VIPAS: Virtual Link Powered Authority Search in the Web

Chi-Chun Lin
Electrical Engineering Department
National Taiwan University
Taipei, Taiwan, ROC
megaa@arbor.ee.ntu.edu.tw

Ming-Syan Chen
Electrical Engineering Department
National Taiwan University
Taipei, Taiwan, ROC
mschen@cc.ee.ntu.edu.tw

Abstract

With the exponential growth of the World Wide Web, looking for pages with high quality and relevance in the Web has become an important research field. There have been many keyword-based search engines built for this purpose. However, these search engines usually suffer from the problem that a relevant Web page may not contain the keyword in its page text. Algorithms exploiting the link structure of Web documents, such as HITS, have also been proposed to overcome the problems of traditional search engines. Though these algorithms perform better than keyword-based search engines, they still have some defects. Among others, one major problem is that links in Web pages are only able to reflect the view of the page authors on the topic of those pages but not that of the page readers. In this paper, we propose a new algorithm with the idea of using virtual links which are created according to what the user behaves in browsing the output list of the query result. These virtual links are then employed to identify authoritative resources in the Web. Specifically, the algorithm, referred to as algorithm VIPAS (standing for virtual link powered authority search), is divided into three phases. The first phase performs basic link analysis. The second phase collects statistics by observing the user behavior in browsing pages listed in the query result, and virtual links are then created according to what observed. In the third phase, these virtual links as well as real ones are taken together to produce an updated list of authoritative pages that will be presented to the user when the query with similar keywords is encountered next time. A Web warehouse is built and the algorithm is integrated into the system. By conducting experiments on the system, we have shown that VIPAS is not only very effective but also very adaptive in providing much more valuable information to users.

**Proceedings of the 29th VLDB Conference,
Berlin, Germany, 2003**

1 Introduction

The World Wide Web has become the most important communication channel on the Internet since its first introduction in 1989. After more than 10 years' development, the Web now consists of billions of pages which provide us with an enormous amount of information. The type, format and category of information available in the Web ranges from the entertainment to technological news, from simple texts to audio-video clips, and from casual writings to academic research reports. The Web is undoubtedly one of the greatest inventions in the last century, and is gaining increasing attention for years to come. However, the Web is loosely structured and lacks of means for efficient storage management and retrieval. While the amount of documents is growing at rapid pace, the issue of devising a method to well manage this rich data source is of increasing importance.

The purpose of data mining [8, 10] is to discover knowledge from large databases, using various mining algorithms such as association rule mining [8], classification [22], clustering, sequential pattern mining, etc. Since the Web can be viewed as a huge document database, the idea of applying data mining techniques to the Web, referred to as Web mining [15, 24], was explored by many researchers. Briefly speaking, the field of Web mining can be divided into three categories: Web content mining [18], Web structure mining [11, 20] and Web usage mining [21]. Unlike many previous studies that aimed merely at one of the three Web mining fields, the focus of this paper is a combination of Web structure and usage mining. To be more specific, we are interested in the problem of looking for Web pages that match the user's interest. It is noted that there have been many search engines [1, 9, 12, 17, 23] that can be used for this purpose. However, these search engines utilize keyword-based search method and often return a long list of search results, many of which are not necessarily what the user wants, thus likely leading to a tedious process for the users to run through all links of the list to find the truly relevant information. One way to reduce the number of search results is to provide more keywords and concept constraints to narrow down the scope of searching. The drawback of this approach is that an ordinary user is not always able to compose precise constraints for searching. Moreover, sometimes even though the keywords do effectively constrain the search space, the results may still not be satisfactory. This is because relevant contents do not necessarily contain the keywords in their page text. For an example adopted from [14], Netscape's homepage does not contain the phrase "Web browser" even though it is a software company for the famous Netscape browser.

Consequently, alternatives to keyword-based searching have arisen. Researchers have proposed algorithms such as HITS [7, 14], ARC [6], PageRank [4] and WebQuery [5] that use link analysis to determine the authority of Web pages. In the HITS analysis, a "base set" of pages, which is obtained from an ordinary search engine with a keyword-based query and then expanded according to links contained in the documents, is taken as the input to the algorithm. The output is an authority and a hub score for each page. The authority score of a page serves as the indication of quality or relevance to the keywords for itself. After the computation of scores, pages with high authority scores will be placed at the top of search results for the user to examine first. Algorithms of this sort can in many cases identify pages that are most relevant to the user's need. Though being generally better than keyword-based ones, such algorithms still have some shortcomings. Among others, one major problem is that links in Web pages are only able to reflect the view of the *page authors* on the topic of those pages but not that of the *page readers*. Moreover, these algorithms are not adaptive in that once the authority scores are computed, they are fixed unless the link structure of the Web has changed and scores are re-computed. After all, the problem of relevant information seeking in the Web is still a research field worthy of exploring.

In order to remedy the difficulties stated above, we propose a new algorithm with the idea of using virtual links which are created according to what the user behaves in browsing the output list of the query result. These virtual links are then employed to identify authoritative resources in the Web. A Web warehouse will be built and utilized in searching for relevant pages that a user is interested in. Specifically, the algorithm, referred to as algorithm VIPAS (standing for virtual link powered authority search), is divided into three phases. The first phase, called the *initialization phase*, performs basic link analysis for each keyword and stores the initial result in the warehouse. When a user comes to the system and issues a query, we record the submitted keyword and show the pre-computed authority list to the user. We then observe in the second phase (called the *virtual link collection phase*) the user behavior on how he/she clicks the listed URLs in the query result. In light of the statistics observed, we create virtual links going to pages that are deemed most relevant to users' need. A weight determined by the sequence by which the user clicked the URLs in the query result list is assigned to each virtual link. These steps concerning the creation and weight determination of virtual links are what the second phase of VIPAS includes. Finally in the third phase (called the *refinement phase*), these virtual links as well as real ones are employed together to compute updated authority scores for pages archived in the warehouse. As will be validated by our experimental results, these newly computed authority scores are able to better suit the users' information need. Consequently, when the same query is encountered next time, the list of pages ranked by the updated scores will be presented to the user.

In our work, the integration of the new algorithm with the warehouse is conducted. We have implemented a prototype of the proposed warehousing system. Technical issues in the implementation of the system are discussed. Several experiments are conducted to evaluate the performance of VIPAS and compare VIPAS with the original HITS. It is empirically shown that VIPAS is not only very effective but also very adaptive in providing much more valuable information to users. Specifically, the longer the system runs, the better it performs in retrieving the user's desirable information, showing very good adaptability of the system built.

The rest of this paper is organized as follows. Section 2 contains problem description and preliminaries. Section 3 will describe algorithms and the architecture of the proposed system. Section 4 discusses the implementation and technical issues. Experimental results and comparison with other related works can be found in the same section. This paper concludes with Section 5.

2 Problem Description

The goal of our work is to build a Web warehouse capable of performing a new information retrieval algorithm. Specifically, the problems we face can be divided into the following. First of all, we must be able to capture the users' behavior while they are browsing the Web pages. Secondly, we have to use some algorithms to determine whether a page satisfies the interest of a user, taking both the link structure of pages and the browsing behavior of users into account. Finally, a mechanism for building, managing and using the warehouse is required.

2.1 Authority Evaluation

Kleinberg's HITS (*Hyperlink Induced Topic Search*) algorithm [14] aims at finding good authority Web pages given a collection of pages on the same topic. HITS analyzes the hyperlinks contained in the pages to find the most authoritative ones among them. A page pointing to many other pages is called a "hub" page, while the one being pointed to by many pages is considered to be a good "authority." In addition, a hub page pointing to many good authority

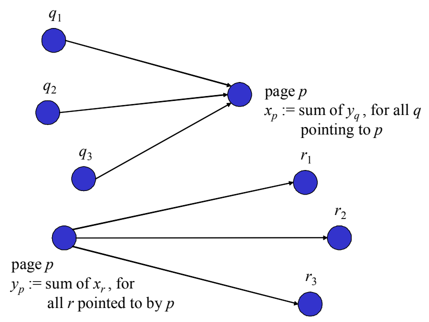

q_1
q_2
q_3
page p
x_p := sum of y_q, for all q pointing to p
r_1
r_2
r_3
page p
y_p := sum of x_r, for all r pointed to by p

Figure 1: The relationship between hubs and authorities.

pages is deemed a better hub page. Similarly, authority pages being pointed to by many good hub pages are better authorities. This forms a circular relationship called *mutual reinforcement* between authorities and hubs. The goal of HITS is to find the best authority pages. To break the circularity, the algorithm first gives each page an initial authority and hub score, and then repeatedly updates the scores in an iterative manner. Figure 1 depicts the relationship between hub and authority scores, where x_p and y_p denote the authority and hub score of page *p*, respectively.

To show the mutual reinforcement relationship numerically, consider the following inference: if *p* points to many pages with large *x*-scores, then it should receive a large *y*-score; if *p* is pointed to by many pages with large *y*-scores, then it should receive a large *x*-score. We now describe the HITS algorithm in details. The input of the algorithm is an arbitrary set of hyperlinked pages, represented as a directed graph $G = (V,E)$, where V consists of all pages in the environment, and a directed edge $(p,q) \in E$ indicates the presence of a link from p to q. We then define two operations: $\mathcal{I}$ and $\mathcal{O}$. Given scores $\{x_p\}$ and $\{y_p\}$, the $\mathcal{I}$ operation updates the *x*-score as follows:

$$x_p \longleftarrow \sum_{q:(q,\,p)\in E} y_q \ .$$

The $\mathcal{O}$ operation updates the *y*-scores as follows:

$$y_p \longleftarrow \sum_{q:(p,\,q)\in E} x_q \ .$$

The $\mathcal{I}$ and $\mathcal{O}$ operations are the basic operations by which hubs and authorities reinforce each other. To find the desired "equilibrium" values for *x* and *y*, one can apply the $\mathcal{I}$ and $\mathcal{O}$ operations in an alternating fashion, and see whether a fixed point is reached. The procedure of HITS is shown below. More details can be found in [14].

Algorithm HITS: Hyperlink Induced Topic Search

1. For a query term, obtain a set of pages using a search engine. Keep the top k pages in a "root set" R;
2. Join R with pages pointed to by R and those pointing to R to form a "base set" B;
3. Assign each page in B an authority score of 1 and a hub score of 1;
4. In each iteration, update the authority and hub scores of each page using the $\mathcal{I}$ and $\mathcal{O}$ operations;
5. Normalize the scores so that $\sum_{p\in B}(x_p)^2 = 1$ and $\sum_{p\in B}(y_p)^2 = 1$;
6. Repeat steps 4 and 5 until all x_p and y_p converge.

It is shown in [14] that after several runs of update, the hub and authority scores of each page will converge to a certain value[1], and the iteration can be terminated. The final authority score of a document can be taken as the indication of its relevance to the keyword. Therefore documents with high authority scores are expected to have relevant contents, whereas documents with high hub scores are expected to contain links to relevant contents. This implies that the URLs of pages with highest authority scores can be placed at top of the query result list to be inspected by the user first, and pages with highest hub scores will be returned to the user if this user needs a collection of links to pages with content valuable to him/her.

With the help of this algorithm, one may find good authority pages and settle the problem encountered in keyword-based indexing schemes. There is a big defect, however, that links in Web pages only reflect page creators' regard, but not page readers'. Note that in many causes the author of a page will not put a link in his/her page even though its destination is really very relevant. For example, a company is rare likely to put links to its competitors in its homepage. By analyzing merely the links put by page creators, we are unable to find all of the companies' homepages in the same competing industry. In other words, we argue that page readers' regard has to be considered as well. We will describe an improvement that takes this point into account in subsequent sections.

2.2 Client-side Data Collection

As pointed out in [21], an extensive survey on Web mining, there are three types of sources that can provide data to serve as the input to mining processes, namely (1) server logs, (2) proxy logs and (3) client-side data. Most of the existing Web mining projects use server logs as the source data. However, this type of data suffers from several problems, such as the difficulty in user session identification. These problems arise mainly from the presence of cached resources in the browser or the proxy servers. For example, when a user presses the "Back" button of the browser to return to the former page this user just went by, the cached copy will be displayed without being explicitly requested from the server. This phenomenon will cause the preprocessing step of Web mining to become a laborious task. The data from proxy logs has similar problems. In fact, there is little difference between these two types of logs. Problems we encounter from using server logs will still present when proxy logs are used instead.

To remedy this, one way is to employ a client-side data collection module in the system. There is no standard way

[1]This is under the condition that there are no negative scores.

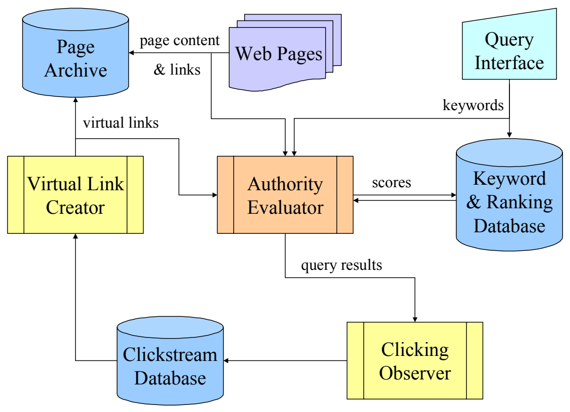


Figure 2: The framework of the warehousing system.

for performing client-side data collection. The use of Java Applets [19, 25] or Java Scripts is one solution which however suffers from limited functionality and access restriction. A more powerful way is to use a modified browser with users' consent. By employing this, many statistics of client behavior can be collected such as the time a user spends on a page before he/she jumps to another one by following a link in that page, and whether the user has saved the URL of the page in the bookmark. For an obvious reason, the use of a modified browser is not suitable for ordinary users. Without loss of generality, we choose to use a carefully designed ASP script to meet our data collection purposes. This script is meant to record whether a URL in the result list output by the system after a query execution is clicked or just skipped by the user. Details for this will be described later in Section 3.4.

3 Algorithms and System Scheme

In this section, we will describe VIPAS algorithm which is designed to further improve the search results over HITS by taking the user behavior into account. Firstly, the system framework is described in Section 3.1. The basic idea of our algorithm is presented in Section 3.2. Section 3.3 contains the detailed steps for the new method. Finally, the client-side data collection module is developed in Section 3.4.

3.1 The Whole Framework

We will build a warehouse and utilize it to satisfy users' information need. The user queries our system with keywords, and then the query result of page URLs ranked by some criteria will be presented. We next observe and record the user behavior with the query result, and then, in light of these records, devise virtual links to improve the system.

Figure 2 shows the framework of the proposed system. The user submits the query from the query interface, and the query keywords will be recorded in a database. A page archive stores all Web pages obtained from a keyword-based search engine[2] and those that are linked together via hyperlinks, as well as various page information such as title, size, date of modification, etc. Moreover, hyperlinks between pages are also stored. This archive serves as the source of input to the authority evaluator, which will be discussed in the following subsections.

There is a clicking observer module that observes whether a URL in the query result is indeed clicked by the user or not. The observed clickstreams are stored too. These will be fed to another component which adds the discovered virtual links among relevant pages. The mechanism for creating virtual links will be described later in this section.

3.2 The Notion of Virtual Links

As previously stated, the major drawback of HITS is that page readers' regard cannot be reflected in the search results. The links between Web pages are fixed after being created, and will not change until an updated copy of the document they reside in is provided by the author. Once the authority and hub scores are computed, the ranking of pages according to these scores will remain the same unless we run the algorithm periodically. And even if we do monitor the page update and continuously re-rank the pages, it is not guaranteed that the updated ranking will be more satisfactory to the user than the previous ranking.

Consequently, it is desirable to have a mechanism to adaptively adjust the authority scores according to the users' behavior. Consider the following scenario: There is a user interested in the Java programming language and who wants to find Web pages that help the learning of the language. To fulfil this information need, the keyword "Java" is used to obtain a root set from a search engine and HITS is run to produce a list of authority pages. The user finds that the top ten pages in the list except the sixth one are indeed good resources for learning the Java language, for there are lots of stuff such as the language's specification, the language's syntax and semantics, programming tutorials and examples, etc. in these Web pages. The sixth URL in the list, however, is a homepage of a book publisher that publishes many Java-related books. Though such a page is considered relevant in general cases, it is not suitable for this specific user's need, namely looking for Web pages that are *directly* helpful to his/her learning the language. In such a case, we want to lower the ranking of the sixth URL in the list. In other words, we have to reduce the authority score of the sixth URL's page or, alternatively, magnify those of other URLs.

3.2.1 Virtual Links from a Virtual Hub to Hot Pages

As stated above, we need a way to modify the scores. Since we present the list of page URLs for the user to click and browse, we may also record whether a certain URL in the

[2]This part of search engines is flexible and orthogonal to the key component of VIPAS. In fact, some popular engines such as AltaVista, Lycos or Yahoo can be used instead.

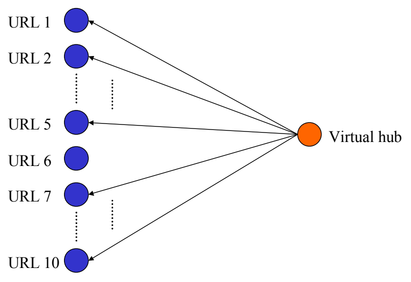


Figure 3: Virtual links from the virtual hub to hot pages.

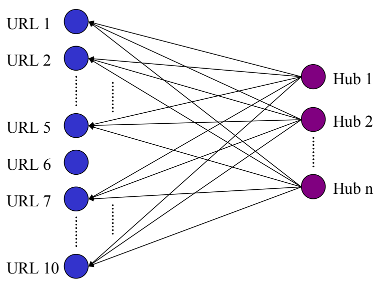


Figure 4: Virtual links from top n hubs to hot pages.

list is often clicked or is often ignored by the user. The feature of HITS algorithm is that if a page has the more pages linking to it, the higher its authority score is. Therefore we come up with the idea of putting "virtual links" going to pages that are often accessed by the user. There are two alternatives of the source of these virtual links.

Criterion 1 for adding virtual links: We create an imaginary page, called the "virtual hub", with links going to pages whose scores are what we want to modify. The set composed of those target pages are referred to as the "*hot set.*" In this specific example, the hot set consists of the top ten URLs excluding the sixth one. Figure 3 contains an illustration for this. With these virtual links incorporated into the HITS computation, pages in the hot set will be assigned with even higher authority scores. Hence, the page of the sixth URL will become relatively less important after the addition of virtual links.

The virtual hub can be viewed as an analogy to the query result page. Since the result page contains many links to (possibly) authoritative pages, it can be thought as an imaginary hub page. The concept of virtual hubs originates from this idea.

3.2.2 Virtual Links from Top Hubs to Hot Pages

On the other hand, it is observed that if a page A_1 is pointed to by a hub H_1 with a higher hub score than a hub H_2 pointing to a page A_2, A_1 will receive a higher authority score than A_2. Based on this observation, we have the other criterion of creating virtual links.

Criterion 2 for adding virtual links: We pick up the top *n* hubs and put virtual links from these hubs to pages in the hot set, where *n* is an adjustable parameter for the warehousing system. There is an exposition for this in Figure 4. Since the authority score of a page is determined by the hub scores of pages pointing to it, with the virtual links added we can significantly enlarge the authority scores of the hot set pages, thereby achieving the goal of raising their rank as well as letting the book publisher's homepage be less relevant to the user's information need.

Note that we do not try to directly reduce the score of the sixth URL's page here. The reason is that if we want to reduce a page's authority score by virtual links, these links must come from pages with negative hub scores (i.e. these links are "negative links") or be given negative weights. However, as pointed out in [14], there is a risk that the iterative update cannot terminate because the scores will not converge due to the presence of negative links. This is the very reason we avoid directly reducing the score of the sixth URL's page here.

3.3 Design of Algorithm VIPAS

We now describe the complete process of the improved method we propose. The process can be briefly outlined as follows. For a query term we first use the ordinary HITS to obtain an initial ranking of pages. Each keyword as well as the relevant pages with the computed authority and hub scores are stored in the database. Afterwards, each time when a user queries the system with a certain query term, the list of URLs ordered by each corresponding page's authority score will be presented. The user will click the listed URLs from the top of the list to the bottom one by one, or ignore some URLs at the top and jump to subsequent ones listed at the latter. We record URLs that are clicked by the user to form a hot set, discarding those that are ignored. Then, virtual links are added and incorporated into the computation of new scores. Finally the list of pages ranked by the newly computed scores is presented next time when the query with the same query term is encountered.

The algorithm devised is called VIPAS (***V**irtual L**I**nk **P**owered **A**uthority **S**earch*), whose procedure can be divided into three phases. The first phase, called the *initialization phase*, performs the regular HITS analysis. A database will be created to store the document information, link topology, and the authority and hub scores for each page in the base set. The second phase is the *virtual link collection phase*, which monitors the user behavior to observe whether a link in the list is clicked by the user or not. After a period of observation, virtual links are created according to one of the two criteria described previously. These virtual links are stored in the database as well. The final phase, called the *refinement phase*, uses the original authority and hub scores plus the ordinary and virtual links to obtain a new ranking that better suits the users' interest.

```
Algorithm VIPAS: Virtual Link Powered Authority Search
Initialization Phase:
1. For a query term, perform the regular HITS analysis;
2. Collect a base set of pages with computed authority
    and hub scores and store them in the database.
```

Virtual Link Collection Phase:

3. Monitor the user behavior to see whether a URL in the list is clicked by the user or not;
4. After a period of user behavior observation, put URLs that are often accessed into the "hot set";
   (described in Section 3.3.1)
5. Create virtual links for pages in the hot set.

Refinement Phase:

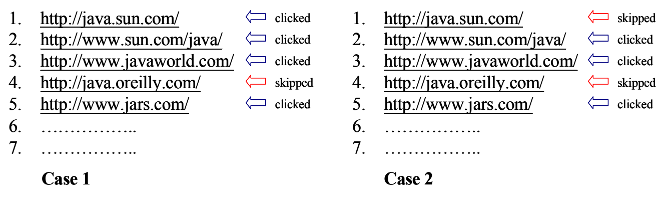
6. For each page in the hot set, compute its new authority and hub scores;
7. Run several iterations of score update for pages in the base set.
   (see Section 3.3.3 for detailed discussion on steps 6 and 7)

Note that there are two criteria for creating virtual links in step 5 of VIPAS, as described in Section 3.2.1 and Section 3.2.2. In the discussion of experiment results to be appeared in Section 4, we will refer to that using Criterion 1 as VIPAS-VH (VIPAS with virtual links from a Virtual Hub), while one using Criterion 2 is called VIPAS-TH (VIPAS with virtual links from Top Hubs). In the following subsections, we shall detail some important processes of VIPAS. Section 3.3.1 discusses issues on the creation of hot sets. Section 3.3.2 shows how to assign weights to virtual links. Finally, the computation of new scores after virtual links are added is presented in Section 3.3.3.

3.3.1 Issues on Hot Sets

The way by which the hot set is formed needs to be further explored. In the scenario in Section 3.2, the hot set consists of the top ten pages in the list except the sixth one. Now consider that another user with the same interest queries the system to find Java-related Web pages. This user has already browsed the pages of the first and second URLs in the list before coming to our system, perhaps because he/she has ever used some search engine to find out those pages and browse them. Therefore this user will skip the first two pages and begin from the third one. Under this situation, we obviously should not exclude the pages of the first and second URLs from the hot set. Hence, thinking in the reverse side, we propose the forming of the hot set as follows. We define a period of time, say two weeks, for observing the users' browsing behavior. Within this period, we pay attention to clicks of *continuous* URLs in the list. When a user continuously clicks several URLs according to the order of the list and then skips one or more URLs following and goes on with subsequent ones, we mark the URLs being skipped. For example, if there is such a clickstream from a user: 3rd, 4th, 5th, 7th and 8th. We mark the sixth URL as having ever been skipped in this period. After the period is over, we exclude all the pages of URLs that were marked with frequency greater than α from the forming of the hot set. Among the pages left, those that are accessed by at least β percent of users with the same query are put into the hot set. α and β are system-dependent parameters.

Note that the marking of being skipped must follow the requirement that the skipping by the user is occurred after some continuous clicks of URLs in the list. For example, if the user skips the first n URLs and begins with the n+1th one, these first n URLs will not be marked because they do not satisfy the condition. The purpose of setting this constraint is to avoid marking URLs that are in fact important but their pages usually have already been browsed by the users before they seek the aid of our system. Because they are often put at the top of the list, users are likely to skip them and begin with URLs listed behind them. We shall not mark these URLs under such circumstances.



1. http://java.sun.com/ clicked
2. http://www.sun.com/java/ clicked
3. http://www.javaworld.com/ clicked
4. http://java.oreilly.com/ skipped
5. http://www.jars.com/ clicked
6. ……………..
7. ……………..
Case 1
1. http://java.sun.com/ skipped
2. http://www.sun.com/java/ clicked
3. http://www.javaworld.com/ clicked
4. http://java.oreilly.com/ skipped
5. http://www.jars.com/ clicked
6. ……………..
7. ……………..
Case 2

Figure 5: An example of user behavior on clicking URLs.

Figure 5 shows an example of the observed user behavior. In case 1, URL 4 is marked as being skipped. In case 2 though both URL 1 and URL 4 are skipped, only URL 4 is marked because the former does not satisfy the condition stated above.

3.3.2 Assigning Weights to Virtual Links

After the hot set is constructed, we create virtual links according to one of the two criteria described in Section 3.2.1 and Section 3.2.2. To let the effect of virtual links reflect more precisely what the user acts, we assign each virtual link a weight which is determined by the observed user clickstream. Here we explain how to establish these weights by illustrative examples.

Example 1 : Initially after the first observation period T_1 is over and the hot set for each keyword is formed, we set weights as follows. Let the members of a hot set corresponding to a certain keyword be t_1, t_2, ... and t_n. In the observation period, if there was a session with clickstream $(t_1, t_2, t_3, t_4, x_1, x_2)$ by the user where x_1 and x_2 are pages not in the hot set, we give weights $w_{1,1}, w_{1,2}, ..., w_{1,n}$ to links going to $t_1, t_2, ..., t_n$ respectively as

$$\begin{aligned} w_{1,1} &= \tfrac{4}{6} \times \tfrac{4}{1+2+3+4} \quad (= 0.267) \\ w_{1,2} &= \tfrac{4}{6} \times \tfrac{3}{1+2+3+4} \quad (= 0.200) \\ w_{1,3} &= \tfrac{4}{6} \times \tfrac{2}{1+2+3+4} \quad (= 0.133) \\ w_{1,4} &= \tfrac{4}{6} \times \tfrac{1}{1+2+3+4} \quad (= 0.067) \\ w_{1,5} &= w_{1,6} = ... = w_{1,n} = 0 \end{aligned}$$

where $w_{c,m}$ is the weight determined by clickstream c for t_m. The $\frac{4}{6}$ comes from the fraction of the number of hot set pages among total number of URLs clicked (i.e. the user clicked 6 URLs in total, but only 4 of them are in the hot set). The $\frac{k}{1+2+3+4}$ reflects the order by which the hot set pages are clicked in the clickstream. Since t_1 is clicked first, it should receive a higher weight than t_2, t_3 and t_4. Therefore the numerator is the click order counted in the reverse direction (i.e. in the sequence t_1 t_2 t_3 t_4, t_1 should be given 4, t_2 should be given 3, and so on). The denominator is the sum of all orders. Because $w_{1,5}$ through $w_{1,n}$ are not accessed in this session, their weights are given 0.

Example 2 : If there was another clickstream (t_3, x_1, t_1), we have

$$w_{2,1} = \frac{2}{3} \times \frac{1}{1+2} \quad (= 0.222)$$
$$w_{2,3} = \frac{2}{3} \times \frac{2}{1+2} \quad (= 0.444)$$
$$w_{2,2} = w_{2,4} = w_{2,5} = ... = w_{2,n} = 0$$

because in the sequence $t_3\ t_1$, t_3's order is 2 and t_1's order is 1.

If there was a clickstream consisting of a single click (t_j), then from this session the weight of t_j will be 1 with weights of all other hot set pages being 0. For each observed clickstream in the period, we compute the weights by the way just described. Finally we take the average as the final weight value for this period. For example, if there were totally N clickstreams in the period, we compute N weights respectively for each of t_1, t_2, ... and t_n and then take the average over the N ones from clickstream 1, 2, ..., N as the final value:

$$w_h(T_1) = \frac{\sum_{k=1}^{N(T_1)} w_{k,h}}{N(T_1)}$$

for $h = 1, 2, ..., n$. Going on from the previous two examples, if there were only two clickstreams in total from this period, we have $w_1(T_1) = \frac{0.267+0.222}{2} = 0.245$, $w_2(T_1) = \frac{0.200+0}{2} = 0.100$, and so on.

After the weights for period T_1 are determined, we take these weights into the computation of new scores by the way to be described in Section 3.3.3. However, the weights for subsequent periods are produced in a slightly different manner. For a period T_i where $i \geqq 2$, the weights are computed as:

$$w'_h(T_i) = \frac{\sum_{k=1}^{N(T_i)} w_{k,h}}{N(T_i)}$$

$$w_h(T_i) = \frac{1}{3} \times w_h(T_{i-1}) + \frac{2}{3} \times w'_h(T_i).$$

The purpose of this is to let weights from more recent data represent higher significance than those from clickstreams of previous periods, with a degeneration factor of 1/3.

3.3.3 Computing the New Scores

We now describe the computation of the new scores after virtual links are created. Consider a simplified example for illustration purpose. Figure 6 depicts several pages and their link topology, where a_n and h_n denote authority and hub page, respectively. The two pages t_1 and t_2 in the figure are pages in the hot set, and the page hv is the virtual hub. The links from hv to t_1 and t_2 are shown in border lines, meaning that they are virtual links we added.

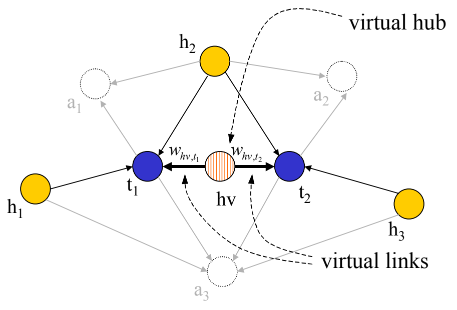


Figure 6: An example of pages and links in calculating authority scores by Equations (1) and (2).

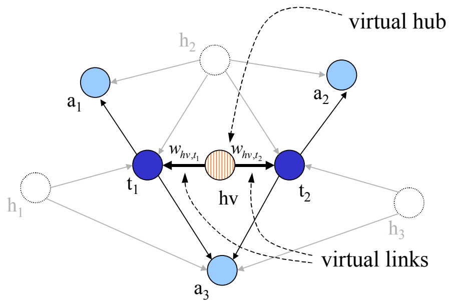


Figure 7: An example of pages and links with different parts involved in calculating hub scores by Equations (3), (4) and (5).

It can be seen that this form of virtual link creation follows Criterion 1 described in Section 3.2.1.

Following the convention from Section 2.1, let x_p denote the authority score of page p whereas y_p is used to represent page p's hub score. These scores have been pre-computed using the HITS algorithm in Section 2.1. Recall that we update the scores by the $\mathcal{I}$ and $\mathcal{O}$ operations in a regular HITS iteration. Now we show the computation of new scores after virtual links are incorporated as follows. For each page p, we update p's authority score by

$$x_p \longleftarrow \sum_{q:(q,\,p)\in E} y_q + \gamma_A \sum_{q':(q',\,p)\in E'} w_{q',\,p}\, y_{q'}$$

where *E'* is the set consisting of all virtual links created, and γ_A is a parameter used to represent the relative importance of virtual links to real links. As will be shown in Section 4, the value of γ_A is empirically determined with a typical value around 10. The larger γ_A is, the more significantly virtual links affect the authority scores. In addition, $w_{q',\,p}$ denotes the weight of the virtual link going from q' to p. In this specific example explained in Figure 6, we have

$$x_{t_1} = y_{h_1} + y_{h_2} + \gamma_A w_{hv,\,t_1} y_{hv} \quad (1)$$

$$x_{t_2} = y_{h_2} + y_{h_3} + \gamma_A w_{hv,\,t_2} y_{hv}. \quad (2)$$

In Figure 6 the pages shown in dotted lines (i.e. a_1, a_2 and a_3) are those not involved in the computation of new scores at this stage.

Now consider the update of hub scores. The procedure is similar. See Figure 7 for the same set of pages and links but with different parts highlighted. For each page p, we update p's hub score by

$$y_p \longleftarrow \sum_{q:(p,\,q)\in E} x_q + \gamma_H \sum_{q':(p,\,q')\in E'} w_{p,\,q'}\, x_{q'}$$

where γ_H is another parameter serving the analogous function to γ_A but this time for hub scores. In this example, we have

$$y_{t_1} = x_{a_1} + x_{a_3} \tag{3}$$

$$y_{t_2} = x_{a_2} + x_{a_3} \tag{4}$$

$$y_{hv} = \gamma_H w_{hv,\,t_1} x_{t_1} + \gamma_H w_{hv,\,t_2} x_{t_2}. \tag{5}$$

In each iteration we update the authority and hub scores of all pages by these formulas, and then normalize them as in the regular HITS.

A complete iteration is processed this way, but several additional iterations must be run. We note that only a few iterations are required for the whole system to enter the "steady state", i.e. the scores converge. This can be explained by the reason that we begin with an initial vector of scores that are already in the equilibrium state before virtual links are added, not those consisting of all 1's (as in step 3 of algorithm HITS). As long as the virtual links do not dramatically distort the overall link topology, we can see that the scores converge quickly. It was found in our experiments that the final sets of scores were the same as what we obtained if we had begun with all scores of 1 and run the iterative update many times until the scores converged.

The computation of new scores when virtual links are created by Criterion 2 (Section 3.2.2) is the same. In a word, the principle is that when a virtual link is encountered, we multiply the score with its weight and the importance factor so as to adjust its impact on the scores. The appropriate values for parameters γ_A and γ_H can be empirically determined.

3.3.4 Virtual v.s. Real Links

It is worth mentioning that one may encounter the situation where there is already a real link presented while a virtual link is being created. Since our goal is to magnify the importance of the hot set pages by adding virtual links, one should still create virtual links even in such cases so as to preserve every opportunity to reflect users' interest. Since we use different weights for computing scores contributed by real and virtual links, the addition of virtual links should not be viewed redundant.

3.4 User Behavior Observation Module

Figure 8 is the process we employed in observing the user behavior. We first replace each URL in the query result page with a wrapper written as an ASP script receiving the original URL as its parameter. After the user clicks some item of the list, the wrapper will be executed. The ASP script will increment the click count of that URL it received, and record the time at which the user clicked the item. The purpose of recording the click time is to determine the sequence by which the user clicked the URLs listed in the result page. Finally after these two steps are done, the user will be redirected by the wrapper to the original URL. In fact, the use of the wrapper is transparent to the user.

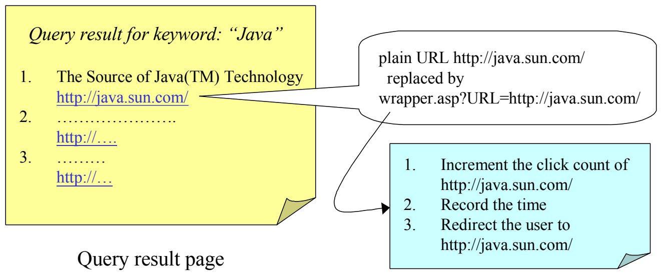

Query result for keyword: "Java"
1. The Source of Java(TM) Technology
http://java.sun.com/
2. ………………….
http://….
3. ………
http://…
plain URL http://java.sun.com/ replaced by wrapper.asp?URL=http://java.sun.com/
1. Increment the click count of http://java.sun.com/
2. Record the time
3. Redirect the user to http://java.sun.com/
Query result page

Figure 8: The client-side data collection process.

We comment that more sophisticated methods are conceivable for user behavior collection. Clearly, methods with different complexities will be able to collect the data to different levels of details. If the practical application environment permits, a more complicated client-side data collection module such as that proposed in [16] can also be used. For the demonstration of VIPAS, however, the wrapper described above suffices.

4 Experimental Results

In order to validate the proposed mechanism, we have implemented a warehousing system according to principles described in the previous sections. Here we will present the implementation details and show the experiments conducted on the system. Section 4.1 contains issues on implementation, while Section 4.2 describes system parameters employed. For the purpose of examining the effectiveness of the system, evaluation metrics are devised in Section 4.3. Finally, Section 4.4 is devoted to experimental results and comparison with the original HITS.

4.1 Implementation Details

While implementing the system, some practical considerations should be taken into account. These issues will be discussed in the following subsections.

4.1.1 The Experimental Testbed

Before going into themes of implementation, we have to first describe the environment upon which our system is built. We have chosen the website of Department of Electrical Engineering, National Taiwan University (NTUEE)[3] as the "database" where Web documents are to be searched from. Intuitively, the best environment for this kind of systems will be a traditional keyword-based search engine, into which our new ranking algorithm can be incorporated

[3]http://www.ee.ntu.edu.tw/

to provide the user with a self-learning facility according to how the user uses the engine. Since there is an in-site search function in the NTUEE website and we are free to access the site's administration permission, we are able to put the warehousing system behind that searching component. For a preliminary analysis of our work, we choose this site as our testing environment for experiments. The site's original search function exists as a CGI program in a machine-executable binary form, and therefore we use a wrapper to replace the CGI program being called after a query is submitted. What the wrapper does is summarized into the following: (1) Receive the parameters (keywords) submitted by the user, (2) Call the original searching executable to do the search and get a result list, (3) Re-arrange the ranking in the list according to our new proposed algorithm and (4) Present the result to the user, and record what the user reacts. These four steps form the main skeleton of the whole system.

4.1.2 Preparation for Document and Link Information

Because our algorithm is based on link-analysis, we have to first collect the information of documents and links in the whole website, otherwise there will be no way for the system to operate on-line. This is done in advance by a crawler. The crawler retrieves each document in the root directory of the site, parses them and extracts hyperlinks embedded in the HTML tag, and then advances to each subdirectories and repeats the same procedure all over. A database is built with Microsoft SQL Server DBMS to store these information. There are primarily two tables: Document and Link. The Document table contains each Web document's title, URL, file size, modification date, authority and hub scores[4]. Each record of the table corresponds to a specific document in the site and will be given a unique serial number (SN). The Link table is used to keep information of hyperlinks between pairs of documents. Each record of this table is composed of the following fields: SN of source document, SN of target document, anchor text[5] and link weight[6].

4.1.3 Data Collection

Recall Figure 2 in Section 3.1, where there is a clicking observer and a clickstream database in the system. The clicking observer is used to record whether a URL in the query result is indeed clicked by the user. This is done by the module described in Section 3.4. Similar to the Document and Link tables mentioned above, the recorded clicking time and count corresponding to each document is stored in a ClickTab table, which is mainly the clickstream database.

[4]Initially these scores are all set to 1.0.

[5]For further expansion purpose only, currently not used.

[6]Weights for real links are all ones, while virtual links will have weights determined by the method derived in Section 3.3.2.

SN	URL	Title
5633	H/www/faculty/rb-wu/rb-wu.htm	Homepage of professor Ruey-Beei Wu
7228	H/html_2000/WWW/faculty/english/Wu-Rei-Bei.html	[no title]
8682	H/html_2000/www/faculty/rb-wu/rb-wu.htm	Homepage of professor Ruey-Beei Wu

Figure 9: Authority list for keyword "Ruey-Beei Wu."

4.2 Parameter Sets

We next describe various parameters employed in the experiment. We do not set the upper limit for the size of root set as a regular HITS often does, while [14] used the AltaVista search engine to get the root set with a size limited to 200. This is because the number of documents in the NTUEE website is within a reasonable range. Due to the same reason, we also do not limit the number of parents brought by a page in the root set while expanding the root set to form the base set. The length of observation period is about 9 weeks[7]. The two parameters α and β in constructing the hot set are 20% and 40%, respectively. The relative importance of virtual links to real ones: γ_A and γ_H, are both 10. This means that the strongest virtual link is 10 times more important than a real link, in determining both authority and hub scores.

4.3 Evaluation Method

Here we explain how we assess the effectiveness of the system. For each keyword with a result list produced by the system, we review each page in the list and evaluate its relevance to the keyword manually with our personal point of view. A selected list of best authorities is generated from this step. Afterward we compare this authority list with that returned by the system. The more similar the two lists are, the more effective our system is.

Note that there are many evaluation methods conceivable. One such as comparing the percentage of clicks of in-the-front URLs in the list (i.e. see if the users click more on URLs placed in the front of the list and click less on those placed in the rear) before and after the new algorithm is applied can be employed if desirable. However, it is believed that these evaluation methods will lead to similar relative performance results for the algorithms evaluated.

4.3.1 Keyword for Demonstration

We have conducted experiments on several keywords. Here we use one keyword "Ruey-Beei Wu" of them to demonstrate the process of result evaluation. This keyword is the name of a professor in NTUEE. Therefore, we can easily conclude that the best authorities for this keyword should be the professor's homepages in the NTUEE website. As just described, we composed manually the authority list for this keyword, which is shown in Figure 9 where H in URL represents "http://www.ee.ntu.edu.tw." In the figure, we show each authority page's serial number in the Document table as well as its URL and title. From the URLs and titles there are apparent evidences that they are indeed the homepages of professor Wu.

[7]From March 28, 2002 to May 31, 2002.

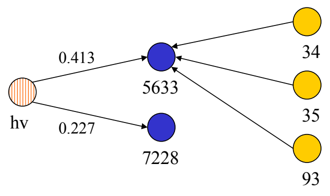


Figure 10: The link topology of pages involved in processing the example keyword “Ruey-Beei Wu.”

Rank	SN	URL	Title
1	**5633**	H/www/faculty/rb-wu/rb-wu.htm	Homepage of professor Ruey-Beei Wu
2	93	H/professor_c.html	Faculty members of NTUEE
3	34	H/prodata_c.html	Faculty members of NTUEE
4	94	H/professor_e.html	Faculty members of NTUEE
5	**8682**	H/html_2000/www/faculty/rb-wu/rb-wu.htm	Homepage of professor Ruey-Beei Wu
6	7229	H/html_2000/WWW/faculty/english/Cao-Heng-Wei.html	[no title]
7	7269	H/html_2000/WWW/faculty/english/Chen-Qiu-Lin.html	[no title]
8	5892	H/html_2000/WWW/faculty/NoSort.html	[no title]
9	4959	H/content/chinese/required/differential_equations.html	Engineering Mathematics I: Differential Equations
10	8904	H/html_2000/content/chinese/required/differential_ equations.html	Engineering Mathematics I: Differential Equations
41	**7228**	H/html_2000/WWW/faculty/english/Wu-Rei-Bei.html	[no title]

Figure 11: “Ruey-Beei Wu” - Regular HITS.

In Figure 10, we show the involved pages and virtual links created according to observed user clickstreams from the observation period. The weights assigned to virtual links are depicted as well. The number under each page is its serial number.

After the observation period was over, we performed the VIPAS analysis for this keyword. There were two alternatives of the origination of virtual links: one using a virtual hub (VIPAS-VH) and another using top *n* hubs (VIPAS-TH), corresponding to Criterion 1 and Criterion 2 stated in Section 3.2.1 and Section 3.2.2, respectively. We conducted experiments on both of the two flavors, with the latter one using *n* = 1 for comparative purposes since there is only one virtual hub in the former case. The resulted authority lists returned by the system are shown in Figures 11, 12 and 13. We only show in the figures top 10 authorities for each case for the sake of a limited space.

4.3.2 Measurements by Discrepancy and Grouping Coefficients

By comparing the authority lists with that given in Figure 9, we can see that results from VIPAS are indeed better

Rank	SN	URL	Title
1	**5633**	H/www/faculty/rb-wu/rb-wu.htm	Homepage of professor Ruey-Beei Wu
2	93	H/professor_c.html	Faculty members of NTUEE
3	34	H/prodata_c.html	Faculty members of NTUEE
4	94	H/professor_e.html	Faculty members of NTUEE
5	**8682**	H/html_2000/www/faculty/rb-wu/rb-wu.htm	Homepage of professor Ruey-Beei Wu
6	**7228**	H/html_2000/WWW/faculty/english/Wu-Rei-Bei.html	[no title]
7	7229	H/html_2000/WWW/faculty/english/Cao-Heng-Wei.html	[no title]
8	7269	H/html_2000/WWW/faculty/english/Chen-Qiu-Lin.html	[no title]
9	5892	H/html_2000/WWW/faculty/NoSort.html	[no title]
10	4959	H/content/chinese/required/differential_equations.html	Engineering Mathematics I: Differential Equations

Figure 12: “Ruey-Beei Wu” - VIPAS-VH.

Rank	SN	URL	Title
1	**5633**	H/www/faculty/rb-wu/rb-wu.htm	Homepage of professor Ruey-Beei Wu
2	93	H/professor_c.html	Faculty members of NTUEE
3	34	H/prodata_c.html	Faculty members of NTUEE
4	94	H/professor_e.html	Faculty members of NTUEE
5	**7228**	H/html_2000/WWW/faculty/english/Wu-Rei-Bei.html	[no title]
6	**8682**	H/html_2000/www/faculty/rb-wu/rb-wu.htm	Homepage of professor Ruey-Beei Wu
7	7229	H/html_2000/WWW/faculty/english/Cao-Heng-Wei.html	[no title]
8	7269	H/html_2000/WWW/faculty/english/Chen-Qiu-Lin.html	[no title]
9	5892	H/html_2000/WWW/faculty/NoSort.html	[no title]
10	4959	H/content/chinese/required/differential_equations.html	Engineering Mathematics I: Differential Equations

Figure 13: “Ruey-Beei Wu” - VIPAS-TH.

than that from regular HITS. The best authorities for keyword “Ruey-Beei Wu”: documents with SN 5633, 7228 and 8682 respectively are listed in front of the query results. In order to show the effectiveness of VIPAS in a more descriptive manner, we have to perform some formal evaluations. Traditionally, researches in information retrieval often adopt the measurement of precision and recall for the evaluation of retrieval systems [2]. Precision is defined to be the percentage of retrieved documents that are in fact relevant to the query, whereas recall is the ratio of relevant documents that are retrieved to the total number of relevant documents in the search space. However, these two measurements are not directly suitable for the evaluation of our system, because we have to focus on the comparison of two authority lists, where one is that returned by VIPAS and the other is the manually composed list. Therefore, we define two measurements: *discrepancy coefficient* and *grouping coefficient*. The discrepancy coefficient, denoted by μ, is used to measure the extent to which the authority pages are put in front of the output list. We define it as the average over the difference between each authority page's ranking in the output and its “ideal” ranking. An authority page's ideal ranking is a number among 1, 2, ... till *n*, where *n* is the number of all authority pages. For example, if there are totally three authorities, any authority page's ideal ranking should be either 1, 2 or 3. Therefore, the formula of discrepancy coefficient is

$$\mu = \frac{\sum_{k=1}^{n}(R_k - k)}{n}$$

where R_k is the ranking of the k-th encountered authority in the output list as we browse from the top of the list to the bottom. For example, in Figure 11, R_1 is 1 (authority with SN 5633), R_2 is 5 (authority with SN 8682), and R_3 is 41 (authority with SN 7228). The discrepancy coefficient is $\frac{(1-1)+(5-2)+(41-3)}{3} = 13.67$. Similarly, in Figure 12 we calculate the discrepancy coefficient as $\frac{(1-1)+(5-2)+(6-3)}{3} = 2$. The smaller the discrepancy coefficient, the more frequently authority pages are put in front of the result list.

The grouping coefficient, denoted by σ, is a measure of the locality of authorities' position in the result list. The grouping coefficient is defined as

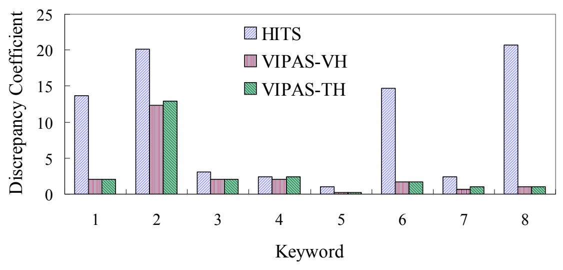


Figure 14: Comparison among HITS, VIPAS-VH and VIPAS-TH on discrepancy coefficient where the smaller the discrepancy coefficient, the more frequently authority pages are put in front of the result list.

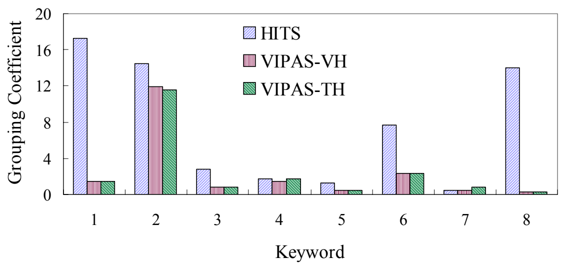


Figure 15: Comparison among HITS, VIPAS-VH and VIPAS-TH on grouping coefficient where the smaller the grouping coefficient, the more closely authority pages are put together.

$$\sigma = \sqrt{\frac{\sum_{k=1}^{n} [(R_k - k) - \mu]^2}{n}},$$

where μ is the discrepancy coefficient.

From the formula we can see that the coefficient is the standard deviation of the difference between each authority page's ranking in the output and its ideal ranking. The grouping coefficient calculated from Figure 11 is $\sqrt{\frac{[(1-1)-13.67]^2 + [(5-2)-13.67]^2 + [(41-3)-13.67]^2}{3}} = 17.25$ while that from Figure 12 is $\sqrt{\frac{[(1-1)-2]^2 + [(5-2)-2]^2 + [(6-3)-2]^2}{3}} = 1.41$. The smaller the grouping coefficient, the more closely authority pages are put together in the result list.

4.4 Overall Results and Discussion

4.4.1 Comparison on Performance and Stability

We conducted experiments on eight keywords and calculated the discrepancy and grouping coefficients for all of them. Figure 14 shows the comparison among HITS, VIPAS-VH and VIPAS-TH on discrepancy coefficient, while Figure 15 shows that on grouping coefficient. It is seen that VIPAS significantly outperforms HITS. In addition, VIPAS-VH is somewhat better than VIPAS-TH as far as discrepancy and grouping coefficients are concerned.

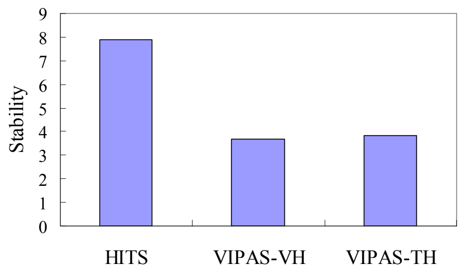


Figure 16: Comparison among HITS, VIPAS-VH and VIPAS-TH on stability.

We also calculate another measurement, which is the standard deviation of each algorithm's discrepancy coefficients for the eight keywords. We use this as the indication of the stability of an algorithm. A more stable algorithm should produce a smaller value of this measurement. The results are shown in Figure 16 where VIPAS-VH is slightly more stable than VIPAS-TH, but both of them have much better stability than HITS. The reason why VIPAS-VH performs more stably is that virtual links of VIPAS-TH originate from the top hub with a score determined by both real and virtual links. This leads to the problem of noises since real links may go to non-authoritative pages. In contrast, VIPAS-VH has virtual links going out from the virtual hub whose score is purely determined by the scores of linked authorities. Note that VIPAS-VH in essence outperforms VIPAS-TH at the expense of more storage to keep the information of one additional page, i.e. the virtual hub.

4.4.2 Remark

From the experiments, it is worth mentioning that after the practice of virtual links, only about three runs of iterative updates are needed for the scores to converge to a steady value. Note that we have pre-computed the scores before virtual links are added and archived them in the Web warehouse. Since virtual links will not significantly alter the whole link topology, the difference between two consecutive steady states will not be prominent. Therefore a few iterations, ranging from 3 to 5, are usually sufficient for the system to jump from the state corresponding to the link topology of merely real links to the one with virtual links, indicating the very advantage of Web warehousing.

Finally, we comment that some search engines have also used the idea similar to incorporating users' feedback into the computation of document ranking. Excite [9] is one example of such engines. However, the mechanism utilized by Excite was *relevance feedback* [3], which needs users' explicit reaction with the engine's interface. In [13], there is a study on the transaction log analysis of queries and user sessions for Excite. As pointed out in that literature, there was a surprisingly low percentage of sessions with relevance feedback. In contrast, our method does not need any cooperation from the user, and is able to avoid this deficiency.

5 Conclusions

In this paper, we have proposed the architecture of a Web warehousing system that is able to find authoritative Web pages for the information need of the user. By identifying hot sets consisting of pages heavily accessed by the user and creating virtual links with dynamic weights determined by the observed user behavior, VIPAS is able to provide the user with results that are progressively improving as the system is continuously being used. By conducting experiments on the system, we have shown that VIPAS is not only very effective but also very adaptive in providing much more valuable information to users.

Acknowledgement

The authors are supported in part by the National Science Council, Project No. NSC 91-2213-E-002-034 and NSC 91-2213-E-002-045, Taiwan, Republic of China.

References

[1] AltaVista. http://www.altavista.com/.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[3] N. Belkin, C. Cool, J. Koenemann, K. Ng, and S. Park. Using relevance feedback and ranking in interactive searching. *Proc. of the 4th Text Retrieval Conference*, 1996.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Proc. of the 7th International World Wide Web Conference*, 1998.

[5] J. Carriere and R. Kazman. WebQuery: Searching and visualing the Web through connectivity. *Proc. of the 6th International World Wide Web Conference*, 1997.

[6] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource compilation by analyzing hyperlink structure and associated text. *Proc. of the 7th International World Wide Web Conference*, 1998.

[7] S. Chakrabrti, B. Dom, D. Gibson, J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tompkins. Mining the link structure of the World Wide Web. *IEEE Computer*, August 1998.

[8] M.-S. Chen, J. Han, and P. S. Yu. Data mining: An overview from database perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–883, December 1996.

[9] Excite. http://www.excite.com/.

[10] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54, 1996.

[11] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring Web communities from link topology. *ACM Conference on Hypertext and Hypermedia*, 1998.

[12] Google. http://www.google.com/.

[13] B. J. Jansen, A. Spink, and T. Saracevic. The use of relevance feedback on the Web: Implications for Web IR system design. *Proc. of the 1999 World Conference on the WWW and Internet*, pages 550–555, 1999.

[14] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Proc. of ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[15] R. Kosala and H. Blockeel. Web mining research: A survey. *ACM-SIGKDD Explorations*, July 2000.

[16] I.-Y. Lin, X.-M. Huang, and M.-S. Chen. Capturing user access patterns in the Web for data mining. *IEEE 1999 International Conference on Tools with Artificial Intelligence*, pages 345–348, November 1999.

[17] Lycos. http://www.lycos.com/.

[18] M. Pazzani, L. Nguyen, and S. Mantik. Towards a WWW information filtering and seeking agent. *IEEE 1995 International Conference on Tools with Artificial Intelligence*, 1995.

[19] C. Shahabi, A. M. Zarkesh, J. Adibi, and V. Shah. Knowledge discovery from users Web-page navigation. *Workshop on Research Issues in Data Engineering*, 1997.

[20] E. Spertus. Parasite: Mining structural information on the Web. *Proc. of the 6th International World Wide Web Conference*, 1997.

[21] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: Discovery and applications of usage patterns from Web data. *ACM-SIGKDD Explorations*, January 2000.

[22] S. M. Weiss and C. A. Kulikowski. *Computer systems that learn: Classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. Morgan Kaufmann Publishers, 1991.

[23] Yahoo. http://www.yahoo.com/.

[24] O. Zaiane, M. Xin, and J. Han. Discovering Web access patterns and trands by applying OLAP and data mining technology on Web logs. *Proc. on Advances in Digital Libraries*, pages 19–29, 1998.

[25] A. Zarkesh, J. Adibi, C. Shahabi, R. Sadri, and V. Shah. Analysis and design of server informative WWW-sites. *Proc. of the 6th International Conference on Information and Knowledge Management*, 1997.