Improvement of HITS-based Algorithms on Web Documents *

Longzhuang Li
Dept. of Computer Engr. and Computer Sci.
Uni. of Missouri-Columbia
Columbia, MO 65211
ll059@mizzou.edu

Yi Shang
Dept. of Computer Engr. and Computer Sci.
Uni. of Missouri-Columbia
Columbia, MO 65211
yshang@cecs.missouri.edu

Wei Zhang
Dept. of Computer Engr. and Computer Sci.
Uni. of Missouri-Columbia
Columbia, MO 65211
wz206@mizzou.edu

ABSTRACT

In this paper, we present two ways to improve the precision of HITS-based algorithms on Web documents. First, by analyzing the limitations of current HITS-based algorithms, we propose a new weighted HITS-based method that assigns appropriate weights to in-links of root documents. Then, we combine content analysis with HITS-based algorithms and study the effects of four representative relevance scoring methods, **VSM, Okapi, TLS**, and **CDR**, using a set of broad topic queries. Our experimental results show that our weighted HITS-based method performs significantly better than Bharat's improved HITS algorithm. When we combine our weighted HITS-based method or Bharat's HITS algorithm with any of the four relevance scoring methods, the combined methods are only marginally better than our weighted HITS-based method. Between the four relevance-scoring methods, there is no significant quality difference when they are combined with a HITS-based algorithm.

Categories and Subject Descriptors

H.3 [**Information Systems**]: Information Storage and Retrieval; G.3 [**Mathematics of Computing**]: Probability and Statistics; D.2 [**Software**]: Software Engineering

General Terms

Algorithms, Measurement, Performance

Keywords

HITS-based algorithms, relevance scoring methods, information retrieval

1. INTRODUCTION

Kleinberg's hypertext-induced topic selection (HITS) algorithm [16] is a very popular and effective algorithm to rank documents based on the link information among a set of documents. The algorithm presumes that a good hub is a document that points to many others, and a good authority is a document that many documents point to. Hubs and authorities exhibit a *mutually reinforcing relationship*: a better hub points to many good authorities, and a better authority is pointed to by many good hubs. To run the algorithm, we need to collect a base set, including a root set and its neighborhood, the in- and out-links of a document in the root set.

Because the HITS algorithm ranks documents only depending on the in-degree and out-degree of links, it will cause problems in some cases. For example, Bharat [1] identified two problems: *mutually reinforcing relationships between hosts* and *topic drift*. Both problems can be solved or alleviated by adding weights to documents. The first problem can be solved by giving the documents from the same host much less weight [1], and the second problem can be alleviated by adding weights to edges based on text in the documents or their anchors [1, 2]. Bharat [1] showed that the simple modification of HITS algorithm for the first problem achieved a remarkable better precision, while further precision can be obtained by adding content analysis. Furthermore, by manipulating the weights of documents, Chang [3] created customized authority by adding more weights to the documents that users are interested in. Farahat [10] improved the precision through Web log records which store the number of users' visits to the documents.

Changing the weight of a document has been shown to be an effective way to improve the precision of HITS algorithm, but these methods need either content analysis [1, 2], user feedback [3], or Web log records [10]. As we know, content analysis usually takes a long time, and it is almost impossible to get users' feedback or visiting times for most Web documents. In the paper, we propose a new way to improve HITS-based algorithms' precision by only considering the link information in the base set.

Disregarding the time it may take, combining connectivity and content analysis has been proven to be useful in improving precision. But the similarity measure currently used is vector space model [1] or just a simple occurrence frequency of the query words in the text around the anchors [2], which may not be the best method to evaluate the relevance of Web documents because most queries submitted to search engines are short, consisting of three terms or less [15]. Although we can expand the short queries by adding more related words, expanding itself can cause *topic drift*.

In this paper, we statistically compare the performance of four relevance scoring methods when they are combined with HITS-based algorithms. Three of them are variations of

*Research supported in part by the National Science Foundation under grants DUE-9980375 and EIA-0086230.

WWW2002, May 7–11, 2002, Honolulu, Hawaii, USA.
ACM 1-58113-449-5/02/0005.

methods widely used in the traditional information retrieval field. They are cover density ranking (CDR) [7], Okapi similarity measurement (Okapi) [13], and vector space model (VSM) [21]. In addition, the fourth one is called three-level scoring method (TLS) [19], which mimics commonly used manual similarity measuring approaches. In comparing the performance of the four relevance scoring methods, we apply two statistical metrics: *expected loss* [4] and *probably approximately correct* [5].

This paper is organized as follows. In Section 2, we discuss the limitation of current HITS-based algorithms. In Section 3, we propose a new method to improve the HITS algorithm by assigning appropriate weights to in-links of documents in the root set. In Section 4, we present the ways of combining four different relevance scoring methods with HITS-based algorithms. In Section 5, we show the experimental results. Finally, in Section 6, we summarize the paper.

2. CURRENT HITS-BASED ALGORITHMS AND THEIR LIMITATIONS

In the context of Web search, a HITS-based algorithm first collects a base document set for each query. Then it recursively calculates the hub and authority values for each document. To gather the base document set I, first, a root set R that matching the query is fetched from a search engine; then, for each document $r \in R$, a set of documents L that point to r and another set of documents L' that are pointed to by r are added to the set I as R's neighborhood. For a document $i \in I$, let a_i and h_i be the authority and hub values respectively. To begin the algorithm, a_i and h_i are initialized to 1. While the values have not converged, the algorithm iteratively proceeds as follows:

1. For all $i' \in I$ which points to i,
$$a_i = \sum_{i'} h_{i'} \quad (1)$$
2. For all $i' \in I$ which is pointed to by i,
$$h_i = \sum_{i'} a_{i'} \quad (2)$$
3. Normalize a_i and h_i values so that $\sum_i a_i = \sum_i h_i = 1$.

Kleinberg showed that the algorithm will eventually converge, but the bound on the number of iterations is unknown. In practice, the algorithm converges quickly.

Bharat [1] improved Kleinberg's HITS algorithm by giving a document an authority weight of $1/k$ if the document is in a group of k documents on a first host which link to a single document on a second host, and a hub weight of $1/l$ if there are l links from the document on a first host to a set of documents on a second host. In the following discussion, we denote Bharat's improved HITS algorithm as BHITS algorithm.

Although BHITS generally works well, there are cases where it generates bad results. Let's use a simple query *cruises* to illustrate the problem. The top half of Table 1 shows the top 10 authorities and top 10 hubs returned by BHITS. All the top 10 authorities are from the out-links of a root link *www.cyprus-cruises.com*, which has only 1 in-link but 271 out-links, most of which are in different domains, while the average in-degree and out-degree of a root set link are 34 and 17 respectively. Top 3 to top 10 hubs are from another root link *cyprus-car-hire.com*, which is also an out-link of the root link *www.cyprus-cruises.com*. Actually, the two root links and their neighborhood form a tightly-knit community (TKC) [18]. After several iterations of BHITS algorithm, the hub value of the root link *www.cyprus-cruises.com* dominates all the others, as a result, the documents it points to get the largest authority values. Consequently, the in-links of root link *cyprus-car-hire.com* get the top hub values because it is assigned the largest authority value. After we manually visited these top 10 authorities and top 10 hubs, we found most of the links are of very low relevance to the query *cruises*.

The major problem of the above example is that the root link *www.cyprus-cruises.com* has few in-links but a large number of out-links, most of which are not very relevant to the query. Let's call such a root link *small-in-large-out* link. Usually, the average in-degree and out-degree of a root link are much smaller than the out-degree of a *small-in-large-out* link.

3. A NEW WEIGHTED HITS-BASED ALGORITHM

Without content analysis, both BHITS and HITS can not solve the above problem. BHITS only gives the links on the same host/document small weights, whose sum equals to 1. While in the above problem, most of the out-links of a *small-in-large-out* link are in different domains. HITS usually makes the problem worse because it gives the same weight to all the documents in the base set. In this section, without resorting to content analysis, user feedback, or Web log records, to prevent the BHITS algorithm and HITS algorithm from converging to such *small-in-large-out* links, we use the link information in the base set, and add more weights to the in-links of root set links if a *small-in-large-out* link exists. In the new method, the equations (1) and (2) are modified as follows:

1. For all $i' \in I$ which points to i,
$$a_i = \sum_{i'} w_{a_{i'}} \cdot h_{i'} \quad (3)$$
2. For all $i' \in I$ which is pointed to by i,
$$h_i = \sum_{i'} w_{h_{i'}} \cdot a_{i'} \quad (4)$$

In equation (3), the value of $h_{i'}$ can be HITS hub weight, BHITS hub weight, or the hub weight of other HITS-based algorithms. Similarly, this applies to the value of $a_{i'}$ in equation (4).

In our current implementation, all the $w_{h_{i'}}$ values are set to 1. The setting of $w_{a_{i'}}$ consists of two parts:

1. Before starting a HITS-based algorithm, if there exists a root link whose in-degree is among the three smallest ones and whose out-degree is among the three largest ones, then set $w_{a_{i'}}$ to 4 for in-links of all the root links.
2. Otherwise, set all $w_{a_{i'}}$ to 1. Run the HITS-based algorithm for one iteration without normalization. If there exists a root link whose authority value is among the three smallest ones and whose hub value is among the

Table 1: Top 10 authorities and top 10 hubs obtained by BHITS algorithm and WBHITS algorithm for query *cruises*, respectively. We ignore the prefix "http://" for each link.

Top 10 Auth. by BHITS	Top 10 hubs by BHITS
www.cyprus-property.net	www.cyprus-cruises.com
www.cyprus-hotels.com	cyprus-car-hire.com
www.windowoncyprus.com	cruisecyprus.com
www.cyprus-villas.com	www.sunnycyprus.com/links/transport.html
cruisecyprus.com	www.armata.net/banner_advertising_opportunities.htm
www.fly-2-cyprus.com	www.caribbean-media.com/amex_why.htm
cyprus-car-hire.com	www.cyprus-hotels.com/ayia_napa.htm
www.cyplon.com	www.cyprus-holidays.com/cyprus.htm
www.benacon.com	windowsoncyprus.com/contacts_for_window_on_cyprus.htm
www.yacht-sale.com	sybar.com/d/Autos/autos.html
Top 10 Auth. by WBHITS	**Top 10 Hubs by WBHITS**
www.costacruises.com	www.wxusa.com/Travel
www.rssc.com	www.cruiseplanners-acruz4u.com/cruises.html
www.cruisehawaii.com	www.cruisesonly.net/CLIAlinks.htm
www.regalcruises.com	www.anchorsaweigh.com/destinations.html
www.princesscruises.com	stablescruise.cruisesonly.net/CLIAlinks.htm
www.first-european.com	www.cruisespecials.com/newlinkspage.htm
www.silversea.com	www.caribbean-on-line.com/cruise-lines
www.princess.com	www.justcruising.org/cruise_lines.htm
www.royalolympiccruises.com	www.all-global.com/cruiselineslinks.htm
www.rivercruises.com	www.tradewinds-travel.net/cruises.htm

three largest ones, set $w_{a_{i'}}$ to 4 for in-links of all the root links.

In the above two steps, usually the in-degree of a *small-in-large-out* link is as small as 0, 1, or 2, while the out-degree can be more than several hundred. Intuitively, in most cases, it is hard to believe that a root link with no or few in-links can point to many highly relevant documents. Even if it points to many good documents, due to the large number of documents in the base set, there may be some duplicates between the out-links of the *small-in-large-out* link and the neighborhood of other links, and these good duplicated documents still have the chance to top the hub set or the authority set. The method of setting in-link weights are very simple and can be further improved by adaptively changing the weights of both in- and out-links of a *small-in-large-out* link.

Let's use WBHITS to represent the weighted BHITS algorithm. The bottom half of Table 1 shows the top 10 authorities and top 10 hits from WBHITS, which are much better than those from BHITS.

4. COMBINING THE HITS-BASED ALGORITHMS WITH RELEVANCE SCORING METHODS

The hub value and authority value are regulated in the way proposed in [1]. If s_i is the relevance score of a Web page i and h_i the hub value, $s_i \cdot h_i$ instead of h_i is used to compute the authority values of Web pages it points to. Similarly, if a_i is its authority value, $s_i \cdot a_i$ instead of a_i is used to compute the hub values of Web pages that point to it.

We combine four relevance scoring methods, **VSM**, **TLS**, **Okapi**, and **CDR**, with a HITS-based algorithm, respectively. In **VSM**, cosine normalization is used to normalize the relevance scores to range [0,1]. For popular topic, the relevance scores of the single term query, such as *blues*, will be 1 for almost all the Web pages using both **VSM** and **TLS**. In this case, there is almost no difference between the HITS-based algorithm and the combination the HITS-based algorithm with **VSM** or **TLS**. Next, we present the four relevance scoring methods and how they are applied to the Web-based content analysis.

4.1 Vector Space Model (VSM)

The vector space model has been widely used in the traditional IR field [11, 12]. Most search engines also use similarity measures based on this model to rank Web documents. The model creates a space in which both documents and queries are represented by vectors. For a fixed collection of documents, an m-dimensional vector is generated for each document and each query from sets of terms with associated weights, where m is the number of unique terms in the document collection. Then, a vector similarity function, such as the *inner product*, can be used to compute the similarity between a document and a query.

In **VSM**, weights associated with the terms are calculated based on the following two numbers:

- term frequency, f_{ij}, the number of occurrence of term y_j in document x_i; and
- inverse document frequency, $g_j = \log(N/d_j)$, where N is the total number of documents in the collection and d_j is the number of documents containing term y_j.

The similarity $sim_{vs}(q, x_i)$, between a query q and a document x_i, can be defined as the inner product of the query

vector Q and the document vector X_i:

$$sim_{vs}(q, x_i) = Q \cdot X_i = \frac{\sum_{j=1}^{m} v_j \cdot w_{ij}}{\sqrt{\sum_{j=1}^{m}(v_j)^2 \cdot \sum_{j=1}^{m}(w_{ij})^2}} \quad (5)$$

where m is the number of unique terms in the document collection. Document weight w_{ij} and query weight v_j are

$$w_{ij} = f_{ij} w_{ij} = f_{ij} \cdot \log(N/d_j) \qquad \text{and}$$

$$v_j = \begin{cases} \log(N/d_j) & y_j \text{ is a term in } q \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Cosine normalization is used to normalize the similarity weight although queries are short in our experiments.

Due to the dynamic nature of the Web and inability to access the whole Web, the **VSM** method cannot be applied directly in evaluating the precision of search engines [20]. In Equation (5), the inverse document frequency g_j is not available because N and d_j are often unknown for the documents on the Web. There are three ways to deal with this problem: (1) making simple assumptions such as g_j is a constant for all the documents [22]; (2) estimating parameter values by sampling the Web [1]; and (3) using information from search engines, or/and experts' estimation. In this paper, we use a combination of the second and third approaches.

To apply **VSM** to the Web, we treat the Web as a big database and estimate the parameter values of **VSM**. To estimate N, we use the coverage of *Google* because *Google* is the search engine with the largest size, which contains about 1 billion pages in August, 2001 as reported in [24]. But because Google uses link data, it can actually achieve a larger coverage of over 1.3 billion pages. Thus, we set $N = 1,387,000,000$ which is the number of pages in *Google*'s index. To estimate d_j, the number of documents containing a certain term y_j, we use information extracted from several search engines as follows:

1. Submit the term y_j to several search engines (we use five search engines in our experiments). Each search engine i returns α_i, the number of documents containing the term.

2. Calculate the normalized value γ_i for each search engine i, based on α_i and the relative size (β_i) of the search engine to that of the whole Web: $\gamma_i = \alpha_i/\beta_i$. The sizes of the five search engines, *AltaVista, Fast, Google, HotBot*, and *NorthernLight*, used in our experiments are 550, 625, 1000, 500, and 350 million documents, respectively, as reported in previous studies [24]. Since the total size of the Web is estimated to be about 1.387 billion documents, the relative sizes (β_i) of these search engines are 0.397, 0.451, 0.721, 0.360, and 0.252, respectively.

3. Take the median of the normalized values of the search engines as the final result.

After getting the values of N and d_j, sim_{vs} can be easily computed.

4.2 Okapi Similarity Measurement (Okapi)

Okapi similarity measurement is one of the most popular methods used in the traditional IR field. Unlike **VSM**, the **Okapi** method not only considers the frequency of the query terms, but also the average length of the whole collection and the length of the document under evaluation. In the **Okapi** method, the similarity between a query q and a document x_i, $sim_o(q, x_i)$ can also be described as the inner product of the query vector Q and the document vector X_i as follows [13, 23]:

$$sim_o(q, x_i) = Q \cdot X_i = \sum_{j=1}^{m} v_j \cdot w_{ij} \quad (7)$$

where m is the number of unique terms in the document collection; v_j is the frequency of a term y_j in the query q; and w_{ij} is the document weight:

$$w_{ij} = \frac{f_{ij} \cdot log(\frac{N - d_j + 0.5}{d_j + 0.5})}{2 \cdot (0.25 + 0.75 \cdot \frac{dl}{avdl}) + f_{ij}} \quad (8)$$

where f_{ij} is the term frequency of a term y_j in the document x_i; N is the total number of documents in the collection; d_j is the number of documents in the collection that contain the query term y_j; dl is the length of the document (in bytes); and $avdl$ is the average document length in the collection (in bytes).

For reasons similar to the **VSM** method, the **Okapi** similarity measurement cannot be applied directly in evaluating the precision of search engines [20]. We need values for N, d_j, and $avdl$. In our research, we estimate the values of N and d_j in the way described in the last section for **VSM**. In addition, the average length of a Web document ($avdl$) is estimated as to be 10,939 bytes after removing all the HTML tags and Java scripts.

4.3 Cover Density Ranking (CDR)

Instead of computing the relevance based on term appearance, such as **VSM**, other methods including **CDR** are based on the appearance of phrases. **CDR** is developed to meet user expectation better – a document containing most or all of the query terms should be ranked higher than a document containing fewer terms, regardless of the frequency of term occurrence [25]. In **CDR**, the results of phrase queries are ranked in the following two steps [7]:

1. Documents containing one or more query terms are ranked by coordination level, i.e., a document with a larger number of distinct query terms ranks higher. The documents are thus sorted into groups according to the number of distinct query terms each contains, with the initial ranking given to each document based on the group in which it appears.

2. The documents at each coordination level are ranked to produce the overall ranking. The score of the cover set $\omega = \{(p_1, q_1), (p_2, q_2), \ldots, (p_n, q_n)\}$ is calculated as follows:

$$S(\omega) = \sum_{j=1}^{n} I(p_j, q_j) \qquad \text{and}$$

$$I(p_j, q_j) = \begin{cases} \frac{\lambda}{q_j - p_j + 1} & \text{if } q_j - p_j + 1 > \lambda \\ 1 & \text{otherwise} \end{cases} \quad (9)$$

where (p_j, q_j) is an ordered pair over a document, called *cover*, specifying the shortest interval of two distinct terms in the document [7]. p_j is the position of

Table 2: The 10 HITS-based algorithms used in our experiments

Alg.	Description	Alg.	Description
B	BHITS	WB	WBHITS
CB	Combination of BHITS and CDR	WCB	Combination of WBHITS and CDR
OB	Combination of BHITS and Okapi	WOB	Combination of WBHITS and Okapi
TB	Combination of BHITS and TLS	WTB	Combination of WBHITS and TLS
VB	Combination of BHITS and VSM	WVB	Combination of WBHITS and VSM

one term, q_j the position of another term, and q_j is assumed to be larger than p_j. λ is a constant and is set to 16 in our experiments because it has been shown to produce good results [6]. Covers of length λ or shorter are given score 1, and longer covers are assigned scores less than 1 in proportional to the inverse of their lengths.

To adapt **CDR** to the Web, we need to find out how many distinct query terms a document has and rank the documents with more distinct terms higher. Our version of **CDR** method computes the relevance scores of documents in two steps:

1. Documents are scored according to the regular **CDR** method. Each document belongs to a coordination level group and has a score within that group.

2. The scores are normalized to range (0, 1] for documents containing only one term, to range (1, 2] for documents containing two different terms, and so on, so forth.

The benefit of this method is that it not only considers the number of distinct terms in a document, but also how these distinct terms appeared in the document, such as how close they are.

4.4 Three-Level Scoring Method (TLS)

The **TLS** method is developed to take human expectation of search results into account and is modeled after existing manual approaches. Many existing manual methods use the following criteria to assign relevance scores [8, 9, 17]:

- Relevant links are links that are related to information needs of a query or have many links to other Web pages which may be useful to the query. They get score 2.

- Slightly relevant links are the pages that are partially related to a query. They contain too short a definition to be useful, or contain the introduction of company products that have some technical solutions to a problem relevant to the query, or contain some description to a related textbook. They are given score 1.

- Duplicate links are the pages that appear in the returned links with the same URL more than one time, and the links that differ in that one has index.htm or index.html as the suffix and the other does not. They are given score 0. Mirror sites are not counted as duplicates.

- Inactive links are the links that give error messages like file not found (404), forbidden, or server not responding (603) errors. These links are given score 0.

- Irrelevant links are the links containing irrelevant information to the query. They get score 0.

TLS method computes the relevance of a Web page to a query in the following two steps:

1. Given a query phrase q with n terms and a Web page x, a raw score is calculated as:

$$A(q,x) = \frac{t_n \cdot k^{n-1} + t_{n-1} \cdot k^{n-2} + \cdots + t_1}{k^{n-1}} \qquad (10)$$

where k is a constant, corresponding to the weight for longer sub-phrases; $t_i, 1 \leq i \leq n$, is the number of occurrence of the sub-phrases of length i, i.e., containing i terms. The order of the terms in the sub-phrases should be exactly the same as that in the original query phrase q.

2. Convert the raw score $A(q,x)$ to a three-level relevance score $sim_{tls}(q,x)$ through thresholding, with value 2 for relevant, 1 for partially relevant, and 0 for irrelevant:

$$sim_{tls}(q,x) = \begin{cases} 2 & \text{if } A(q,x) \geq \Theta \\ 1 & \text{if } \Theta > A(q,x) \geq \alpha\Theta \\ 0 & \text{if } A(q,x) < \alpha\Theta \end{cases} \qquad (11)$$

where Θ is a constant threshold, representing the requirement for being relevant; α is a value between 0 and 1, representing the requirement for being partially relevant.

The **TLS** method is modeled after the way of how manual evaluation of relevance is commonly done. Its benefit is that it not only gives higher scores to the occurrence of substrings with more distinct terms, but also considers the order of query terms because the changing of the order of query terms may change the meaning of the phrase.

Let's use a simple example to illustrate how the method works. For query "*distributed computing systems*", there are 1 phrase with 3 terms (*distributed computing systems*), 3 sub-phrases with 2 terms (*distributed computing*, *computing systems* and *distributed systems*), and 3 sub-phrases with 1 term (*distributed*, *computing* and *systems*). Assume that in a given Web page, the number of occurrence of the exact phrase is $t_3 = 2$, the total number of occurrence of the sub-phrases with two terms is $t_2 = 7$, and the total number of occurrence of the sub-phrases with one term is $t_1 = 18$. If the three parameters in the algorithm are set as $\Theta = 1$, $\alpha = 0.1$, and $k = 10$, then we get $A = (2 * 10^2 + 7 * 10 + 18)/10^2 = 2.88$, and $sim_{tls} = 2$. In the previous work, it is observed that performance evaluation results were not very sensitive to the parameter values of **TLS** and the default setting, $\Theta = 1$, $\alpha = 0.1$, and $k = 10$, generally worked well [19]. Thus, we used the default setting in the experiments.

5. EXPERIMENTS

In our experiments, 28 queries and five search engines are used. The queries are exactly the same ones as those used in [1, 2], they are *vintage car (VC), recycling cans (RE), zen buddhism (ZB), thailand tourism (TH), parallel architecture (PA), stamp collecting (SC), telecommuting (TE), sushi (SU), alcoholism (AL), classical guitar (CG), lyme disease (LD), bicycling (BI), field hockey (FH), amusement park (AP), table tennis (TT), rock climbing (RC), computer vision (CV), shakespeare (SH), cruises (CR), gulf war (GW), gardening (GA), cheese (CH), hiv (HI), affirmative action (AA), mutual funds (MF), blues (BL), graphic design (GD)*, and *architecture (AR)*. For each query, to build a base set, we started five threads simultaneously to collect top 20 hits and their neighborhood from five search engines, *AltaVista, Fast, Google, HotBot*, and *NorthernLight*, respectively. The combination of these top hits and their neighborhood forms the base set. For a document in the root set, we limit it to at most 50 in-links and collect all its out-links. The default search mode of the five search engines and lower-case queries were used. The way we construct the base set is different from the previous works [16, 1, 3], which usually build the base set from only one search engine, e.g., *AltaVista*. Combining top 20 hits and their neighborhood from five search engines gives us a more relevant base set. Running on a Sun Ultra-10 workstation with 300 MHz UltraSPARC-IIi processor connected to the Internet by the 100 Mbps fast Ethernet, our Java program took about three to five minutes to gather the base set for each query.

Duplicates or intra-domain links are then removed. For a link in either in-link set or out-link set of a root link, (1) it is considered a duplicate if it has the same name as another link in the same set, whether they are in lower case or not; (2) it is an intra-domain link if it is on the same domain as the root link. If two links only differ in that one ends with *index.html, index.htm, home.htm*, or *home.html*, while the other does not, they are considered duplicates. In addition, if two links are the same except that one begins with *www* and the other does not, they are also regarded duplicates. For example, link *http://www.zenki.com/* and link *http://zenki.com/* are duplicates.

Table 3 shows the base sets for the various queries. The number of distinct links in these sets ranges from 912 to 4037. The average in-degree and out-degree of a base-set link are 31 and 14, respectively, and about 43% of base-set links have 50 in-links. These sets of links are used for the BHITS algorithm or WBHITS algorithm. Let's call these sets of links *set 1*.

Another sets of links, called *set 2*, were generated from *set 1* by further reducing the number of links. Each link is followed to evaluate its relevance to the query. We first remove broken links. If two links have the same document length and the same relevance score using the relevance scoring methods, they are considered duplicates, and one of them is discarded if they are in the same in-link set or out-link set of a root link. Set 2 were used by the methods combining BHITS algorithm or WBHITS algorithm with one of the four scoring methods.

Table 2 lists the 10 HITS-based algorithms used in the experiments. They are five unweighted algorithms and five weighted algorithms. The five unweighted algorithms are BHITS and its combination with one of the four relevance scoring methods. The five weighted algorithms include our weighted BHITS and its combination with one of the four relevance scoring methods.

To compare the performance of the above mentioned 10 algorithms, we first use the pooling method [14] to build a query pool formed by the top 10 authority links and the top 10 hub links generated by each of the 10 algorithms, then recruit three graduate students to personally visit all documents in each query pool, and manually score them in a scale between 0 and 10, with 0 representing the most irrelevant and 10 most relevant. A Web page receives a high score if it contains both useful and comprehensive information about the query. Also, a page may be given a high score it has many links which lead to relevant message because we encouraged three evaluators to follow outgoing link and browse a page's neighborhood. We did not score the broken links and the pages that are written in language we do not understand. We did not tell the evaluators the algorithm from which a set of links are derived, and take the average of three scores as the final score for a link. the average relevance score of each query is shown in Table 3.

The average relevance scores of BHITS and WBHITS on 28 queries is plotted in Fig. 1, which clearly demonstrates that WBHITS remarkably improves BHITS. In our experiments, we notice that *small-in-large-out* links are very common in the base sets of queries, and we find *small-in-large-out* links in all the 28 queries except query 5, 15, 16, and 18. WBHITS shows good job on almost all the queries with *small-in-large-out* links except that the relevance scores are slightly worse than those of BHITS algorithm for queries 1 and 13. Especially, WBHITS gets significant improvement over BHITS on queries 2, 8, 17, 19, and 26 by preventing the results from converging to *small-in-large-out* links.

Table 4 presents the average improvement/degradation (%) of relevance scores between two algorithms. It shows that the combinations of BHITS/WBHITS algorithm with a relevance scoring method outperform BHITS algorithm, with improvement ranging from 18.3% to 22.7%. For example, *WB* improves the relevance scores of *B* by 18.3%. The improvement of *CB*, *OB*, *TB*, and *VB* over *WB* is marginal, ranging from 0.1% to 2.7%. The improvement of *WCB*, *WOB*, *WTB*, and *WVB* over *CB*, *OB*, *TB*, and *VB* is 0.9%, 0.9%, 1.0%, and 1.7% respectively, which tells us that a weighted in-link approach has little effect on the BHITS algorithm when both of them combine content analysis. Table 4 shows that the combination of BHITS algorithm with any of the four scoring methods has comparable performance, with **CDR** the best and **VSM** the worst. The best algorithm *CB* improves *OB*, *TB*, and *VB* by only 0.7%, 0.7%, and 2.9%, *WCB* improves *WOB*, *WTB*, and *WVB* by only 0.7%, 0.7%, and 2.4%, respectively.

Besides the improvement of average relevance scores between two algorithms, we compare the performance of different algorithms using two statistical metrics: *expected loss* (EL) [4] and *probably approximately correct* (PAC) [5]. We set the confident level 0.95 for PAC and the loss threshold 0.05 for EL. The results are shown in Table 5. Both metrics give consistent results: *WB* is better than *B*; *CB* is the best and *VB* is the worst among the four algorithms combined with relevance scoring methods; and *OB* better than *TB*, although two PAC values are a little below the confident level.

Table 3: The number of links in the base set for each query and the average relevance scores of 28 queries computed by the 10 HITS-based algorithms.

	Query	Links	HITS-based Algorithm									
			B	WB	CB	OB	TB	VB	WCB	WOB	WTB	WVB
1	VC	2065	6.9	5.95	7.5	7.5	7.35	7.05	7.4	7.4	7.4	6.0
2	RE	912	4.95	7.55	7.9	7.25	7.7	6.15	7.9	7.25	7.7	6.15
3	ZB	2017	8.5	8.5	8.45	8.35	8.55	8.55	8.9	8.7	8.65	8.5
4	TH	2634	6.7	6.95	7.55	7.6	7.75	6.05	7.95	7.95	7.65	8.0
5	PA	2645	8.25	8.25	8.5	8.45	8.45	8.25	8.5	8.4	8.4	8.15
6	SC	1994	8.45	8.45	8.6	8.45	8.45	8.45	8.6	8.45	8.45	8.45
7	TE	1958	8.5	8.5	8.6	8.5	8.7	8.7	8.5	8.4	8.4	8.4
8	SU	2068	3.15	7.95	8.25	8.2	8.25	8.25	8.25	8.2	8.25	8.25
9	AL	2299	7.95	8.75	8.25	8.7	8.35	8.35	8.7	8.75	8.75	8.75
10	CG	2060	9.5	9.5	9.6	9.7	9.7	9.7	9.6	9.7	9.7	9.7
11	LD	2243	8.5	8.85	8.7	8.8	8.2	8.25	8.9	8.65	8.8	8.9
12	BI	2534	7.2	7.7	8.7	8.4	8.5	8.5	8.7	8.35	8.6	8.6
13	FH	1453	7.9	7.45	7.8	7.8	7.8	7.8	7.8	7.8	7.8	7.8
14	AP	1732	8.5	8.5	8.5	8.5	8.5	8.5	8.5	8.5	8.5	8.5
15	TT	1499	8.0	8.0	8.25	7.65	7.6	7.55	8.6	8.7	8.65	8.65
16	RC	2049	7.9	7.9	7.55	8.05	8.0	7.65	7.55	8.05	8.0	7.65
17	CV	3589	6.0	8.25	8.25	8.25	8.25	8.25	8.25	8.25	8.25	8.25
18	SH	2909	8.4	8.4	8.1	8.45	8.45	8.45	8.1	8.65	8.65	8.65
19	CR	2608	3.45	7.2	7.5	7.45	7.45	7.45	7.5	7.35	7.3	7.3
20	GW	1806	7.3	7.95	8.3	8.3	8.3	8.3	8.45	8.1	8.1	8.4
21	GA	3459	7.95	8.55	8.4	8.65	8.8	8.8	9.0	8.75	8.8	8.8
22	CH	2336	8.15	8.15	8.2	8.2	8.25	8.25	8.2	8.2	8.2	8.2
23	HI	3028	7.2	8.75	8.8	8.7	8.7	8.7	8.8	8.7	8.7	8.7
24	AA	1425	7.75	8.25	8.6	8.6	8.65	8.65	8.4	8.4	8.25	8.35
25	MF	3357	7.55	8.3	8.8	8.4	7.9	8.1	8.25	8.4	8.35	8.4
26	BL	4037	4.05	7.95	8.65	7.55	8.15	8.15	8.8	8.45	8.25	8.25
27	GD	2818	8.0	8.0	8.0	8.05	8.05	8.05	8.0	8.05	8.05	8.05
28	AR	3322	8.25	8.6	8.15	8.5	8.15	8.15	8.45	8.45	8.45	8.45

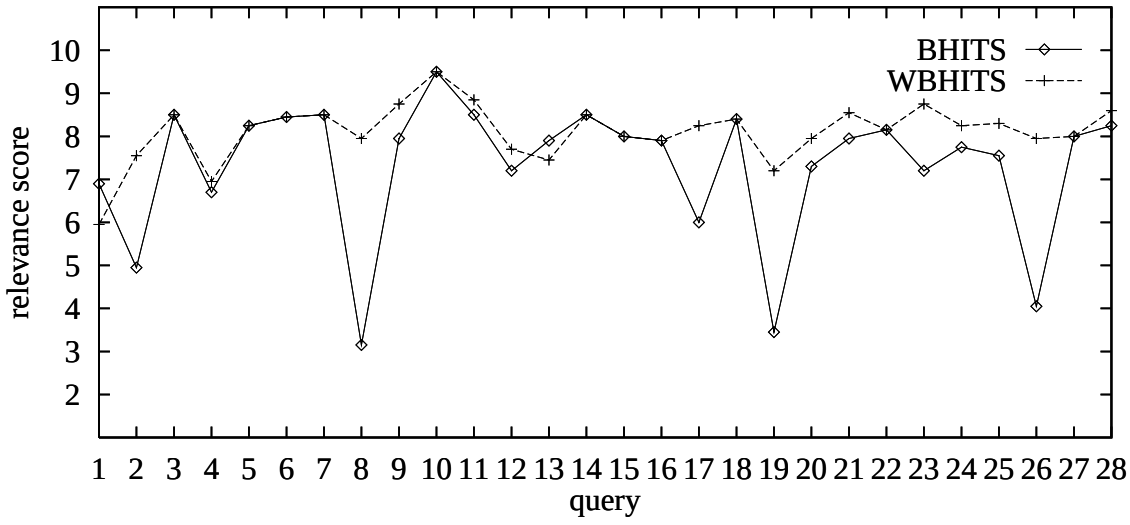

Figure 1: The average relevance scores of 28 queries computed by BHITS and WBHITS, respectively.

Table 4: Average improvement/degradation (%) of relevance scores between two algorithms. Each number in the table is the improvement/degradation of the method in the column over the method in the row. Positive numbers mean improvement, and negative numbers mean degradation.

	B	WB	CB	OB	TB	VB	WCB	WOB	WTB	WVB
B	0.0	-18.3	-21.7	-20.3	-20.7	-18.3	-22.7	-21.5	-21.6	-19.8
WB	18.3	0.0	-2.7	-2.0	-2.1	-0.1	-3.6	-2.9	-2.9	-1.5
CB	21.7	2.7	0.0	0.7	0.7	2.9	-0.9	-0.3	-0.3	1.5
OB	20.3	2.0	-0.7	0.0	0.0	2.2	-1.7	-0.9	-1.0	0.8
TB	20.7	2.1	-0.7	0.0	0.0	2.2	-1.6	-0.9	-1.0	0.8
VB	18.3	0.1	-2.9	-2.2	-2.2	0.0	-3.9	-3.1	-3.2	-1.7
WCB	22.7	3.6	0.9	1.7	1.6	3.9	0.0	0.7	0.7	2.4
WOB	21.5	2.9	0.3	0.9	0.9	3.1	-0.7	0.0	0.0	1.6
WTB	21.6	2.9	0.3	1.0	1.0	3.2	-0.7	0.0	0.0	1.7
WVB	19.8	1.5	-1.5	-0.8	-0.8	1.7	-2.4	-1.6	-1.7	0.0

Table 5: Statistical performance comparison of different algorithms. *WBvB* means the statistical comparison of *WB* over *B*, and the others are similarly defined.

Statistical Method	Statistical Comparison of a Pair of HITS-Based Algorithms						
	WBvB	CBvOB	CBvTB	CBvVB	OBvTB	OBvVB	TBvVB
EL	0.0	0.007	0.005	0.001	0.017	0.001	0.001
PAC	0.999	0.941	0.964	0.995	0.878	0.993	0.989

6. CONCLUSION

In this paper, we present a new weighted HITS-based algorithm that improves the Bharat's pure link weighting approach. Our experimental results show that there are no significant performance differences among the four relevance scoring methods when they are combined with HITS-based algorithms. Our weighted HITS-based algorithm is simple, yet effective. Without any content analysis, it achieved good results comparable to those obtained by combining connectivity and content analysis.

7. REFERENCES

[1] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 104–111, 1998.

[2] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource compilation by analyzing hyperlink structure and associated text. *Computer Network and ISDN Systems*, 30:65–74, 1998.

[3] H. Chang, D. Cohn, and A. McCallum. Creating customized authority lists. In *Proceedings of the Seventeenth International Conference on Machine Learning*, Stanford, CA, 2000.

[4] S. Chien, J. Gratch, and M. Burl. On the efficient allocation of resouces for hypothesis evaluation: A statistical approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):652–665, July 1995.

[5] S. Chien, A. Stechert, and D. Mutz. Efficient heuristic hypothesis ranking. *Journal of Artificial Intelligence Research*, pages 375–397, 10 (1999).

[6] C. L. A. Clark, G. V. Cormack, and F. J. Burkowski. Shortest substring ranking. In *Fourth Text Retrieval Conference (TREC-4)*, pages 295–304, Gaithersburg, MD, 1995.

[7] C. L. A. Clarke, G. V. Cormack, and E. A. Tudhope. Relevance ranking for one to three term queries. *Information Processing & Management*, 36:291–311, 2000.

[8] S. J. Clarke and P. Willett. Estimating the recall performance of web search engines. *Aslib Proceedings*, pages 184–189, July/August 1997.

[9] W. Ding and G. Marchionini. A comparative study of web search service performance. In *ASIS'96: Proc. 59th ASIS Annual Meeting*, pages 136–141, Medford, NJ: Information Today, Inc., 1996.

[10] A. Farahat, T. LoFaro, and J. C. Miller. Modification of kleinberg's hits algorithm using matrix exponentiation and web log records. In *Proceedings of the 24th International Conference on Research and Development in Information Retrieval (SIGIR 2001)*, New Orleans, USA, September 2001.

[11] L. Gravano, H. Garcia-Molina, and A. Tomasic. Gloss: Text-source discovery over the internet. *ACM Transactions on Database Systems*, 24(2):229–264, June 1999.

[12] D. Grossman, O. Frieder, D. Holmes, and D. Roberts. Integrating structured data and text: A relational approach. *Journal of the American Society for Information Science*, 48(2), February 1997.

[13] D. Hawking, P. Bailey, and N. Craswell. Acsys trec-8 experiments. In *Proceedings of the TREC-8*, 1999.

[14] D. Hawking, N. Craswell, and P. Thistlewaste. Overview of the trec-7 very large collection track. In *Proceedings of the TREC-7*, 1998.

[15] B. J. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: A study of user queries on the web. *SIGIR Forum*, 32(1):5–17, 1998.

[16] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. Ninth Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 668–677, ACM Press, New

York, 1998.

[17] H. V. Leighton and J. Srivastava. First 20 precision among world wide web search services (search engines). *J. of the American Society for Information Science*, 50(10):870–881, 1999.

[18] R. Lempel and S. Moran. The stochatic approach for link-structure analysis (salsa) and the the tkc effect. In *Proceedings of the 9th International World Wide Web Conference*, pages 387–401, Elsevier Science, New York, 2000.

[19] L. Li and Y. Shang. A new statistical method for evaluating search engines. In *Proc. IEEE 12th Int'l Conf. on Tools with Artificial Intelligence*, 2000.

[20] S. L. MacCall and A. D. Cleveland. A relevance-based quantitative measure for internet information retrieval evaluation. In *Proceedings of the American Society for Information Science 1999 Annual Meeting*, pages 763–768, 1999.

[21] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Series in Computer Science. Addison-Wesley Longman Publ. Co., Inc., 1989.

[22] Y. Shang and L. Li. A new method for automatic performance comparison of search engines. *World Wide Web*, 3(4), December 2000.

[23] A. Singhal, G. Salton, M. Mitra, and C. Buckley. Document length normalization. technical report tr95-1529. Technical report, Department of Computer Science, Cornell University, Ithaca NY, 1995.

[24] D. Sullivan. www.searchenginewatch.com/reports/sizes.html, Search Engine Sizes, August 15, 2001.

[25] R. Wilkinson, J. J. Zobel, and R. Sacks-Davis. Similarity measures for short queries. In *Fourth Text Retrieval Conference (TREC-4)*, pages 277–285, Gaithersburg, MD, 1995.