

Structural Abstractions of Hypertext Documents for Web-based Retrieval

Jitender S. Deogun

The Department of Computer Science & Engineering
University of Nebraska
Lincoln, NE 68588, USA

Hayri Sever

Department of Computer Science & Engineering
Hacettepe University
06532 Beytepe, Ankara, Turkey.

Vijay V. Raghavan

The Center for Advanced Computer Studies
University of Southwestern Louisiana
Lafayette, LA 70504, USA

Abstract

There have been conflicting views in the literature on the capability of tools and mechanisms for storing and accessing information over Internet. On one hand it has been claimed for a long time that World Wide Web offers a chaotic environment for Web agents to extract information because the description of a document by HTML is easily comprehensible by humans, but is not so by machines. On the other hand, it has been hypothesized that information is sufficiently structured to facilitate effective Web mining, especially for electronic catalogs. In this article we do not intend to take position on this matter, but rather investigate the performance of a search engine while indexing more logical elements of HTML documents and while increasing the scope of indexing process.

1. Introduction

The exploration of structures is not a new approach. Etzioni in [3] argued the structured Web hypothesis: Information on the Web is sufficiently structured to facilitate effective Web mining. He suggested following three subtasks for Web mining, namely *resource discovery*, *information extraction*, and *generalization*. The resource discovery focuses on the automatic in-

dexing and (furthermore classifying) Web documents. Information extraction involves in dynamically extracting specific information from newly discovered Web resources, such that the need for hand-coded "wrappers" to access the resource and parse its response can be eliminated or reduced. Generalization discovers regular patterns at (or inductively learns) individual Web sites and across multiple sites.

ShopBot [2], a Web mining agent specialized on electronic catalogs, uses descriptions of domains and vendors as prior knowledge to compare vendors by an attribute (say, price) for given a characterization of the desired product. The domain description includes information about product attributes useful for discriminating between different products and between variants of the same product, heuristics for understanding vendor pages, and seed knowledge for inductive learning. On the other hand, the vendor description consists of the URL address of a search form provided by the vendor, a function mapping product attributes to fields of that form, and a parsing function that extracts a set of individual product descriptions from pages returned by the search form. Thus, the learning module of ShopBot involves the structures in which product descriptions are presented and the search form from which the desired product attributes are extracted. Structural regularities are based on the following observations: (a) Online stores are designed

so consumers can find things quickly, (b) Vendors attempt to create a sense of identity by using a uniform look and feel, and (c) Merchants use whitespace to facilitate customer comprehension of their catalogs, even if they use different product description formats. A case study on electronic commerce sites such as "Internet Shopping Network (<http://www/internet.net>)" and "NECX Direct (<http://necxdirect.necx.com>)" reveals that `
<a>texttext` and `<a>texttext
` are common line descriptions in which product information is encoded [2].

Similarly, WebSeek, image and video search engine located at <http://www.ctr.columbia.edu/webseek>, also utilizes structural regularities to index the images and videos [5]. It uses key terms contained in `` and `[hyperlink text]` as well as terms in directory names to classify the subjects of hyper objects through a key-term dictionary. This dictionary provides a set of mappings from key terms to subject classes.

The characteristic features of Web catalogs¹ we consider important for the retrieval engine are as follows.

- **Highly structured documents.** Of them we are mainly interested in paragraphs, lists, and tables that are used in characterizing description of an information unit², in regard to user's need. In the case of tables it is usual to find the information unit defined by attributes well as their domain values (or index words). A concept can be defined in terms of either its intension (i.e., its attributes) or its extension (i.e., its tuples). For the sake of clearness, we call a relation a complex concept and a single attribute a simple concept. The table structure contains both intension and extension of a complex concept. A paragraph contributes the extension of given a concept and is represented in terms of index words. A list structure is a hybrid structure of above two. The contents of items of a list structure represent the extension of a given concept and the items themselves constitute the intensional part of a complex concept with missing attribute names.
- **Hierarchical organization of Web pages.** Products in Web catalogs are usually presented to prospective customers through a number of hierarchies based on some predefined (and non exclusive) categories (e.g., computer products may be

categorized by PC, MAC, Supplies, Vendors, and Network). Given that there is no enough textual content to capture an embedded hierarchy and absence of appropriate mechanism to label semantical links, it is almost impossible to extract the embedded hierarchies, though some generalization relationships can be very valuable from the perspective of processing queries for gathering relevancy information. For example the quality of the retrieval output obtained as a result of processing the query that requires to find 166MHz desktops is likely to be improved in terms of precision and recall metrics when the categorization of PCs with respect to the case and processor type is available to the retrieval engine. It is, however, our experience that the Web pages located at close proximity of bottom nodes of hierarchies (when visited by breadth-first expansion upon disregarding backup links) usually contain product information or aggregation of products.

Given that a hypertext database of products is a collection of highly structured pages, the interesting two questions would be what the logical units of the information, which are useful enough to satisfy users' needs, and in which nodes (or region) we should attempt to locate these useful structures. To address these issues, in this article, we propose to experimentally determine

- the types of HTML structures to be utilized for indexing purposes, and
- the types of Web pages in given a hypertext database used as a document collection of the retrieval engine.

2. Functional Modules of Architecture

This work is a prelude step for the implementation of a prototype that had been proposed in response to the need for an intelligent information system to support various missions of the Defense Logistics Agency (DLA). The goal of the proposed system is to develop a distributed information system in which (a) the suppliers can potentially identify the number or volume of their parts needed by customers, (b) the customers in need of parts can locate and possibly order parts from suppliers, and (c) the DLA can evaluate the overall state of the military readiness. In this section we review some corresponding components of the architecture.

¹A Web catalog is an electronic catalog described by HTML and accessed through HTTP.

²Throughout the paper we use the terms *concept* and *information unit* interchangeably.

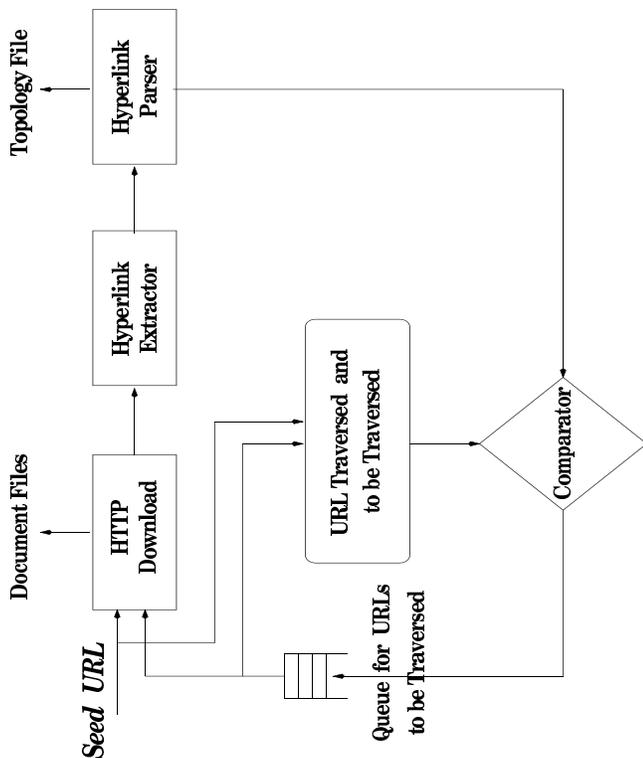


Figure 1. The View of Web Crawler

2.1. Web Crawler

A WebCrawler has been designed and implemented to retrieve documents from the world wide web and create a database. The implementation has been done in JAVA to effectively utilize the power of its platform independence, secured access and powerful networking features.

Methodology. As seen in Figure 1, WebCrawler agent is composed of three tightly-coupled submodules, namely Downloader, Extractor, and Parser. Initially the downloader starts with a seed (root) URL and then navigates Web catalog by a breadth-first expansion using the queue for URLs to be traversed. It retrieves Web documents via hypertext transfer protocol (http) and in the process, passes the HTML document to the extractor. The detection of bad links are handled by the downloader by trapping the HTTP status code returned by the HTTP server of the URL to be visited. The hyperlink extractor detects the hyperlinks in the web documents, extracts them and passes them to the parser for further processing. It searches for HTML tags of the form `< a href = "http : //..." >` or `< frame src = "..."`. The parser converts these relative URLs to absolute URLs following the Internet Standards (RFC 1738, 1808) drafted by the Network

Working Group. No parsing is necessary for absolute URLs. The host name (more specifically, the domain name of the network host) of each of these URLs is then compared with the host name of the seed URL and only those URLs whose host name match with that of the seed URL are added to the queue. Care should be taken so that any new URL that is added to the queue is not a repetition of any of the URLs that has been already traversed or will be traversed. Thus the WebCrawler retrieves all the web documents within the site specified by the root URL. URLs added by the parser to the queue are restricted to certain specific types only.

Each of the downloaded HTML document is stored in a file for further processing. For each seed URL specified, the program also outputs a list of the URLs traversed and their topology information. The topology information contains the list of (absolute) URLs (*i.e.*, links) that are referred to in each document URL.

2.2. Retrieval Engine

Retrieval engine stands on two main legs: descriptor-level and concept-level retrieval. For descriptor-level document/query processing we use the Isite/Isearch system, which is a freeware software for text retrieval. Isite database is composed of documents indexed by Iindex and accessible by Isearch. Isite/Isearch allows one to retrieve documents according to several classes of queries. Isearch features give the user many options for composing queries with search and target elements. The simple search allows the user to perform case insensitive search on one or more search elements (fields). Partial matching to the left is allowed. The Boolean search allows the user to compose a two-term query where the two terms are related by one of the Boolean operators AND, OR, and ANDNOT. "Full Text" is the default search domain unless the user selects a particular element for a term from the term's pull-down menu. The search elements we experimentally found useful for electronic catalogs are table and list structures. The advanced search form accepts more complex Boolean queries that are formed by nesting two-term Boolean expressions. To narrow a search domain from "Full Text" to a single search element, the term is prefixed with the element name and a forward slash. The information targeted for return by a query may be specified by choosing target elements from a pull-down menu. Isearch is capable of performing a weighted search that employs the inverse document frequency of terms.

We consider concept-level retrieval a natural application of query expansion techniques and an alterna-

tive model (to descriptor-level retrieval) in which a user may express its needs through a query topic (or a concept). Our view of a concept includes its intensional and extensional descriptions. In other words, the user may define a query topic in terms of either its subqueries or its relevant documents (or objects). A query topic is said to be simple if it is solely based on text references such as keyword(s) or phrase(s). We use simple topics as building blocks to introduce the notion of composite topics. A composite topic might be either an aggregation or generalization of simple or other composite topics. We conceptualize a query topic by a set of production rules. A production rule is simply implication of a query topic (or a decision concept) by subqueries (or conditional subconcepts) that are conjoined by logical 'AND' operator. The logical 'OR' operator is implicitly employed in evaluating alternative definitions of the same topic. In other words, the expression on the left side of a rule is called a pattern in the sense that the existence of a pattern implies the applicability of a decision concept to a document. A weight may be associated with a production rule as an indication of the user's preference (or confidence) on the existence of the pattern within a document. A set of rules for a query topic constitutes a goal tree involving AND/OR arcs. Furthermore, to interpret weights it is satisfactory to use the functions minimum, maximum, and product to propagate the weights across AND or OR arcs and implication nodes, respectively [4]. Finally the evaluation of a goal tree yields the extension of the query topic; that is, a set of ranked documents is returned by the concept-level retrieval engine in response to the selected query topic.

The concept-level module is implemented on top of the descriptor-level retrieval. This is because a rule base for a query topic is converted to a set of conjunctive normal forms (CNFs) of explicit terms, called minimum term set (MTS). Each CNF in MTS is treated as an independent query and submitted to the retrieval system. The retrieval output consists of a set of pairs containing the identifier and the retrieval status value (RSV) for each document. The confidence factor of the CNF is reflected to each document in proportion to its rank in the retrieval output. This process continues until all CNFs in MTS are exhausted. To accumulate retrieval set of documents in an iteration we keep a global retrieval output. Just before starting a new iteration (to evaluate next CNF in MTS), the retrieval output of currently processed CNF is merged with the global retrieval output. If duplicated documents are encountered during merging two lists, only the one with greater RSV is picked up.

3. Experiment

We configure an experiment such that performance of the system with respect to a predefined collection of queries is evaluated within controlled search region for the cases in which the effect of using only (a) table structures, (b) table and list structures, and (c) table, list, and paragraph structures. For each case, the size of search region (i.e., the set of Web pages) is accumulatively increased with respect to the type of pages.

3.1. Method

We categorize nodes of a Web catalog into mutually exclusive sets of referential, special, and miscellaneous pages. Referential node is a node in which detail information for a product is specified. Special node is a node in which a number of products sharing some common features are grouped. Miscellaneous node is a node which is neither referential nor special node; that is, a miscellaneous node does not contain useful information for our indexing purposes. To identify the types of nodes, we use a subjective classifier based on words in URLs as well as patterns in Web pages.

We measure the quality of retrieval output by MZ metric [1]. Assuming at least one document is relevant to a user query, it is defined by the formula

$$MZ(a, b, c) = 1 - \frac{a}{a + b + c},$$

where a , b and c are the number of relevant & retrieved, nonrelevant & retrieved, and relevant & not retrieved documents, respectively. Notice that higher quality of retrieval corresponds to lower MZ value, which varies between zero and one. The main steps of the experiment is outlined as follows.

Step 1. Choose a Web catalog.

Step 2. Determine a set of queries.

Step 3. Let a set of structural abstractions, denoted by $S_{structures}$, be $\{\{table\}, \{table, list\}, \{table, list, paragraph\}\}$ and i index variable on such set. Let $i = 0$.

Step 4. Let the set of the search region, denoted by S_{region} , be $\{r, rUs, rUsUm\}$, where r, s , and m are set of referential, special, and miscellaneous pages. Let j be index variable on S_{region} . Let $j = 0$.

Step 5. Filter out the structures pointed by $S_{structures}(i)$ from the Web pages in $S_{region}(j)$.

Step 6. Construct the inverted-term file by using the `Lindex` software.

	autos	warehouse
total number of nodes	6320	10826
number of ref. nodes	1061	5078
number of special nodes	587	1006
number of misc. nodes	4672	4742
mean indegree	2.99	3.81
standard deviation of in-degree	7.52	10.26
standard deviation of outdegree	5.33	3.15

Table 1. Features of Web Catalogs

Step 7. Run each query and determine its MZ metric value.

Step 8. Record the average MZ metric value for $S_{structures}(i)$ and $S_{region}(j)$.

Step 9. Increase j by 1.

Step 10. If $j < 4$ then goto Step 5.

Step 11. Increase i by 1.

Step 12. If $i < 4$ then goto Step 4 otherwise stop.

3.2. Configuration

The preliminary tests made on two Web catalogs: <http://autos.yahoo.com>, which we call *autos*, and <http://www.warehouse.com>, which we call *warehouse*. In table 1, statistics (mean and spread) of in/out degrees of nodes for these two catalogs are shown. Note that distribution parameters were computed after removing backup links³.

To identify the types of Web pages, we hard-coded the classifier defined by some members of our research team using URLs, in/out-degrees (not found significant), and contents of different types of Web pages as background knowledge. In the following, the classifier of autos catalog is given as an example.

"ppage" or "info" \in words(URL) \Rightarrow referential
 "newcars" \in words(URL) and
 $\langle make, model, type, baseprice \rangle \in$
 Intension(WebPage(table)) \Rightarrow special
 otherwise \Rightarrow miscellaneous

Three random samples were taken from the classified pages and found perfectly accurate. The answer of how to automatically model such classifier is beyond the

³A backup link is identified as follows. Let μ and σ stands for mean and standard deviation of in-degrees of nodes. If a node's in-degree value is greater than $\mu + 2\sigma$, then it is simply assumed that any edge coming to that node is a backup link.

	{t}	{t,l}	{t,l,p}
{r}	0.8	0.8	0.57
{r,s}	0.22	0.22	0.34
{r,s,m}	0.25	0.26	0.36

Table 2. Average MZ values

scope of this article. The consistency in naming directories in URLs and columns of tables allowed us to come up with simple classifiers.

The extensive use of tables in autos catalog is reflected in the results shown in Table 2. In warehouse catalog, we see that the uses of tables and lists are predominant – it may be worth noting that we only downloaded pages of PC, Mac, and Supplies hierarchies from that catalog. The warehouse catalog substantially differs from autos catalog in its use of CGI queries to pull out Web pages. It uses, for example, "catproduct.dll" for retrieving referential nodes and "searchinterface.dll" for retrieving special nodes, which made generation of classifier easier for that site.

For the experiment we have used the advanced query interface of Lsearch package that allows user to define nested Boolean queries. The set of queries (10 for each catalog) were selected with respect to the criterion that a query set should not favor particular structure(s).

3.3. Discussion

In Table 2, each row corresponds a particular combination of referential, special, and miscellaneous Web pages; each column shows a particular combination of HTML structures we indexed – note that header elements of HTML document were indexed by default. Each cell in this table, shows average MZ values of a corresponding strategy, e.g., indexing only tables in collection of referential Web pages yields 0.8 MZ value in average.

The results in Table 2 shows that the best performance of retrieval engine is attained when indexing tables in referential and special pages. In Table 1 we notice that 45% of total nodes in these two catalogs are made up of referential and special nodes. Combining this information with the best performance strategy in the context of our preliminary work, we see that indexing 55% more pages would not necessarily yield better performance. In fact the performance is degraded when Web pages that are not directly related to products are indexed. The effect of the trend in representing products by tables and lists on the performance of retrieval engine can be observed when we compare the third col-

umn of Table 2 with preceding two. It is shown that no added-value can be obtained by indexing paragraphs.

4. Future Directions

There are three problems with off-line retrieval engines explained as follows.

1. Suppose that Web catalog is kept in back-end database and the only way for a remote user to access product information is to interact with a front-end engine (e.g., <http://www.{mwmicro,computerhouse}.com>). In this case the off-line agent without human intervention attempts to index useless Web pages which constitute a big fraction of total pages in corresponding URL site.
2. Since the contents of Web pages (and hence, links) change over time, local database of the off-line agent might contain absolute indices. Given that current version of HTML does not even support incorporating the version information into Web documents, the interesting question is here as to how to manage evolution (insertion of new pages, deletion of old pages, and changes to existing forms).
3. When CGI queries become only way to get product information, there would be more than one way to access the same Web page in given a catalog. In this case, WebCrawler will treat those aliases as distinct URL addresses, and in result the same page will be indexed as many as its aliases present in the catalog.

The first problem actually warrants incorporation of on-line agents into the retrieval engine. Since the second one requires mechanisms for periodic maintenance, we plan to update Isearch package such that it is possible to make dictionary operations on the inverse-index file. The last problem is an open question as at present we do not know of any practical solution.

Acknowledgment

This work is supported by the Army Research Office of USA, Grant No. DAAH04-96-1-0325, under DEP-SCoR program of Advanced Research Projects Agency, Department of Defense. Hayri Sever is a visiting assistant professor at Department of Computer Science & Engineering, University of Nebraska, Lincoln, NE 68588, USA.

References

- [1] P. Bollmann and V. S. Cherniavsky. Measurement-theoretical investigation of the mz-metric. In R. N. Oddy, S. E. Robertson, C. J. van Rusbergen, and R. W. Williams, editors, *Information Retrieval Research*, pages 256–267. Butterworths, Boston, 1981.
- [2] R. B. Doorenbos, O. Etzioni, and D. S. Weld. A scalable comparison-shopping agent for the world-wide web. Technical Report UW-CSE-96-01-03, Department of Computer Science and Engineering, University of Washington, 1996.
- [3] O. Etzioni. The world-wide web: Quagmire or gold mine? *Communications of the ACM*, 39(11):65–68, Nov. 1996.
- [4] B. P. McCune, R. M. Tong, J. S. Dean, and D. G. Shapiro. RUBRIC: A system for rule-based information retrieval. *IEEE Trans. on Software Engineering*, 11(9):939–944, 1985.
- [5] J. R. Smith and S.-F. Chang. Visually searching the web for content. *IEEE Multimedia*, pages 12–20, July 1997.