A New Study on Using *HTML* Structures to Improve Retrieval

M. Cutler, H. Deng, S. S. Maniccam, and W. Meng
Department of Computer Science
State University of New York at Binghamton
Binghamton, NY 13902
cutler@binghamton.edu

Abstract

Locating useful information effectively from the World Wide Web (WWW) is of wide interest. This paper presents new results on a methodology of using the structures and hyperlinks of HTML documents to improve the effectiveness of retrieving HTML documents. This methodology partitions the occurrences of terms in a document collection into classes according to the tags in which a particular term appears (*such as Title, H1-H6, and Anchor*)*. The rationale is that terms appearing in different structures of a document may have different significance in identifying the document. The weighting schemes of traditional information retrieval were extended to include class importance values. We implemented a genetic algorithm to determine a "best so far" class importance factor combination. Our experiments indicate that using this technique the retrieval effectiveness can be improved by* 39.6% *or higher.*

1. Introduction

The World Wide Web has become an important information resource today. Many search tools have been created to help ordinary users find useful information from the Web. A search tool typically consists of two components. One is a robot-based indexing engine which recursively downloads web pages, indexes their contents, extracts their URLs 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 [5, 6]. This model represents each document as a vector of term weights. These weights indicate the significance of the term in identifying the contents of a document. The weight of a term is usually computed from two factors. The first is term frequency, *tf*, which is the number of its occurrences in a document. The second is the document frequency, *df*, which is the number of documents in the collection that contain it. *idf* is the inverse document frequency of the term in the collection. A commonly used formula For calculating *idf* is $idf = log(N/df)$ [5], where N is the number of documents in the collection. A commonly used formula for computing weights is $tf \cdot idf$. Each user query is also represented as a vector. Different similarity functions are used to compute the closeness (or similarity) of a query and a document.

The following two observations about the differences between WWW documents and those used in traditional IR systems motivate this research:

1. The structures of *HTML* documents are easily available through *HTML* tags. Such structures provide information about the content of a document and can be useful in improving the retrieval effectiveness. Traditional IR systems typically disregard structure information.
2. *HTML* collections contain additional information about each document that has hyperlinks to it. Typically, authors include in the Anchor tag a description in addition to a URL. These descriptions have the potential of being very important for retrieval since they include the perception of these authors about the contents of the document. So by adding the anchor information of *HTML* documents to the index we may be able to improve retrieval. Traditional IR uses only terms included in a document.

A preliminary study is described in [1]. It had several shortcomings. First, the set of queries used was too small. Second, the formation of the six classes was not well justified. Third, the existence of duplicates among the web pages used in the test could lead to inaccurate results. Fourth, no systematic method was used to find the optimal class importance vector. This paper reports

a study that overcomes all the weaknesses in our previous work. First, a set of 10 new queries was created for this study. As a result, experiments could be carried out based on the 10 old queries, the 10 new queries and the combined 20 queries. Second, six new classes that better reflect the intended usage of different *HTML* tags were used to replace the old six classes. Third, duplicate pages were removed. Finally, a genetic algorithm was implemented to systematically search for the optimal class importance vector. The new experiments indicate that our approach can improve the 11-point average precision by about 40%.

The rest of the paper is organized as follows. In Section 2, we present Webor, which is the tool we built [7] and used for this study. In Section 3, the testbeds are described. In Section 4, we describe the experiments, the genetic algorithm and the results obtained. Section 5 provides some conclusions and discusses future work.

2. The research tool - Webor

The tool we developed and used in this research, Webor (Web-based search tool for Organization Retrieval) [7] is based on the vector space model and uses the Cosine formula for calculating the similarity of a query to a document [5]. 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 grouped most frequently used *HTML* tags into the following six classes: Plain Text, Strong, List, Header, Anchor and Title (See Table 1). The rationale for using the above six classes is given below:

1. The Title class - terms in a document's title provide information on what a document is about.
2. The Header class - terms in the headers intend to provide descriptions of the main structure and topics of a document.
3. The List class - *HTML* list tags are often used to provide semantic information such as the subtopics of a more general subject, the parts of an object, etc,.
4. The Strong class - authors emphasize key concepts by using the strong, emphasized, bold, underscored, and italic tags.
5. The Anchor class of a document contains all the term occurrences that appear in anchor tags of hyperlinks that point to it. Note that a document's Anchor class contains term occurrences from *other* documents. This class is included in the index since it provides additional knowledge about the main subject of the document. WWWW [4] and Google [2] also used terms in anchor tags to index a referenced page. The difference is that we treat the Anchor class as one of six classes for which their relative importance will be determined. Previous approaches do not have this context.
6. The Plain Text class - includes the rest of the terms.

HTML tags may be nested in an *HTML* page. To solve this problem, the following precedence order is used: Title tag > Header tags > Strong tags > List tags > None of the above.

Table 1: The classes and associated tags

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

2.2 The indexing engine

The index built by Webor consists of web page and keyword indexes. The web page index contains IDs and URLs. The keyword index is an inverted file. For each term *t* Webor keeps the total number of web pages that have *t* (*df*) and an inverted list. The inverted list is a sequence of pairs. The first element in each pair is a page *ID*, and the second element is a *Term Frequency Vector*, *TFV*. *TFV* contains the frequency of occurrence of *t* in that page for each class. The term frequency vector is $(tfv_1, tfv_2, tfv_3, tfv_4, tfv_5, tfv_6)$ 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, List, Header, Anchor and Title classes, respectively.

Webor parses each nonstop word of a document and stems it. Then, it assigns the occurrence of the stem to a class and increments the class's count. During indexing Webor collects all the anchor text. After Webor finished downloading and indexing the collection, it indexes the collected anchor text. Each occurrence of a term in anchor description of document *d* is used to increment tfv_5 of the term in *d*.

2.3 The search engine

The search engine is a CGI (Common Gateway Interface) program that takes a query via an HTML form and returns a ranked list of HTML hyperlinks. The query can be an AND/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 pages returned to them. For each experiment, Webor needs the *Class 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 index built by the *indexing engine*, to retrieve all web pages which contain at least one of the query terms. Webor uses the formula $w = (TFV \bullet CIV)\ idf$, where the inner product of the two vectors $TFV \bullet CIV$ represents the importance of term *t* to document *d*, and *idf* is the inverse document frequency of 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

When $CIV = (1, 1, 1, 1, 0, 1)$ $TFV \bullet CIV = TFV \bullet (1, 1, 1, 1, 0, 1) = (tfv_1 + tfv_2 + tfv_3 + tfv_4 + tfv_6) = tf$. So with this *CIV* the retrieval results obtained by Webor are equal to those obtained by any vector space based IR system that ignores *HTML* tags, and uses $tf \cdot idf$ and Cosine. We call retrieval with this *CIV*, *normal retrieval*. The normal retrieval results are compared to the results of experiments with other *CIVs*, and used to show the percentage of improvement achieved by better *CIV*s.

3. The testbed

3.1 Document collection and queries

Table 2: Term occurrences among the classes

Class	Terms	Percentage
Anchor Class	50,094	5.3 %
Header Class	46,314	4.9 %
List Class	22,684	2 .4%
Plain Text Class	723,060	76.5 %
Strong Class	91,684	9.7 %
Title Class	11,342	1.2 %
Total	945,176	100 %

The document collection of the testbed includes all WWW pages that belonged to Binghamton University at the end of 1996. There were 3,040 distinct pages. 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, Strong, and Anchor.

Table 3: The old queries

Original Queries	Modified Queries	# Rel.
web-based retrieval	1. web-based OR retrieval 2. web AND search	15
neural network	neural AND network	20
master thesis in geology	geology	2
prerequisite of algorithm	algorithm	3
handicap student help	1. handicap OR disable 2. physical AND challenge	8
promotion guideline	promotion	4
grievance committee	grievance	12
laboratory in electrical engineering	1. electrical 2. laboratory	8
anthropology chairman	anthropology OR chairman	3
computer workshop	workshop OR seminar	15

In our previous study, we created 10 (old) queries for our experiments (see Table 3). In this study, 10 (new) queries were created (see Table 4). This enabled experimenting with the old, new and combined 20 queries.

Table 4: The new queries

Original Query	Modified Query	# Rel.
concert and music	concert OR music	13
intramural sports	sports	9
cognitive science	cognitive OR cognition	5
campus dining	dining	3
career development	career	22
research centers	centers	15
engineering program	engineering	21
papers in philosophy and computer and cognitive system	papers	14
student associations	associations	6
non-matriculated admissions	admissions	11

3.2 Relevant document identification

To find the relevant set of documents for a given query, we substituted the query with a set of more general queries (see Table 3 and 4), and used Webor to create an expanded set of retrieved documents. This expanded set was checked manually to determine the subset of relevant documents. We determined that a web page is relevant to a query when it was *about* the topic of the query, or a *resource* (for example, a list of hypertext links) for finding information on the topic of the query. Column 3 of Table 3 and 4 shows the number of documents relevant to each query.

4. The experiments

Experiments were conducted to find an optimal *CIV* and compute the improvement it provides over the normal retrieval for each of three sets of test queries (old, new and combined). The evaluation was based on the 11-point average-precision metric widely used in information retrieval [5, 6].

In order to carry out the experiments in a more systematic manner, we implemented a genetic algorithm [3] for finding the best *CIV*. The evolutionary programming algorithm starts with a random population of *CIV*s. The value of each vector component is a small non-negative integer (in the range 1 to 15). The initial population had α (30) *CIV*s. 25 *CIV*s were generated randomly, and 5 were chosen from the good *CIV*s found during the previous study from manual experiments. The genetic algorithm terminates after β (25) generations were evolved, and the fittest *CIV* of all generations is selected as the "optimal" *CIV*.

The fitness of a *CIV* is computed as follows. A *CIV* has an *initial* fitness of 0 when the 11-point average precision computed by Webor with this *CIV* is less than γ (.22). Otherwise the initial fitness is equal to (the 11-point average precision - γ). The *final* fitness of a member is its initial fitness divided by the sum of the initial fitness of all the *CIV*s in the current generation. Once the fitness of every member in the current population is known the population can evolve and the next generation is created. In each generation the following three processes are executed sequentially: reproduction, crossover and mutation.

Reproduction uses a wheel of fortune scheme, to select *CIV*s in the current generation to be the *parent* population. The scheme selects fit *CIV*s with high probability and unfit *CIV*s with low probability. The same *CIV* may be selected more than once. Crossover is done for each consecutive pair of *CIV*s in the parent population, with probability δ (0.75). When crossover is not done for a pair of *CIV*s they are added unchanged to the next generation. Otherwise, a randomly generated binary mask with the same number of components as the *CIV* is created. Each mask is used to generate a *child* from a pair of parents. The binary values, zero or one, in each mask are used to select the value of a *CIV* component from either the first or the second parent, respectively. Mutation is performed on each child *CIV* with probability ε (0.1). When mutation is performed, each *CIV* component is either decreased or increased by one with equal probability.

The experimental results were as follows. For the old, new, and combined sets, the 11-point average precision for the normal retrieval is 0.182, 0.172 and 0.177, respectively. The best *CIVs* found by the genetic algorithm were (181882), (181782), and (181582). For these *CIVs*, the 11-point average precision was 0.254, 0.255, and 0.254. The improvements were 39.6%, 48.3%, and 43.5%.

The following observations can be made from the above results. First, relative to the plain text class, the Strong and Anchor classes are the most important class. Second, by appropriately assigning importance factors to different classes, the retrieval performance can be significantly improved.

5. Conclusions and future work

We proposed a method for making use of the structures and hyperlinks of *HTML* documents to improve the effectiveness of retrieving *HTML* documents. Using a genetic algorithm, we showed that with our method it is possible to substantially improve the retrieval effectiveness. In particular, we found that the terms in the Strong and Anchor classes are the most useful for improving the retrieval effectiveness. To further validate our results we plan to conduct more experiments using an expanded set of queries and possibly different web page collections. 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.

References

[1] M. Cutler, Y. Shih, and W. Meng. "*Using the Structure of HTML Documents to Improve Retrieval*", USENIX Symposium on Internet Technologies and Systems, 1997.

[2] S. Brin, and L. Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine", WWW7 Conference, 1998.

[3] D. Goldberg. "*Genetic Algorithms in Search, Optimization and Machine Learning*", Addison Wesley, 1989.

[4] O. A. McBryan. "*GENVL and WWWW: Tools for Taming the Web*". WWW1 Conference, 1994.

[5] G. Salton and M. J. McGill, *"Introduction to Modern Information Retrieval",* McGraw-Hill, New York, NY, 1983.

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

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