by web surfers. The overall discussion of these approaches is given in [8].

The basic difference between the existing methods and the one discussed here is that these methods rely on a structure of web page linkages that lead from or to the indexed page. In contrast, our method uses the content of the pages linked to or from the indexed page for indexing. So our method uses a structure of words used by the linked pages, whereas the current methods use the structure of the connections between linked pages.

In this paper we describe and demonstrate usage of this new method in implementation that utilize bots for gathering and analyzing content of the pages linked to or from the page of interest. We analyze the similarity of the word usage at the different link distance from the page of interest and demonstrate that a structure of words used by the linked pages enables more efficient indexing and search.

Our method of indexing documents, including web pages, can be formalized as follows. Consider a document p_0 to be indexed. Let w_0 denote a set of words that document p_0 contains, and l_0 denote the set of links to other documents existing in document p_0. Current indexing methods process the set of words w_0 to index web page p_0 and occasionally use the set of links l_0 to identify expert web pages.

Let P_1 denote a set of all documents $P_1 = \{p_1, p_2, \dots p_n\}$, for some n, such that for each $1 \leq i \leq n$ set of links l_i of document p_i contains document p_0. We denote by W_1 a collection of sets of words $(w_1, w_2, \dots w_n)$, where w_i is the set of words on document p_i for $1 \leq i \leq n$.

Let P_2 denote a set of all documents $P_2 = \{p_{n+1}, p_{n+2}, \dots p_{n+m}\}$, for some m, such that for each $n < i \leq n+m$ the set of links l_i of document p_i does not contain a link to document p_0 but either it contains a link to document from set P_1 or there is a document p_j in P_1 that contains a link to p_i. We denote by W_2 a collection of sets of words $(w_{n+1}, w_{n+2}, \dots w_{n+m})$, where w_i is a set of words on document p_i, for $n < i \leq n+m$.

To index document p_0, our method uses a collection of sets of words that appear in any document p for which either there is a documents in P_1 with a link to document p or there is a link in document p to a document in P_1, where P_1 is a set of documents that contain a link to document p_0.

To limit processing, we consider the collections of words W_1 and W_2 only. Let s_i be a list of distinct words appearing in document p_i, for $1 \leq i \leq n+m$. Consider histograms h_1 and h_2 of frequency of appearance of words in lists s_i's, for $1 \leq i \leq n$ and $n \leq i \leq n+m$, respectively. We will cut out the tails of histograms h_1 and h_2, creating histograms H_1 and H_2 in which only those words are left that appear with very high frequency for H_1 and reasonably high frequency for H_2. Then, $H1 - H2$ is (contains) the set of words indexing content of document p_0.

Indeed, words that appear in H_1 are shared by all documents that link to document p_0, therefore they are highly likely to be relevant to the content of document p_0 or some other topic shared by documents in P_1. On the other hand words in H_2 are shared among those documents that link to or are linked from documents in P_1. By eliminating them, we remove topics which are shared among documents in set P_1 but not relevant to document p_0.

Depending on the need, the set of documents $H_1 - H_2$ can then be used in identifying words indexing document p_0 in combination with other methods of finding indexing words.

The advantage of the proposed method is that it evaluates content of the document not (or not only) on what the document claim to be (e.g., an expert on topic X) but on the bases of what documents that contain links to it consider it to be.

Consider the following example. A web page shown in Figure 1 uses words "Opera," "Verdi," " Ballet" but will be indexed by our method as relevant for searches using a keyword "Verdi" but neither "Ballet" (for which web page p4 is a better reference) nor "Opera" (for which web page p2 is a better reference).

This paper focuses on analysis of the relationship of web contents of pages with different depth of link interconnections based on exploring a large number of web pages by the web bots as the instrument.

Web bots automate the process of retrieving content of web pages and their interconnections. They recursively retrieve web pages by following hyperlinks in retrieved pages. Our result shows that content of two directly linked levels are highly clustered. However, the content for the interconnection levels which are not directly linked are not highly related. This observation led us to introduction of a novel method for indexing the web page content.

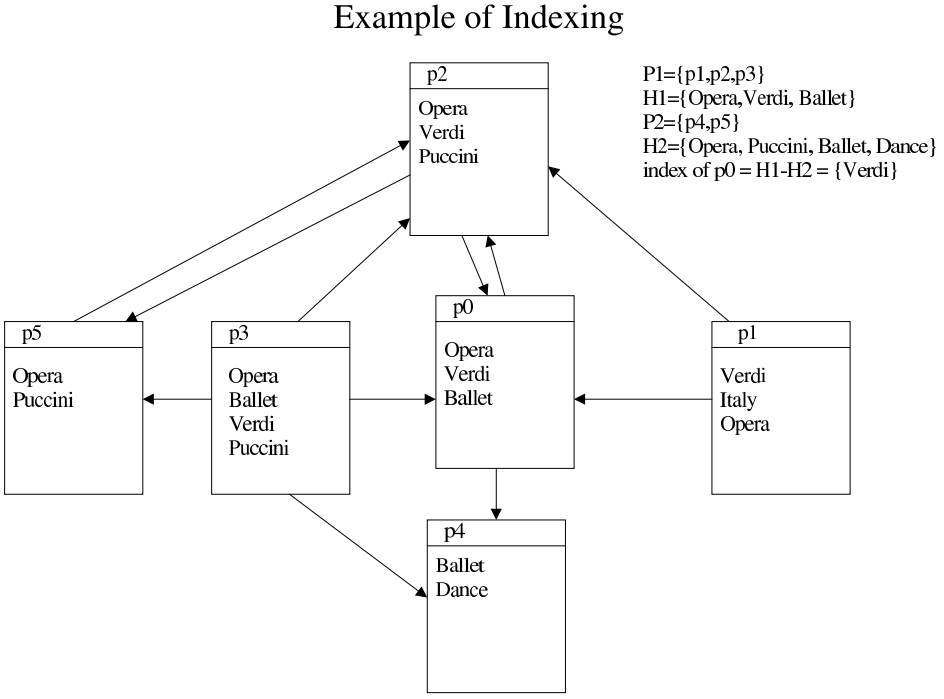

Figure 1: Example of Page Indexing

2 Design and Implementation of the Web Bots

A Web Bot is an automaton for retrieving and collecting Web resource on the Internet. Wet bots are also called web spiders, web worms, or web wanderers [10]. They are the programs recursively retrieving web pages by following hyperlinks in retrieved pages. Web bots are used in many ways. One of the popular uses is finding and indexing web pages inside search engines.

The Internet users usually surf the Internet by following links from page to page. This action can easily be mimicked by web bots. The users surf web pages randomly but often link to web pages highly relevant to their interests. Our goal is to explore the relations between web pages at different level of linkage revealed in breadth crawling. Hence, we look at web content of all pages linked to a target page, to understand the content of the target page. We refer to all pages linked to the target page as pages at same (first) link level of the target page. The pages that do not belong to the first link level but which are directly linked through a hyperlink to the first link level pages, are called the second link level pages of the target page. The third link level of the target page is defined inductively.

Our indexing counts the number of repeating words for each level. We then select certain number of words (twenty in the current implementation) with the highest frequencies of repeating at the first level as the candidates for indexing terms of the target page. Then we select certain number of words (again twenty in the current implementation) with the highest repeating frequencies at the second level. Any candidate indexing term found on this list is removed.

We implemented a web bot for collecting and counting words. We first discuss some design issues for implementing web bots. Then we describe the implementation in some detail. Finally, we show the result of our web exploration.

2.1 HTML Hyperlinks

2.1.1 Resource Anchor

HTML has a very forgiving syntax. More strictly speaking, web browsers can parse HTML sentences and tolerate sloppiness or error to certain extent. This make parsing hyperlinks more complicated. Our web bots parse hyperlinks, therefore we only focus on the syntax of links in our discussion.

2.1.2 Relative URIs

Relative Universal Resource Identifiers (URIs) are not rare in Web pages. It is an easy and convenient way for a HTML author to link different pages residing in different directories of the local sever. It is legal to put a relative URI in a hyperlink. The same relative URIs may appear differently in text but point to the same logical link and must be recognized as such. A robot has to figure out the full URI to avoid access the same page again and again. That is, a robot has to prevent looping during crawling. To this end, a robot calculates the base URI according to the following precedences (highest priority to lowest)[4]: by the BASE element, by meta data discovered during a protocol interaction, or that of the current document.

2.1.3 Protocols

URIs typically consist of three pieces[5]: naming scheme of the mechanism used to access the resource, name of the machine hosting the resource, and name of the resource itself, given as a path. A mechanism used to access the resource is usually referred to as a protocol. A robot does not have to index and access every hyperlinks that it encounters. What action the robot takes, depends on what kind of a resource it intends to collect. A robot could parse the protocol of hyperlinks and determine if the resource is within a server. In our application, we are only interested in web page contents. Thus, our robots collect only the links accessing HTTP (web) servers.

2.2 HTML Character Encoding

Some characters are not valid in URIs, like white spaces, double quotations and single quotations because those characters would confuse the URI expression. As mentioned before, a white space is invalid within URI because it would fragment the URI. A quotation mark would be mistaken for an end of URI, etc. Character encoding can get around this problem. That is, some special characters are encoded into another forms and decoded by the web servers or browsers.

Because space is quite often needed within URIs with parameters, it is encoded as a plus sign [10]. As a result, plus sign is encoded as `%2B`. URI with parameters usually appear in HTTP when POST method is used. Any ASCII characters except space can be encoded as a percent sign and two digital hexadecimal code. Furthermore, different kind of languages could have different encoding schemes.

Another encoding scheme used in HTML are HTML Character Constants. There are many special characters that are encoded as an ampersand(&) and an abbreviation ended with semicolon(;), like `<` for <. Our robots collect English words as a content of a page. Except for http links, they can simply ignore any encoding and only capture text.

2.3 Robot Exclusion

Robot exclusion is a common and simple way for each web server to state visit policy. It became widely adopted by robot community. Robot exclusion uses two mechanisms: the Robots Exclusion Protocol and Robots META tag. Robot exclusion is based on understanding between robot writers and web administrators. There is no ways to strictly prohibit robots from brute-fore crawling without suing a page file privilege restriction or using password protection.

The Robot Exclusion Protocol is very simple. It tells a visiting robot which directory is disallowed for common crawling and which directory is forbidden for some named robots. A web server administrator can state the visit policy in a simple text file named robots.txt and put it at one of the top-levels of URI for the site.

In our experience, even though the Robots Exclusion Protocol is accepted as a way to provide exclusion information to robots, it is not widely used by web administrators.

The Robot Exclusion Protocol must be fulfilled by web sever administrator. The HTML authors can state their exclusion policy without administrator's involvement by using a Robot META tag in the HEAD section of a page. The meta name is `robots` and content is [NO]INDEX,[NO]FOLLOW. It defines whether or not this page should be indexed and/or the hyperlinks in this page should be followed. Unfortunately, only few robots support such tags.

2.4 Implementation

Our robot is implemented in Java 1.2 using classes that can be classified into six categories described below.

2.4.1 Communications

Communications is simple in our robot. A full URI is sent to a web server and the received data stream is transferred to a parser for finding hyperlinks in it. We implicitly use only HTTP GET method, so the robot will not get resource via HTML

forms using POST method. However, HTML forms are usually associated with CGI programs and we do not intend to collect resource via interacting with CGI programs at server side. Furthermore, the CGI directories are usually excluded for robot crawling. Therefore we believe that this restriction of communication is acceptable.

2.4.2 HTTP Parser

The parser reads in data stream and parses it line by line. The parser looks for the character pattern ”href” in each line. If there is no such pattern, the line is simple skipped. Otherwise, hyperlinks are individually captured from the line and the parser checks if the hyperlink is a full URI. If it is not, the full URI have to be constructed. In this case, the parser translates every relative URI to a full URI. Then, the protocol of the hyperlink is checked. If it is other than HTTP protocol, a link is dumped.

2.4.3 Robot Exclusion

To enable efficient data collection, the data structure should be carefully chosen. We use Java containers - HashSet, to store exclusion lists and the sites from which we have gotten them. Every time the robot tries to get a page, the base URL of this page is calculated and check against the exclusion list of the current page. If it is in the list, we give up getting to this page. The URI list and the robot exclusion lists are stored in two different HashSet and can be easily output into a file at the end of the exploration.

2.4.4 Word Collecting

Not every resource that the robot store in the hyperlink list is what we want. For example, links could point to a gif, ps or pdf files. These are not what the robot suppose to collect. Therefore, before the robot tries to gets the resource, the trailing part of its URI is checked first for the resource type.

There are several steps in collecting words. When a page is obtained, all the HTML tags are removed first and the remaining content of the page is split into words. These words often contain some characters others than letters or digits, like comma(,), colon(:), semicolon(;) and so on which are stripped. We remove also the HTML constants or any other special characters in word tokens.

2.4.5 Database

We maintain two databases. One is simple in-memory database which stores all the hyperlinks. Another is a stand alone database, HypersonicSQL, which stores all the words[6].

The in-memory database is implemented by Java containers - *HashMap* and *HashSet*. *HasMap* is used for maintaining link structure of all link levels that we collect (three in all). *HashSet* stores hyperlinks themselves. Every node in the linking structure contains a a key and a link that can be easily obtained via its associated key. When robot want to get to a new page, it checks if the link of the page exists in HashSet to avoid looping in its traversal through the Web.

HypersonicSQL is a small and high-performance database. It supports basic SQL that we need. HypersonicSQL has JDBC driver and the queries can be easily integrated into our program.

2.4.6 Logging

When the robot connects to web servers, there is no guarantee that the communication will succeed. Hence, the robot detects networking and log-in problems. The most typical networking problems that we encountered were:

- Pages not found on servers:
  This error is usually caused by broken links. To parse this error message, the robot reads HTML title to see whether two key words, *Not* and *Found*, are within title tag.

- Unknown host:
  The sever specified in URI cannot be located. This error triggers a Java Exception.

- No route to host:
  This error happens when robot tries to connect to a web server that does not respond for a long time. This is caused usually by temporarily unavailable network or downed server.

3 The Results of Web Exploration

Eight web sites were chosen as the starting points of our exploration. Each of eight robots that we unleashed began from a corresponding root URI and searched all hyperlinks from the first to the third level. Every page associated with a hyperlink that the robot collected was parsed for words. At each level, the words were counted for repeating frequency. Because the number of different words encountered was in thousands, twenty to forty words with highest count were chosen at each level. Words which are not specific to the page content, like pronouns or digits, were excluded. The robots looked for robots exclusion lists for every sites and obeyed exclusion polices.

Table 1: Comparing Levels for the same words among most frequently used 20 words

Root Site	1&2	1&3	2&3	1&(2 or 3)
www.rpi.edu	5	4	6	9
www.cs.rpi.edu	13	10	10	10
www.ecse.rpi.edu	7	4	6	12
www.rpi.edu/dept/biomed/WWW	7	3	4	11
lallyschool.rpi.edu	5	2	8	13
www.albany.edu	7	0	3	10
www.cs.buffalo.edu	8	2	8	16
www.sunysb.edu	6	0	5	10

4 Conclusion

The results of our exploration shows that the the most frequent words at the directly linked levels are highly relative to each other, or, in other words, they are clustered together. However, when we go further away and look at the most frequent words at level one and three three, they diversify from each other. level one. Unless a web site is highly biased for some content, the words frequent at level one are not similar to words frequent at level three. The interesting level is the level two which have similar set of the most frequent words to both level one and three.

Exploring web content is one of the most important applications of the web bots. Our experience indicates that they can be used for classification of the web page content much finer that supported by a META tag keyword. This meta tag is intended for specifying what kind of content the page contains, but it is usually not precise. By using web bots and our technique of selecting words characterizing a web page, we obtain more precise categorization of page content.

References

[1] Clever Project: http://www.almaden.ibm.com/cs/k53/clever.html

[2] Google Project: www.google.com

[3] HTML 4.01 Specification *3.2 SGML constructs used in HTML*.

[4] HTML 4.01 Specification *12.4.1 Resolving relative URIs*.

[5] HTML 4.01 Specification *2.1.1 Introduction to URIs*

[6] http://www.lynx.ch/contacts/ /thomasm/hSql.html

[7] http://www.yahoo.com/robots.txt

[8] "Hypersearching the Web," *Scientifi c American*, June 1999, http://www.sciam.com/1999/0699issue/0699raghavan.html

[9] Jon M. Kleinberg, "Authorative Sources in a Hyperlinked Environment," *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, SIAM/AC-SIGACT, 1998.

[10] David Pallmann, *Programming Bots, Spiders, and Intelligent Agents in Microsoft Visual C++*, 1999.