The following paper was originally published in the
Proceedings of the USENIX Symposium on Internet Technologies and Systems
Monterey, California, December 1997

Using the Structure of HTML Documents to Improve Retrieval

Michal Cutler, Yungming Shih, Weiyi Meng
State University of New York at Binghamton

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org/

Using the Structure of HTML Documents to Improve Retrieval

Michal Cutler, Yungming Shih, Weiyi Meng
Department of Computer Science
State University of New York at Binghamton
Binghamton, NY 13902
{cutler, meng}@binghamton.edu

Abstract

The World Wide Web (WWW) is a gigantic information resource, which is growing daily. As more and more data are added to the WWW, it is becoming increasingly difficult to effectively locate useful information from this environment. In this paper, we propose a method for making use of the structures and hyperlinks of HTML documents to improve the effectiveness of retrieving HTML documents. Our study assigns the occurrences of terms in a document collection into six classes according to the tags in which a particular term appears (such as Title, H1-H6, and Anchor). Based on the assignment, we extend the weighting schemes in traditional information retrieval by incorporating different importance factors to terms in different classes. The rationale is that terms appearing in different places of a document may have different significance in identifying the document. For this research we have built a Web based search tool, Webor, created a testbed, and conducted extensive experiments to determine an optimal class importance factor combination. Our study indicates that substantial improvement of retrieval effectiveness can be achieved using this technique.

1. Introduction

The World Wide Web has become an important information resource today. The popularity of the Web is primarily due to the tremendous amount of information available, and the ability to browse and publish information. At the same time, as more and more data are added to the WWW every day, it has become increasingly difficult to effectively locate useful information from this environment.

There are typically two different approaches for finding information in the WWW. The first is *browsing*. Browsing based systems organize information in the WWW into category hierarchies. Examples of this type of systems are Yahoo [Yaho96] and Magellan [Mage97]. The second approach is *searching*. When using a searching based system (called a search tool), a user submits a query and the system returns a list of web pages (usually the URLs of these pages) that are potentially useful to the user. Many search tools have been employed and some examples are Alta Vista [Alta96], WebCrawler [Pi94], Lycos [CMU95, Mau97], OpenText [OTC96], WISE [YuLe96a, YuLe96b] and Microsoft Index Server [Mi97]. Most browsing based systems also provide a more general searching capability.

A search tool is essentially a Web-based information retrieval system. Such a system typically consists of two components. One is a robot-based *indexing engine* which recursively downloads web pages, indexes the contents of downloaded documents, extracts URLs from them and downloads more web pages using these URLs. In the end, an index database organized as an inverted file is constructed. The second component is a *search engine*, which compares each user query to the downloaded pages through the index database and returns a list of web pages, which are potentially useful to the user. Usually, the returned web pages are ranked based on some similarity function.

The work presented in this paper is based on the vector space model [Salt83]. In traditional vector space based information retrieval, each document is represented as a vector $(w_1, w_2, ..., w_k)$, where k is the number of distinct terms in all documents in the system (usually after

stopwords such as `a' and `of' have been removed and a *stemming* algorithm has been applied to convert words to their stems), and w_i is a real number indicating the *weight* (or significance) of the *i*th term in identifying the document. If a term does not appear in a document, then its corresponding weight in the document vector is zero. The weight of a term *t* in document *d* is usually computed from two factors. The first factor is the *term occurrence frequency*, *tf*, which is the number of times *t* appears in *d*. Intuitively, a larger *tf* should imply a larger weight, so the weight of term *t* should be proportional to *tf*. The second factor is the *document frequency*, *df*, which is the number of documents in the collection that contain *t*. Intuitively, the more documents that contain *t*, the less significant the term is in differentiating *d* from other documents. Therefore, a larger *df* should imply a smaller term weight. In other words, the weight of term *t* should be proportional to the *inverse document frequency*, *idf*, of *t*. A commonly used formula for computing the weight of *t* in *d* is $tf \cdot idf$.

In the vector space information retrieval model [Salt83, Salt89], each user query is also represented as a vector ($q_1, q_2, ..., q_k$), where q_i is the weight associated with the *i*th query term. Usually, q_i is either zero, indicating that the query does not contain the *i*th term, or one, indicating that the *i*th term is in the query. Sometimes $tf \cdot idf$ is also used for computing q_i . The closeness (or similarity) of a query and a document is often computed based on the inner product of their vectors. Similarities are often normalized to between 0 and 1 through the use of a normalization factor.

In this paper, we study extending traditional approaches for information retrieval to the WWW environment. There are two key differences between the documents in this environment and those used in traditional IR systems.

1. The structure of HTML documents is easily available through HTML tags. For example, in an HTML document, we can easily tell whether a term appears in the title, one of the six headings or whether it is emphasized by using an underscore, italics or bold characters. Such structures provide information about the content of a document. Intuitively, terms that appear in the title, header, or are emphasized in the text are more important for retrieval than the rest of the terms. So by storing the structure information of HTML documents in the index and assigning an appropriate importance value to the appearance of the terms in each structure, the structure and the importance information can be used to improve the rank assigned to retrieved documents.

   Traditional IR systems typically disregard information about the structure of a document. The main reason for this is that commonly this information is either not available, or is hard to acquire.

   It is well known that a search can be improved by taking the structure of a document into account. Several search engines already use tag information to improve ranking. AltaVista [Alta96], HotBot [HotBot], and Yahoo [Yaho96] score a document higher if query words or phrases are found in the title of a web page. Lycos [Mau97] uses position information on query term occurrence (title, body, header, one of 100 most relevant words) in the rank function. The insight into why some structure information is used while other structure information is ignored is not published. In addition, the weights assigned to the various structures in the similarity function, and the information on how these weights were derived are not available. This paper presents a systematic investigation of HTML structure information, explains how the importance assigned to tagged terms was derived, and how the similarity function was modified.

2. HTML collections contain additional information about each document *d* that has hyperlinks to it in other documents of the collection. Typically, when authors add a hyperlink to a document *d*, they include in the Anchor tag, a description of *d* in addition to its URL. These descriptions have the potential of being very important for retrieval since they include the perception of these authors about the contents of *d*. In particular, they may provide good synonymous and related terms which are not included in the text of *d*, but may be included in user queries. So by adding the anchor information of HTML documents to the index we may be able to retrieve documents that could not be retrieved otherwise. In addition, using an optimal importance value for anchor terms may improve the rank assigned to retrieved documents.

   In traditional IR approach, only terms included in a document are used to automatically index it

(unless a thesaurus is used). There have been several approaches to combining hypertext with information retrieval [Agost96]. For retrieval from a hypertext medical handbook, Frisse took into account the occurrence of query terms in the descendants of a hypertext document [Frisse88]. Dunlop [Dunlop93] used the cluster of documents citing and cited by a document for retrieval. Retrieval from hypertext was also investigated by Croft [Croft93], and Frei [Frei92].

Research was also conducted on how to take advantage of the additional information available in the hierarchical (or graph) structure of Web collections to improve retrieval. Yuwondo and Lee [YuLe96a, YuLe96b], considered a number of alternative ranking algorithms. The algorithms are based on the idea that neighbors of a web page are related to the page and that this information can be used to improve ranking. *Boolean spread activation* increased the belief in a document if query terms appeared in its neighbors. *Most-cited* increased the belief in a document if query terms appeared in its parent pages. *Vector spread activation* increased the belief in a document depending on the similarity of its children to the query. Hypursuit [Hypur96] combined document similarity with hyperlink semantic similarity. The hyperlink semantic similarity is based on two documents having a path of links connecting them, the number of ancestor documents that refer to both, and the number of descendant documents that both documents refer to.

In this paper, we propose a systematic method for extending retrieval techniques to include HTML structures. We first group subsets of HTML tags into a set of classes. Then when we index each document *d*, we assign the occurrence of each term *t* of *d* into one of these classes. In addition, the occurrences of term *t* in the anchor structure of other documents that have hyperlinks to *d* are assigned to an additional class, called the Anchor class. Next, we attempt to learn an optimal importance factor combination for the classes, by conducting extensive retrieval experiments. Finally, the best importance factor combination found is used to adjust the weights of terms and to compute the similarity of queries to documents. The proposed method provides us with the necessary flexibility to test the effectiveness of making use of HTML tags and hyperlinks to enhance retrieval. To carry out the experiments, we created a testbed with a collection of 4,596 HTML documents and ten queries. For each query, the set of relevant documents and the set of irrelevant documents were identified manually.

This paper has the following contributions. First, a testbed was created. The availability of such a testbed is essential to carry out performance studies in information retrieval. While there are standard testbeds for traditional information retrieval (for example the TREC collections), there is currently no standard testbed for web documents. Creating a testbed requires a lot of effort as relevant and irrelevant documents for each query need to be identified manually. Second, a systematic method was proposed for studying the effectiveness of using HTML tags and hyperlinks to enhance document retrieval in the WWW environment. Third, our experiments indicate that by storing structure information in the index, assigning optimal class importance values, and using the extended index and importance values, retrieval effectiveness is substantially improved.

The rest of the paper is organized as follows. In Section 2, we present Webor, which is the tool we built [Webor96] and used for this study. The classes used by Webor, its *indexing engine*, and its *search engine* are also briefly described. In Section 3, the testbed is described. More specifically, we discuss the document collection and the queries of the testbed, as well as the methodology for determining the sets of relevant and irrelevant documents for each query. In Section 4, we describes the experiments, the results obtained, and how the best importance factor combination is determined. Section 5 provides some conclusions and discusses future work.

2. Webor - A Search Tool Developed for this Research

The tool we developed and used in this research, Webor (Web-based search tool for Organization Retrieval) [Webor96] is based on the vector space model and uses the following Cosine formula for calculating the similarity between a query and a document [Salt83],

$$Sim(q,d) = \frac{\sum_{i=1}^{k} q_i w_i}{\sqrt{\sum_{i=1}^{k} q_i^2 \sum_{i=1}^{k} w_i^2}} \quad (1)$$

where k is the dimension of the vector space, w_i is the weight of the ith term in the document vector d

and q_i is the weight of the *i*th term in the query vector *q*.

Webor contains an *indexing engine* and a *search engine*. Before we can describe how Webor works we must describe how and why some HTML tags were grouped into classes.

2.1 The Classes

We have grouped HTML tags into the following six classes: Plain Text, Title, H1-H2, H3-H6, Strong, and Anchor. The terms in the Plain Text class are terms that do not appear in the text enclosed by the title, header, or emphasized structures of an HTML document (see Table 1).

Table 1: The Six Classes and associated HTML tags

Class Name	HTML tags
Anchor	A
H1-H2	H1, H2
H3-H6	H3, H4, H5, H6
Strong	STRONG, B, EM, I, U, DL, OL, UL
Title	TITLE
Plain Text	None of the above

The reason that we grouped HTML tags into six classes, instead of having a separate class for each type of tag, was to reduce the size of the index, the work needed to find an optimal importance factor combination, and to improve the efficiency of Webor. We will now explain how the six classes were selected. At this point in the research there is no guarantee that this grouping is optimal.

We have divided the HTML header tags into three classes: Title, H1-H2 and H3-H6. The tags in the Title, H1-H2 and H3-H6 classes are TITLE, H1 and H2, and H3, H4, H5, and H6, respectively. The reasoning behind using these three classes was as follows: The terms in a document's title provide information on what a document is about, and thus should belong to a single class. The terms in the H1 and H2 headers provide descriptions of the main structure and topics of a document and thus should be grouped into a second class. The terms in the H3, H4, H5, and H6 headers provide information about the more specific structure and topics of a document and hence should be grouped into a third class.

We have grouped the HTML list tags, and the strong, emphasized, bold, underscored, and italic tags into a single class called the Strong class. The idea here was that terms that are emphasized and terms which appear inside lists, are terms that the author perceived to be important to the contents of the document and thus should be grouped together.

The Anchor class includes all the terms, which occur in the anchor tag of hyperlinks to the document. The justification for including this class in the index is that this information provides additional knowledge about the main subject of the document and should be taken into consideration when a query and a document are matched.

2.2 The Indexing Engine

The index built by Webor consists of a web page index and a keyword index. The web page index contains information concerning the web pages, including their IDs and URLs. The keyword index is organized into an inverted file for efficient retrieval. For each collection term *t* Webor keeps the total number of web pages that have *t* (*df*) and an inverted list. The inverted list for *t* is a sequence of pairs. The first element in each pair is the ID of some web page *d*, and the second element is a *Term Frequency Vector*, *TFV*. *TFV* contains the frequency of occurrence of *t* in each class associated with *d*. The term frequency vector is $TFV = (tfv_1, tfv_2, tfv_3, tfv_4, tfv_5, tfv_6)$ where tfv_1, tfv_2, tfv_3, tfv_4, tfv_5, and tfv_6 are the term frequencies of *t* in the Plain Text, Strong, H3-H6, H1-H2, Anchor, and Title classes, respectively.

Webor parses each word *s* of a document *d*, checks that *s* is not a stop word, and stems it. The result of the stemming process on a non stopword *s* is the term *t*. When Webor encounters the term *t* for the first time in document *d*, it generates a term frequency vector *TFV* = (0, 0, 0, 0, 0, 0) for *t*, and then determines a class assignment for *t*. To determine the class assignment, Webor uses the following precedence order: TITLE tag > H1&H2 tags > H3&H4&H5&H6 tags > Strong tags > None of the above (see Table 1). This means that when a word is enclosed in the TITLE tag it is assigned to the Title class regardless of any other tag in which it is enclosed, and only terms which are not enclosed by any of the header or Strong tags are assigned to the Plain Text class. Next, based on the assigned class, Webor increments one of the counts: tfv_1, tfv_2, tfv_3, tfv_4 , and tfv_6.

Indexing the terms of the Anchor class is the last process performed by Webor because it needs

to collect all the anchor text in the collection with hyperlinks to document *d*. Webor uses another file to keep this information about the anchor text associated with *d*. After Webor visited and indexed all web pages in the collection, it then indexes this file for the Anchor class. Each occurrence of term *t* in anchor descriptions of hyperlinks to *d* is used to increment tfv_5.

2.3 The Search Engine

The *search engine* is a CGI (Common Gateway Interface) program, which takes a query from a Web user via an HTML form and returns to the user a ranked list of HTML hyperlinks. The query can be an AND, or an OR Boolean query, or a list of terms. The user can also input a weight for each term. In addition, users can limit the number of web pages returned to them.

To enable conducting retrieval experiments, Webor requires the user to provide the six class importance values, which will be used in the current experiment. These values are stored by Webor in the C*lass Importance Vector* $CIV = (civ_1, civ_2, civ_3, civ_4, civ_5, civ_6)$, where civ_i is the importance factor assigned to class *i* in the current experiment.

The *search engine* parses the query and uses the inverted file index built by the *indexing engine*, and the *CIV* to retrieve all web pages whose *TFV* values are not all 0 for at least one of the query terms.

To take advantage of the *TFV* information produced by the *indexing engine*, and to take into account the *CIV* provided for the experiment, we needed to modify the computation of the weight *w* of term *t* in document *d* which Webor uses in the computation of the Cosine similarity (see formula (1)). Webor uses the formula $w = (TFV \circ CIV) \cdot idf$ where the inner product of the two vectors, *TFV* and *CIV*, represents the importance of term *t* to document *d*, and *idf* is the inverse document frequency of the term in the collection. For calculating *idf*, Webor uses the commonly used formula *idf = ln(N/df)* [Salt83], where *N* is the number of documents in the collection, and *df* (*document frequency*) is the number of documents that contain the term.

Finally the *search engine* sorts the retrieved documents by nonincreasing similarity and produces a ranked list of hyperlinks to WWW pages which the user can access.

2.4 Normal Retrieval and Normal CIV

Note that when the $CIV = (1,1,1,1,0,1)$, the weight *w* of term *t* reduces to:

$$w = (TFV \circ (1,1,1,1,0,1)) \cdot idf$$
$$= (tfv_1 + tfv_2 + tfv_3 + tfv_4 + tfv_6) \cdot idf$$
$$= tf \cdot idf$$

So with this *CIV* the weight calculated by Webor for each term is $w = tf \cdot idf$. In this case, the retrieval results obtained by Webor are equal to those obtained by any vector space based IR system that ignores HTML tags, uses $tf \cdot idf$ to calculate term weights, and Cosine to calculate the similarity between a query and a document. We call this *CIV* the *Normal CIV* and the retrieval with Normal *CIV*, the *Normal Retrieval*. The Normal retrieval results are compared to the results of experiments with other *CIV*s, and used to show the percentage of improvement achieved by better *CIV*s.

3. The Testbed

3.1 The Document Collection and the Queries

The document collection of the testbed includes all WWW pages that belonged to Binghamton University at the end of 1996. Webor's indexing robot was run with the domain seed "binghamton.edu" and indexed 4,596 HTML documents. The average number of words in an HTML document was 309. Table 2 shows the total number of term occurrences that were assigned to each class, and the percentage of these term occurrences. Note that the classes with the highest percentages are Plain Text (79.8%), Strong (13.2%), and Anchor (2.8%).

Table 2: The Distribution of term occurrences among the six classes

Class	Terms	Percentage
Anchor Class	39,840	2.8 %
H1-H2 Class	21,232	1.5 %
H3-H6 Class	27,266	1.9 %
Plain Text Class	1,131,376	79.8 %
Strong Class	187,094	13.2 %
Title Class	10,955	0.8 %
Total	1,417,763	100 %

The 10 queries used for the experiment (see the left column of Table 3) are the typical short queries used by faculty and students to find information in a university environment. Some of the queries relate to administrative issues, such as "promotion guidelines", while other queries relate to subject matters such as "neural networks".

3.2 Relevant Document Identification

To find the relevant set of documents for a given query, we substituted the query with a set of other queries (see column 2 of Table 3), and used Webor to create an expanded set of retrieved documents. This expanded set was checked manually to determine the subset of relevant documents. New queries were generated by using OR with synonyms, adding AND queries, and by omitting less important terms from the original query. For example, the query, "handicapped student help", was expanded into the queries, "handicap OR disable" and "physical AND challenge". These two queries enable Webor to retrieve all documents which refer to anyone with a disability. Note that the words "help" and "students" were omitted. This method enabled our finding the subset of relevant documents in the document collection for each of the ten queries.

We determined that a web page is relevant to a query if the web page was about the topic of the query, or was a resource (for example, a list of hypertext links) for finding information on the topic of the query. Column 3 of Table 3, shows the number of documents relevant to each query.

4. The Experiments

A large number of experiments were conducted to find an optimal *CIV* and compute the improvement it provides. The evaluation is based on the recall-precision metric widely used in information retrieval. For a given query, when a set of documents is returned from the IR system, the *recall* is defined to be the ratio (number of relevant documents retrieved)/(number of relevant documents in the collection) and the *precision* is defined to be the ratio (number of relevant documents retrieved)/(number of retrieved documents). The best retrieval effectiveness is achieved when both recall and precision are equal to 1. However, in practice, this is unlikely to occur. Usually, when higher recall is achieved, the precision becomes lower.

For each experiment we conducted, the ten testbed queries were used to retrieve documents from the testbed document collection based on a given *CIV*. For each query the retrieval results were evaluated at 11 recall points, starting at 0, ending at 1, and using increments of 0.1 Then, the precision results of the ten testbed queries were averaged at the 11 recall points. Finally the 11 precision values are averaged into a single number which we call the 11-point average precision. Traditionally, this number is used to represent the effectiveness of an information retrieval system. In this study, we have added the 5-point average precision, computed by averaging the testbed queries' average precision at recall values of 0, 0.1, 0.2, 0.3, and 0.4 to provide additional important information about the effectiveness of the system. Because of the large and increasing number of web documents, web search tools often return a

Table 3: The Queries

Original Queries	Modified Queries	# of Relevant Docs
web-based retrieval	1. web-based OR retrieval 2. web AND search	15
neural network	neural AND network	26
master thesis in geology	Geology	3
prerequisite of algorithm	Algorithm	4
handicap student help	1. handicap OR disable 2. physical AND challenge	14
promotion guideline	Promotion	4
grievance committee	Grievance	17
laboratory in electrical engineering	1. electrical 2. laboratory	8
anthropology chairman	anthropology OR chairman	3
computer workshop	workshop OR seminar	16

very large number of retrieved documents. Unfortunately only a relatively small percentage of these documents are useful to the web searcher. To locate some good documents a searcher has to download and browse, or read a summary of a large number of these documents. Since the results are ranked the searcher usually starts with the first document retrieved by the system and then proceeds down the list until a sufficient number of good documents have been located. This is a very time consuming and frustrating process. Often web searchers do not need all good documents for a given query and are satisfied with finding a small number of good documents. This makes high precision for the top documents retrieved by the system very important. In other words, high precision at a lower level of recall is very useful in the WWW environment.

4.1 Retrieval with a Single Class

Our first set of experiments were to compare *Normal Retrieval* with retrieval based only on terms appearing in a single class using an importance factor of 1 (SC/1). As can be expected the results are inferior to those of *Normal Retrieval* (see Table 4). It is interesting to observe, however, what happens when just the Anchor class is used for retrieval. The 11-point average precision is only 5% below the *Normal Retrieval*, and is 33% better than the *Normal Retrieval* for the 5-point average precision. These results indicate the usefulness of the descriptions included in the Anchor class. It is also interesting to observe that the 5-point average precision is comparable to that of the 5-point *Normal Retrieval*, for the Strong class only and the Title class only experiments.

4.2 Finding a Good Importance Factor for one Class of a Normal CIV

To determine the effect of increasing the importance factor of a single class of a Normal *CIV*, we conducted experiments where a number of different importance factors were assigned to one class, while the rest of the factors remained as those of a Normal *CIV*. The importance factor of the plain text class was fixed at 1, and the experiments attempted to find a good (or optimal) value for each of the other classes. In this paper we only include the results obtained by 3 experiments conducted for the Anchor class (the results for the other classes are in [Shih97]).

The results of the experiments for assigning importance factor values 2, 4, and 6 to the Anchor class are shown in Table 5. Note that when the factor is either 2 or 6 the results are almost identical to those of the *Normal Retrieval*, but a factor of 4 gives a better average precision for both 5-point and 11-point average precisions. Compared to *Normal Retrieval* the improvement is 9% for 5-point average and 5% for 11-point average.

Table 6 shows the best results obtained by experiments that vary a single factor in the Normal *CIV*, for the Strong, H1-H2, Anchor and Title classes. It shows that by using a factor of 8 for the Strong class we get improvements of 10% and 7% in retrieval results over *Normal Retrieval*. The table does not include the H3-H6 class since in our experiments a Normal *CIV* with factors larger than 1 for the class H3-H6 does not provide better results.

Table 4: Improvement over Normal using SC/1 for retrieval

Class	*CIV*	5-Point Average Precision	11-Point Average Precision	5-Point Improvement over Normal	11-Point Improvement over Normal
Normal	111101	0.249	0.201		
Anchor only	000010	0.332	0.191	33%	-5%
H1-H2 only	000100	0.274	0.159	10%	-21%
H3-H6 only	001000	0.097	0.047	-61%	-76%
Plain Text only	100000	0.203	0.112	-19%	-44%
Strong only	010000	0.255	0.156	2%	-22%
Title only	000001	0.258	0.187	4%	-7%

Table 5: Improvement over Normal using Anchor factors 2, 4, and 6.

Class/Factor	*CIV*	5-Point Average Precision	11-Point Average Precision	5-Point Improvement Over Normal	11-Point Improvement over Normal
Normal	111101	0.249	0.201		
Anchor/2	111121	0.245	0.200	-1%	0%
Anchor/4	111141	0.271	0.211	9%	5%
Anchor/6	111161	0.245	0.199	-2%	-1%

Table 6: The Best Value of a Single Factor

Class	*CIV*	5-Point Average Precision	11-Point Average Precision	5-Point Improvement Over Normal	11-Point Improvement over Normal
Normal	111101	0.249	0.201		
Strong/8	181101	0.273	0.215	10%	7%
H1-H2/4	111401	0.274	0.213	10%	6%
Anchor/4	111141	0.271	0.211	9%	5%
Title/4	111104	0.272	0.213	9%	6%

4.3 Finding an Optimal CIV

Once a best factor value was determined for each single class in an otherwise Normal *CIV*, we tried the combination of all best pairs (with values greater than 1, i.e., the best pairs from the Strong, H1-H2, Anchor and Title classes) in an otherwise Normal *CIV*. Table 7 shows the improvement over Normal *CIV* achieved by using pairs of best factors in an otherwise Normal *CIV*.

Note that by using *CIV*s with best factor pairs for Strong&H1-H2, Strong&Anchor, and Strong&Title, the improvements are comparable to the sum of the improvements achieved with the best factors for the two corresponding individual classes. For example, when the single best factor for Strong is used the improvements are 10% and 7%, when the best factor for H1-H2 is used the improvements are 10% and 6%, and when both best factors for Strong and H1-H2 are used the improvements are 21%~10%+10% and 13%=7%+6%. The results indicate also that the Strong class is very important. Note that the improvement for the single best factor in the classes H1-H2, and Title are canceled when using the best pair for H1-H2&Title.

Other experiments conducted by us resulted in a better *CIV* = (181181) for the class pair Strong&Anchor in which the value of the Anchor factor was increased to 8. Additional experiments in which a single factor was changed in this *CIV* have not shown any further improvement.

We next experimented with changing the factors of the H1-H2 and Title classes of the *CIV* = (181181). The results are summarized in Table 8.

The best vector found was (181684), which improved the average precision over normal by 26% for the 11-point average precision, and by 44% for the 5-point average precision. Experiments with the effect of changing a single value in the *CIV* (181684) showed no improvements in the results. This was the best *CIV* that we have found.

4.4 Determining the Importance of Each Class

Another way to determine the importance of the terms in a given class, say C, is as follows. We first fix the importance factor for C to that as in the Normal *CIV*. Then we try to find the new best *CIV by* adjusting the importance factors for the other three classes (the importance factors for the plain text class and H3-H6 class are fixed at 1). If the retrieval effectiveness based on the new best *CIV* is about the same as that based on the old best *CIV* (i.e., (181684)), then this indicates that the terms in C are not very important for enhancing retrieval effectiveness. On the other hand, if the retrieval effectiveness based on the new best *CIV* is

substantially lower than that based on the old best *CIV*, then the terms in C are very important for improving retrieval effectiveness as adjusting the importance factors for the other classes alone can not achieve the same level of improvement. The results of this set of experiments are summarized in Table 9. From this table, it is clear that the Strong class and the Anchor class are very important while the H1-H2 class and Title class are less important.

5. Conclusions and Future Work

In this paper, we proposed a method for making use of the structures and hyperlinks of HTML documents to improve the effectiveness of retrieving HTML documents. Our method partitions the occurrences of terms into six classes (title, H1-H2, H3-H6, anchor, strong and plain text) and adjusts the traditional term weighting scheme by incorporating different importance factors to term occurrences in different classes. Through extensive experiments, we showed that by using our method it is possible to substantially improve the retrieval effectiveness (see Table 8). In particular, we found

Table 7: Improvement by using *CIV*s with Best Factor Pairs

	CIV	**5-Point Average Precision**	**11-Point Average Precision**	**5-Point Improvement Over Normal**	**11-Point Improvement over Normal**
Normal	111101	0.249	0.201		
Strong&H1-H2	181401	0.300	0.228	21%	13%
Strong&Anchor	181141	0.300	0.226	21%	13%
Strong&Title	181104	0.296	0.226	19%	13%
H1-H2&Anchor	111441	0.274	0.213	10%	6%
H1-H2&Title	111404	0.248	0.202	0%	0%
Anchor&Title	111144	0.268	0.210	8%	4%

Table 8: Improvement over Normal with *CIV*s (181181) and (181684)

CIV	**5-Point Average Precision**	**11-Point Average Precision**	**5-Point Improvement over Normal**	**11-Point Improvement over Normal**
Normal	0.249	0.201		
181181	0.353	0.251	42%	25%
181684	0.357	0.254	44%	26%

Table 9: The Improvement by each factor of the best *CIV*

Class with Factor fixed to Normal	***CIV***	**5-Point Improvement Over Normal**	**11-Point Improvement Over Normal**	**Contribution of factor in best *CIV* for 5-point**	**Contribution of factor in best *CIV* for 11-point**
Best *CIV*	181684	44%	26%		
Strong to 1	111644	11%	7%	33%	19%
H1-H2 to 1	181181	42%	25%	2%	1%
Anchor to 0	181604	21%	14%	23%	12%
Title to 1	181681	42%	25%	2%	1%

that the terms in the Strong and Anchor classes are the most useful for improving the retrieval effectiveness.

We plan to conduct more experiments using an expanded set of queries and possibly different web page collections. We believe that substantially more extensive experimental results need to be collected and analyzed in order to assess accurately the effectiveness of using HTML structures. Another issue to investigate is the optimal assignment of tagged information to classes. It is possible that information in lists should be in a different class from emphasized terms, or that all headers H1-H6 should be included in one class. In this study, the similarity function used is Cosine, and the term weight function is a modification of $tf \cdot idf$. Other similarity and term weight functions have also been used in traditional IR systems. We are interested in examining how different similarity and weight functions may affect the retrieval effectiveness. Studying the feedback process in the WWW environment is also of interest.

6. Availability

The information about Webor can be accessed at

http://nexus.data.binghamton.edu/~yungming/webordoc.html

It also includes links to download the testbed.

References

[Agost96] M. Agosti and A. Smeaton, *"Information Retrieval and Hypertext"* Kluwer Academic Publishers, 1996.

[Alta96] Digital Equipment Corporation, "*ALTA VISTA: Main Page*", http://altavista.digital.com/cgi-bin/query/, 1996.

[CMU95] Carnegie Mellon University, "*Lycos, The Catalog of the Internet*", http://lycos.cs.cmu.edu/, 1995.

[Croft93] W.B. Croft and H.R. Turtle, *"Retrieval strategies for hypertext"* Information Management and Processing 29(3), 1993, 313-324.

[Dunlop93] M. D. Dunlop and C. J. Van Rijsbergen "Hypermedia and free text retrieval", Information processing and Management 29(3), 1993, 287-292.

[Frei92] H.P. Frei, D. Stieger , "*Making Use of Hypertext Links when Retrieving Information*", Proceedings ACM ECHT'92, Milan, Italy, 1992, 102-111.

[Frisse88] M.E. Frisse, *"Searching for Information in a Hypertext Medical Handbook"* Communications of ACM 31(7) July 1988.

[HotBot] Inktomi, Inc., HotBot Home Page, http://www.hotbot.com/.

[Hypur96] R. Weiss, B. Velez, M.A. Sheldon, C. Namprempre, P. Szilagyi, A. Duda, and D.K. Gifford, *"HyPursuit: A Hierarchical Network Search Engine that Exploits Content-Link Hypertext Clustering"*, Proceedings of the Seventh ACM Conference on Hypertext, Washington, DC, March 1996.

[Mage97] The McKinley Group Inc., *"Magellan Internet Guide"*, http://www.mckinley.com/, 1997.

[Mau97] M.L. Mauldin, *"Lycos: Design choices in an Internet search service"*, IEEE Expert Online, February 1997.

[Mi97] Microsoft Co., *"microsoft.com Search Wizard"*, http://www.microsoft.com/search/default.asp, 1997.

[OTC96] OpenText Corporation, "*Search the World Wild Web – Every word every page*", http://www.opentext.com:8080/, 1996.

[Pi94] B. Pinkerton, "*Finding What People Want: Experiences with the WebCrawler*", Proceedings of the 2nd Int'l World Wide Web Conf., Elsevier Science, http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/WWW2_Proceedings.html, 1994.

[Salt83] G. Salton and M.J. McGill, *"Introduction to Modern Information Retrieval"*, McGraw-Hill, New York,

NY, 1983.

[Salt89]. G. Salton, *"Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer"*, Addison Wesley, 1989.

[Shih97] Y. Shih, "A Study of the Usefulness of the Structure of HTML Documents on Retrieval Effectiveness", MS thesis, Dept. of Computer Science, State of New York at Binghamton, 1997.

[Webor96] J. Lu, Y. Shih, W. Meng, and. M. Cutler, "*Web-based search tool for Organization Retrieval*", http://nexus.data.binghamton.edu/~yungming/webor.html, 1996.

[Yaho96] Yahoo Inc., "*Yahoo Search*", http://www.yahoo.com/search.html, 1996.

[YuLe96a] B. Yuwono and D.L. Lee, *"Search and Ranking Algorithms for Locating Resources on the World Wide Web"*, Proceedings of the 12th International Conference on Data Engineering, New Orleans, Louisiana, Feb. 26 - March 1, 1996, 164-171.

[YuLe96b] B. Yuwono and D.L. Lee, "*WISE: A World Wide Web Resource Database System"* IEEE Transactions on Knowledge and Data Engineering, Special Section on Digital Library, Vol. 8, No. 4, Aug 1996, 548-554.