Web Intelligence and Agent Systems: An International Journal 8 (2010) 149–164
DOI 10.3233/WIA-2010-0184
IOS Press

HITS algorithm improvement using semantic text portion

Bui Quang Hung*, Masanori Otsubo, Yoshinori Hijikata and Shogo Nishida
Graduate School of Engineering Science, Osaka University, Osaka 560-6531, Japan

Abstract. *Kleinberg's Hypertext-Induced Topic Selection (HITS)* algorithm is a popular and effective algorithm to rank web pages. One of its problems is the *topic drift* problem. Previous researches have tried to solve this problem using anchor-related text. In this paper, we investigate the effectiveness of using *Semantic Text Portion* for improving the HITS algorithm. In detail, we examine the degree to which we can improve the HITS algorithm. We also compare STPs with other kinds of anchor-related text from the viewpoint of improving the HITS algorithm. The experimental results demonstrate that the use of STPs is best for improving the HITS algorithm.

Keywords: Anchor-related text, semantic text portion, hyperlink-based ranking algorithm, link structure

1. Introduction

With the growth of information on the Internet, Web search engines have become important tools for Web users. Two problems must be solved by Web search engines to satisfy Web users' information needs: the ***scarcity problem*** and the ***abundance problem*** [1]. The former is the difficulty in handling ***specific queries*** (e.g., "*pet hotel Gansu China*". Using this query, the user might seek a pet hotel in Gansu province of China). Few pages contain the required information. Consequently, it is often difficult to discover these pages. The latter is the difficulty in handling ***broad-topic queries*** (e.g., "*java beginner*". Using this query, the user might seek a page that includes basic knowledge of Java programming). The number of pages containing the given query is extremely large (e.g. hundreds of thousands of relevant pages). It presents the difficulty of determining which pages are most relevant to the given query. Our research specifically examines the abundance problem of web search engines.

1.1. Page-ranking algorithms

People use a page's popularity to solve the abundance problem. Technically web search engines use hyperlinks for estimating the page's popularity. We call its algorithm ***hyperlink-based ranking algorithm*** (Originally, Kleinberg call it ***page-ranking algorithm***). Kleinberg has shown that the main purposes of hyperlink-based ranking algorithms are: (i) measuring the popularity of relevant pages to the given query, and (ii) ordering them in decreasing order of the measured popularity [1].

Two approaches are useful for ranking web pages: ***query-independent*** and ***query-dependent*** [3]. The former is intended to measure the intrinsic popularity of a page at the time of indexing [4]. A score is assigned to every page without considering a specific user query. The most popular algorithm using this approach is the ***PageRank*** algorithm [4]. It is used in Google's Web search engine [7]. The latter conducts the ranking process at the time of searching [1,10,11,14,26–29]. For a given user query, it first collects a set of pages which is called the base set *I*. Ideally, the base set *I* has the following properties: (i) the number of pages is small (e.g. a few thousand), (ii) it contains pages which are most relevant to the given query [1]. Then the approach assigns a score to each page in the base set *I*. This score measures the quality and the relevance of the page to the given user query. The most popular algorithm using this approach is Kleinberg's ***hyperlink-induced topic selection (HITS)*** algorithm [1]. It is used by the Ask

* Corresponding author. E-mail: bqhung@nishilab.sys.es.osaka-u.ac.jp.

web search engine [8], which is one of the four most popular Web search engines (Google, Live Search [22], Ask, and Yahoo! [23]). Originally, query-dependent approach does not consider the type of queries. Recently, some query-dependent approaches judge the type of queries (for example, informational query and navigational query [30]) and change the type of ranking methods [28,29].

It is apparent that the HITS algorithm and PageRank algorithm use the opinions of people who create links on the Web [1,3,10]. When a page has a link to another page, we designate the former ***the original page*** of the link and call the latter ***the target page*** of the link. We can expect that the author of the original page thinks that the target page contains valuable information. In other words, the link is a recommendation of the target page by the author of the original page. Opinions of the author of the original page become more valuable if the original page links to many good target pages. Therefore, the fact that the original page links to the target page suggests that the target page might also be a good page.

1.2. The HITS algorithm and its problems

In this study, we specifically examine the HITS algorithm and improve it by making it work well on the current structure of the Web. The HITS algorithm presumes the existence of two types of quality pages in the base set *I*: ***hub*** and ***authority*** [1]. An authority is a page linked-to by many other good pages. It usually contains relevant contents to the given query. A hub is a page that links to many good pages. The pages linked-to by a hub usually contain good contents for the given query. Hubs and authorities exhibit a *mutually reinforcing relationship*: a good hub links to many good authorities, and a good authority is linked-to by many good hubs. The algorithm tries to determine good hubs and good authorities. It calculates a ***hub score*** and ***authority score*** iteratively for each page in the base set *I* and ranks pages by these scores.

The HITS algorithm relies on the opinions of people who create links on the Web. Therefore, the calculated scores are easily influenced by linking activities of malicious people. On the Web, people have created the two types of meaningless links that do not contain important information [10,11]. Some are links created by people who seek to manipulate authority scores and hub scores of some specific pages. They make many pages in one host link to a single page ("Page A") in another host. Because Page A obtains many incoming links, its authority score improves. Because each page in the first host have a link to a page with high authority score ("Page A"), its hub score improves. The reverse case also occurs, in which one page ("Page B") in a first host links to multiple pages in a second host. Because Page B has many links, its hub score improves. Because each page in the second host have an incoming link from a page with high hub score ("Page B"), its authority score improves. These improper operations work well because HITS algorithm conducts an iterative calculation. This type of link causes a problem known as *mutually reinforcing relationships between hosts* or *link-spamming.*[1]

The second type of link includes links that do not convey people's opinions such as automatically generated links, links for navigation, and banner ads. This type causes a problem known as *topic drift*, by which the most highly ranked authorities and hubs tend not to be about the topic of the given query but tend to be very popular pages such as top pages of Google or Yahoo! [10,11].

The link-spamming problem is solvable using the following approach. Each link (from page *i* to page *i'* in the base set *I*) is given a weight, which indicates whether or not the link is a spam link. This weight is used when the HITS algorithm iteratively calculates the hub and authority scores of the pages in the base set *I*. Spam links are given smaller weights than weights of non-spam links. Using this approach, Bharat and Henzinger proposed an improved version of the HITS algorithm which is called the BHITS method [10] (see Section 2 for the detailed explanation of the BHITS method).

Many researchers have tried to solve the topic drift problem using the following three approaches. The first tries to select pages for the iterative calculation from the base set *I* using the information of hyperlink [26]. In detail, it deletes pages which do not link to pages in the root set *R* (see Section 2.1 for the root set *R*). It also deletes pages which are linked-to by pages in the root set *R*. By selecting pages which are related to a query, it solves the topic drift problem.

The second tries to eliminate non-relevant pages from the base set *I* using the information of content [10,11]. For each page in the base set *I*, the approach calculates the relevancy between the page and the given user query. The page is eliminated from the

[1] This definition is provided in the late 1990s. Current link-spamming is much complex than this definition. This definition can be considered as the most basic technique of link spamming.

base set *I* if the relevancy is smaller than a predetermined threshold.

The third approach is an attempt to identify *important links* from the links among the pages in the base set *I*. An important link is a link that links to an authority [14]. Each link (from page *i* to page *i'* in the base set *I*) is given a weight. Important links are given bigger weights than weights of links that are not important. The weights of links are used when the HITS algorithm iteratively calculates the hub and authority scores of the pages in the base set *I*. Using this approach, Chakrabarti and Nishimura proposed methods for solving the topic drift problem [14,27]. Our research specifically examines their methods.

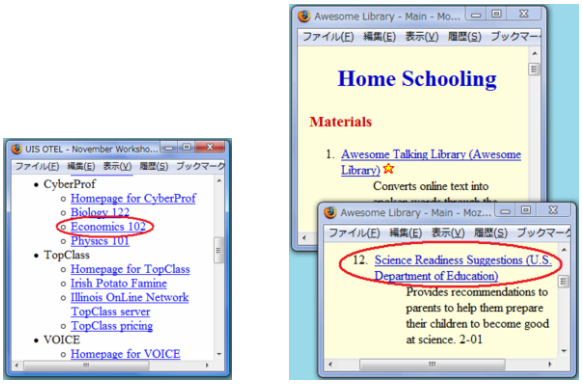
The Chakrabarti's method sees the text portion extracted using a fixed-window of 50 words around the anchor of the link in the original page to decide whether or not a link is an important link. If the given user query occurs in this text portion, it reinforces the belief that the target page of the link is an authority and the link is an important link [14].

The Nishimura's method sees the content of the target page [27]. When the query term is decorated by HTML tags, she gives a bigger weight to all the links to the target page. We think that the content information gives much impact on the importance of links. We examine the Chakrabarti's method after here.

We realize that the Chakrabarti's method presents the following disadvantage. For all links which the method tries to decide whether or not they are important, it conducts the same extraction of the text portion around the anchor of the link, even if the formats of the original pages of the links differ. This might engender misidentification of important links.

Firstly, it might extract text portions that are unrelated to the anchors of the links. In a page where one paragraph explains one target page, the whole of the surrounding 50 words are related to the target page in high possibility. However, in a page with a link collection, the surrounding part of the anchor might be filled with other links. In this case, the surrounding 50 words are not related to the target page. Figure 1(a) gives an example. The user's query is 'economics' and the anchor to the target page is an anchor with a red ellipse. However, many anchors which are not related to 'economics' surrounds this anchor.

Secondly, important explanations about the target page might exit beyond the surrounding 50 words. Especially in a link collection, explanations of the target page exist in the upper level of the itemization of the links. In this case, the Chakrabarti's method decides that the link to the target page is not important. Figure 1(b) gives an example. The user's query is 'home schooling' and the anchor to the target page is an anchor with a red ellipse. However, 'home schooling' does not exist in the explanation for this anchor and exists in the upper-level header of this page.



(a) Surrounding noise anchors

(b) important explanations beyond the surrounding 50 words

Fig. 1. Examples of Web pages in which the Chakrabarti's method does not work well.

1.3. Overview of our research

A method must be created which can extract with high accuracy those text portions which are semantically related to the anchors of links for which we want to determine importance. That method must overcome the disadvantages of the Chakrabarti's method. We already developed such a method in a previously reported study [15]. We designate a text portion that is semantically related to the anchor of a link as a *Semantic Text Portion (STP)*. We conducted a deep investigation into the locations of STPs using real pages on the Web. We then proposed a method for extracting STPs based on this result. Experimental results show that the proposed method can extract STPs with high accuracy [15].

The motivation of our current research is investigation of the effectiveness of using STPs for improving the HITS algorithm. We first study how much we can improve the HITS algorithm using STPs. Then we compare STPs with other kinds of anchor-related text from the viewpoint of improving the HITS algorithm. The experimental results demonstrate that the use of STPs is best for improving the HITS algorithm. Furthermore, we consider what aspect in the anchor-related text is important for improving the HITS algorithm.

The remainder of this paper is organized as follows. Section 2 explains the HITS algorithm and its problems. Section 3 provides an overview of the investigation of STPs and the extraction method of STPs. Section 4 explains our implemented system for conducting experiments. Section 5 presents discussion of the experimental results. Section 6 presents some concluding remarks and directions for future research.

2. The HITS algorithm and its improved versions

This section explains the HITS algorithm and its two improved versions: the BHITS method for overcoming the link-spamming problem and the Chakrabarti's method for overcoming the topic drift problem.

2.1. The HITS algorithm

The base set I is constructed as follows. The HITS algorithm first submits the given user query to a text-based search engine. It then selects the top 200 highest-ranked pages from the pages returned by that text-based search engine. These 200 pages form a set that is designated as the *root set R*. Then the HITS algorithm collects the root set R's neighborhood, which is the set of pages that either link to pages in the root set R or which are linked-to by pages in the root set R. The root set R and its neighborhood together form base set I.

For every page $i \in I$, let a_i and h_i respectively denote its authority score and hub score. The HITS algorithm calculates authority scores and hub scores of the pages in the base set I as follows.

(1) For every page $i \in I$, a_i and h_i are initialized to 1.

(2) Repeats the following calculation until a_i and h_i of every page $i \in I$ do not change further.

- For every page $i \in I$,

$$a_i = \sum_{i' \in O} h_{i'}, \quad h_i = \sum_{i' \in T} a_{i'} \tag{1}$$

where O is the set of pages that are in the base set I and link to page i and T is the set of pages which are in the base set I and linked-to by page i.

- a_i and h_i are normalized so that

$$\sum_{i \in I} a_i = \sum_{i \in I} h_i = 1 \tag{2}$$

Note that Kleinberg shows a theorem that the values to a_i and h_i converge as the number of repetition increases. He also shows that the convergence of this iterative calculation is rapid in his experiment.

2.2. The BHITS method

The BHITS method weights each link (from page i to page i' in the base set I) as follows.

(1) For each page $i \in I$, let k be the number of links to the page i from the same host. The BHITS method gives each link (from page i' to page i) an ***authority weight*** $auth\_wt(i',i)$ using the following formula.

$$auth\_wt(i',i) = 1/k \tag{3}$$

(2) For each page $i \in I$, let m be the number of links from page i to the same host; the BHITS method gives each link a ***hub weight*** $hub\_wt(i,i')$ using the following formula.

$$hub\_wt(i,i') = 1/m \tag{4}$$

(3) The BHITS method iteratively calculates the authority scores and hub scores of the pages in the base set I as same as the HITS algorithm. It uses the following formula instead of formula (1).

- For every page $i \in I$,

$$a_i = \sum_{i' \in O} h_{i'} * auth\_wt(i',i) \tag{5}$$

$$h_i = \sum_{i' \in T} a_{i'} * hub\_wt(i,i') \tag{6}$$

2.3. Chakrabarti's method

The Chakrabarti's method gives each link (from page i to page i' in base set I) a ***query weight*** $query\_wt(i,i')$ as in the following formula:

$$query\_wt(i,i') = 1 + n(query) \tag{7}$$

Therein, $n(query)$ is the number of occurrences of the query in the fixed-window of 50 words around the anchor of the link in the original page i.

After weighting all the links, the Chakrabarti's method iteratively calculates authority scores and hub scores of the pages in the base set I as same as

the HITS algorithm. It uses the following formula instead of formula (1).

$$a_i = \sum_{i' \in O} h_{i'} * query\_wt(i', i) \quad (8)$$

$$h_i = \sum_{i' \in T} a_{i'} * query\_wt(i, i') \quad (9)$$

3. Semantic text portion

In this section, we briefly explain our investigation of STP. Then we provide a short explanation of our proposed method for extracting STPs based on the investigation results. (For detailed information, please see the full version of our previous paper [15].)

3.1. Investigation of STPs

We realized that there are two types of STP on the Web: *Local Semantic Portion (LSP)* and *Upper-level Semantic Portion (USP)*. The LSP is a STP that exists around the anchor and the USP is a STP that exists in the upper-level structure of the original page. We conducted an investigation for each type of STP using real original pages. The objectives of the investigation are (i) to see what parts in original pages contain LSPs and USPs and (ii) to find the HTML tags which can semantically divide LSPs and USPs from other text portions in original pages.

We prepared a dataset of 1108 real original pages (752 original pages of 50 official target pages such as a government's web page and a company's web page and 356 original pages of 50 personal target pages such as an individual's web page about his hobby) in our investigation. In detail, we randomly selected 50 official target pages and 50 personal target pages from Open Directory.[2] For each target page, we found its original pages by using Google's API. We used 20 original pages (actually Google's cache) at most for each target page. Because we want to investigate many kinds of web pages, we excluded ODP copies from the collected pages.

We invited three evaluators to judge real STPs. Although they are not native English speakers, they are well trained for using English in business. For each original page in the dataset, we show the evaluators its content and the anchor pointing to the target page. The evaluators also see the content of the target page. The evaluators judge which text portions are semantically related to the anchor. We consider the text portion which at least two evaluators judge as "semantically related" as STP. We manually examine where STPs exist and what can be a separator for STP from other text portions.

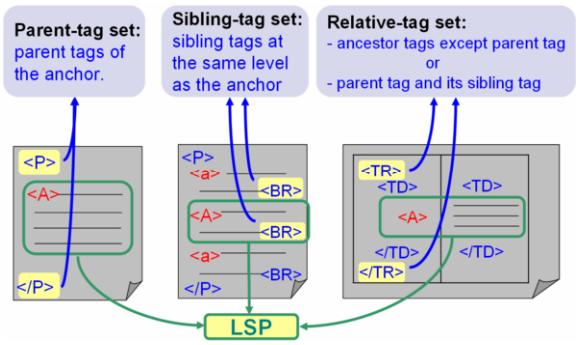


Fig. 2. Examples of HTML-tag sets for separating LSPs from other text portions.

Our investigation of LSPs revealed that LSPs are located in the following four places: table (339 pages), list (410 pages), paragraph (320 pages), and DIV object (39 pages). An object that is one of these four types and which includes the anchor of the link often includes LSP. This finding is useful for narrowing down the part which might include LSPs.

Based on the above finding, we continued to see which kind of HTML tag can semantically divide LSPs from other text segments in original pages. An HTML tag is either a start tag or an end tag of an HTML object. Results showed that an LSP is divided using a set of two tags (hereinafter, an "HTML-tag set"). We found three types of HTML-tag set to divide LSPs from other text segments in original pages semantically: a parent-tag set, a sibling-tag set, and a relative-tag set (see examples of these HTML-tag sets in Fig. 2). The parent-tag set consists of parent tags which directly include the anchor. In Fig. 2, the LSP is the whole text in a paragraph. It is divided by the parent-tag set, which is a <P> tag and </P> tag of the paragraph. The sibling-tag set consists of sibling tags which are at the same level as the <A> tag of the anchor. The sibling-tag set can divide an LSP that is in an HTML object including several anchors. In Fig. 2, the LSP is in a paragraph. Several anchors and several line breaks exist in this paragraph. The LSP is divided using a sibling-tag set, which comprises two line break tags. The relative-tag set consists of either the ancestor tags, except the parent tag, or both the parent tag and its sibling tag. The relative-tag set can divide an LSP that is in a table. In Fig. 2, the LSP is a row of a table. The LSP is divided using a relative-tag set, which is a <TR> tag and a </TR> tag of the

[2] Open Directory, http://dmoz.org/.

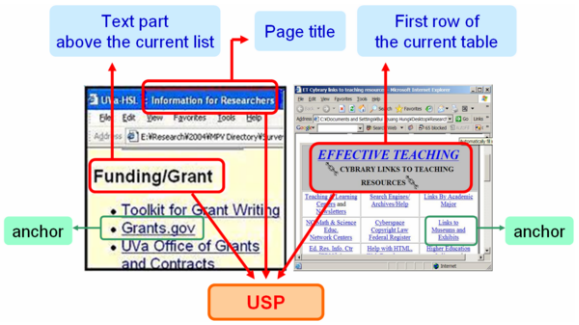


Fig. 3. Examples of USP.

row. If the computer checks the location type, the number of sibling tags, the existence of other anchors, and so on, it can find the HTML-tag set which can separate the LSP from other text portions.

Through investigation of USP, we found that a USP always exists in a specific location in an original page: page title (1097 pages), headers above the anchor (739 pages), table header (6 page), the first row of the current table (48 pages), the first row of an upper-level table (82 pages), another row (not the first row) of the current table (46 pages), another row (not the first row) of an upper-level table (167 pages), the text portion above the list (64 pages), another table (278 pages), another list (36 pages), or another paragraph (372 pages). We also give three examples of the USP in Fig. 3. The USP is in the text portion above the list which directly includes the anchor (the fist example), in the page title (the second example), in the first row of the current table, which directly includes the anchor (the third example).

3.2. Method for extracting STPs

The following is our LSP extraction method. The method first identifies what kind of object (paragraph, list, table, or <DIV> object) includes the anchor.

(i) If the anchor is in a paragraph, in a list item, or in a <DIV> object, the method counts the number of line feeder tags in the object. The method uses parent-tag set to extract LSPs if there is no line feeder tag. It uses a sibling-tag set if there are one or more line feeder tags.

(ii) If the anchor is in a table, the method identifies the current cell including the anchor. Then it expands the extraction area (the area to be extracted) from the current cell to nearby cells lengthwise and crosswise until it meets a cell that includes a different anchor. If the extraction area consists of only the current cell, it extracts LSPs from the current cell, as in (i). If the extraction area consists of more than one cell, the method uses the relative-tag set to extract LSPs.

The following is our USP extraction method.

- The method extract the title of the page and all header(s) above the anchor (extracts the nearest header to the anchor if there are several headers at the same level).
- It checks whether or not the table header exists if the anchor is in a table. If it exists, the method extracts the table header. If it does not exist, the method extracts the first row of the current table and the first row of an upper-level table.
- If the anchor is in a list, it extracts the text part above the current list.

3.3. Evaluation of the method for extracting STPs

We conducted experiments to evaluate our STP extraction method. We created a dataset comprising 200 real original pages. These pages were obtained by randomly selecting 10 official target pages and 10 personal target pages from Open Directory and collecting 20 original pages (Actually Google's cache) in each target page at most by Google's API. Furthermore we randomly selected 200 pages from the obtained original pages. We invited three evaluators to judge real STPs as in Section 3.1. The method to acquire the correct STPs is as same as in Section 3.1. We compared the text segments which were extracted by our method and the STPs. Experimental results are presented in Table 1. From this table, it is apparent that our method extracts STPs with high accuracy. Actually, we compared our method with major extraction methods of anchor-related text and found that our method achieves a good balance among precision and recall.

Table 1
Evaluation of our method for extracting STPs

	Precision	Recall
Extract LSPs	97.01%	93.94%
Extract USPs	89.43%	74.35%
Extract both LSPs and USPs	94.08%	85.03%

3.4. Using STPs for improving the HITS algorithm

We use STPs for weighting each link (from page *i* to page *i'* in the base set *I*) as in the following formula:

$$query\_wt(i,i') = 1 + n(query) \quad (11)$$

where $query\_wt(i,i')$ is the query weight of the link from original page *i* to the target page *i'*, and $n(query)$ is the number of occurrences of the query in the STP of the link in the original page *i*.

After weighting all the links, we iteratively calculate authority scores and hub scores of the pages in base set *I*, as in the Chakrabarti's method (also see Section 2.3).

4. System design

We developed a system for conducting experiments to investigate the effectiveness of using STPs for improving the HITS algorithm. We want to compare STPs with other kinds of anchor-related text that are used for weighting the links among the pages in the base set *I* in the experiments. Therefore, our system must allow the experimenter to change different kinds of anchor-related text easily.

Although we examine the effectiveness of different anchor-related text on the topic drift problem, the link-spamming problem will occur in our experiments. This might influence the results of the experiments and render the evaluation difficult because of the spam links. Therefore, we implemented BHITS method in our system.

Figure 4 portrays the process flow of our system. The system collects the base set *I* that is specific to the given user query when a user inputs a query to the system. Then it calculates query weights for the links among the pages in the base set *I*, as described in Section 3.4. The result of the calculation of the query weight differs according to which kind of text part the system uses as the anchor-related text. The system also calculates the authority weight (as in formula (4)) and the hub weight (as in formula (5)) for each link (from page *i* to page *i'* in the base set *I*). These weights are used for the BHITS method. Next, the system iteratively calculates the authority score and hub score for every page in base set *I*. Finally, it shows the user the top 10 authorities and top 10 hubs. In the remainder of this section, we explain the manner of collecting base set *I* and the calculation of authority scores and hub scores of pages in the base set *I*.

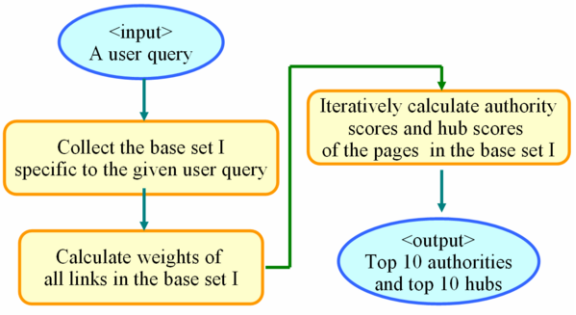


Fig. 4. Flow of the system.

4.1. Collecting base set I

Base set *I* consists of a root set *R* and its neighborhood. In previous studies [1,9–11,14], base set *I* is collected as follows. To collect root set *R*, the given user query is submitted to a text-based search engine. From pages returned by the text-based search engine, the top 200 highest-ranked pages are picked up as the root set *R*. Then the neighborhood of root set R is collected. The neighborhood is a set of pages that either links to pages in the root set *R* or are linked to by pages in the root set *R*. The root set R and its neighborhood together form base set *I*. Finally, all the links that exist between the pages in base set *I* are discovered.

Some researchers have used commercial search engines as text-based search engines such as AltaVista [24] and Hotbot [25]. AltaVista and Hotbot were text-based search engines when these researchers conducted their experiments. Currently, AltaVista and Hotbot have introduced a ranking algorithm with link analysis. To our knowledge, no open text-based search engine exists on the Web today. Current Web search engines order the matched pages from their respective popularities. We cannot compare the ranking from our improved HITS algorithm and the pure ranking from a text-based algorithm when we use search results with ranking from popularity. Therefore, the lack of open text-based search engines poses a difficulty for us in developing our system further.

To solve this difficulty, we used WebBase [12] and Lucene [13]. WebBase is an open repository of web pages; its data size is greater than 100 TB (uncompressed size as of August 2007). Lucene is an information retrieval library. It provides an indexing function and a searching function for full-text documents. It does not introduce a ranking method with link analysis. We built a text-based search engine using WebBase and Lucene for collecting the root set *R*.

WebBase outputs the URLs to a specific query. We built a repository of web page by sending queries to WebBase (actually, we sent 10 queries in our experiment in Section 5). We used Lucene for indexing the above URLs. When we input a query to Lucene, it outputs the above URLS with ranks produced by a content-based ranking like old commercial text-based search engines.

After collecting the root set *R*, our system collects the neighborhood of the root set *R*. We use Google API [5][3] for finding pages which link to a page in the root set *R*. Similarly to some previous studies [1,9–11,14], for each page in the root set *R*, we must acquire pages linking to that page and pages linked to by that page. Our system collects 50 original pages of that page at most. It also collects all pages that are linked to by that page. The system does not collect links among pages on the same host.

4.2. Calculating authority and hub

The system iteratively calculates authority scores and hub scores of the pages in the base set *I* as follows:

(1) For every page $i \in I$, let a_i and h_i respectively represent the authority score and hub score of page *i*.

(2) For every page $i \in I$, a_i and h_i are initialized to 1.

(3) Repeat the following three steps until a_i and h_i of every page $i \in I$ do not change further.

- For every page $i \in I$:

$$a_i = \sum_{i' \in O} h_{i'} * auth\_wt(i', i) * query\_wt(i', i) \quad (12)$$

where O is the set of pages in the base set I and link to page i; $auth\_wt(i, i')$ *and* $query\_wt(i', i)$ *respectively denote the authority weight and the query weight of the link from page i' to page i.*

- For every page $i \in I$:

$$h_i = \sum_{i' \in T} a_{i'} * hub\_wt(i, i') * query\_wt(i, i') \quad (13)$$

where T is the set of pages which are in the base set I and are linked to by page i; $hub\_wt(i, i')$ *and* $query\_wt(i, i')$ *respectively signify the hub weight and the query weight of the link from the page i to page i'.*

- For all pages $i \in I$, a_i and h_i are normalized as in formula (3).

After calculating authority scores and hub scores of the pages in base set *I*, the system shows URLs of top 10 authorities and top 10 hubs to the user. The user can follow the URLs to visit the top 10 authorities and the top 10 hubs.

5. Experiment

The main purpose of our experiments is to investigate the effectiveness of using STPs for improving the HITS algorithm. Through the experiments, we examine the degree to which we can improve the HITS algorithm using STPs and we compare STPs with anchor-related texts of other kinds. In detail, we compare STPs with methods using the following: (i) anchor text [16–18], (ii) text in the paragraph which directly includes the anchor [19,20], (iii) text in the fixed-window of 50 words around the anchor [14,21], and (iv) text in all upper-level headers of the anchor [20].

We used our implemented system described in Section 4 for conducting the experiments. Table 2 presents 13 methods used for the comparison in our experiments. The Random method selects 20 pages randomly from the base set *I,* and considers them as the top 10 authorities and the top 10 hubs. The Link-Frequency method considers the top 10 pages in the base set *I* that have the highest number of incoming links as the top 10 authorities, and considers the top 10 pages in the base set *I* that have the highest number of outgoing links as the top 10 hubs. The remaining nine methods (except HITS and BHITS) are named according to the kind of anchor-related that text they use for weighting the links among the pages in the base set *I.* To compare the performances of the 13 methods, we use the ***pooling method for raking results*** (after here, ***pooling method***), which was also used in previous studies [6,10,11]. In the remainder of this section, we first explain the pooling method and discuss the experimental results.

[3] Google API does not provide all the original pages to a target page. We think that the relative number of original pages among target pages can be kept in Google.

Table 2

List of 13 methods compared in our experiments

Method	Text for weighting links
Random	–
LinkFrequency	–
HITS	–
BHITS	–
AnchorHITS	Anchor text
ParaHITS	Text in the paragraph which directly includes the anchor
LspHITS	Local Semantic Portion
FixHITS	Text in the fixed windows of 50 words around the anchor
HeadersHITS	Text in all upper-level headers of the anchor
UspHITS	Upper-level Semantic Portion
ParaHeadersHITS	Text in the paragraph which directly includes the anchor and text in all upper-level headers of the anchor
FixUspHITS	Text in the fixed-windows of 50 words around the anchor and Upper-level Semantic portion
StpHITS	Local Semantic Portion and Upper-level Semantic Portion

5.1. Pooling method

The pooling method is used for comparing performances of several ranking methods [6,10,11]. In our study, we use it for comparing the above 13 methods. This method requires a set of test queries and human evaluators. For each query, the method builds a *query pool* formed by the top 10 authorities and top 10 hubs ranked by each of the above 13 methods. The evaluators are asked to visit all pages in the query pool. The information related to the method that is used for ranking the pages in the query pool is hidden from the evaluators. The evaluators are then asked to rate the pages on a scale manually between 0 and 10. Rating of a page is influenced by the following four factors: relevancy, usefulness, ease of comprehension, and number of links to good pages. A page receives a high rating if it is related to the query. That page receives a higher rating if it contains useful or comprehensive information about the query. A page is also given a high rating if it contains many links to good pages for the given query. The ***final rating*** of a page is the average of the ratings given by all the evaluators. HITS algorithm (actually 13 methods in Table 2) calculates authority score and hub score for each page and outputs top 10 authorities and top 10 hubs. We calculated the average of the final ratings for these 20 pages. We designate this average as the ***performance score***.

Table 3

The data size and the number of pages obtained from WebBase for each query

	Query	Data size	Number of pages
1	Bicycling	129MB	4819
2	Shakespeare	1.33GB	54,337
3	Cruises	664MB	40,643
4	Affirmative Action	950MB	67,120
5	Alcoholism	578MB	17,088
6	Architecture	2.58GB	126,711
7	Cheese	1.93GB	78,883
8	Gardening	606MB	20,464
9	HIV	569MB	26,634
10	Telecommuting	512MB	17,733

For our experiment, we used 10 queries: *telecommuting (TE), alcoholism (AL), bicycling (BI), Shakespeare (SH), cruises (CR), gardening (GA), cheese (CH), HIV (HI), affirmative action (AA), and architecture (AR)*. These 10 queries were used in previous studies [10,11,14]. Table 3 shows the data size and the number of pages obtained from WebBase for each query. We invited three graduate students to participate as human evaluators in our experiments. Although they are not native English speakers, they are well trained for using English in business.

5.2. Experiment results

Experimental results are presented in Fig. 5. The *y*-axis presents the performance scores of the 13 methods. From this figure, it is apparent that the StpHITS method achieves the best performance among all the methods. It is surprising that the HITS algorithm is the worst method; the BHITS method also yields a very bad result. The HITS algorithm and the BHITS method are even worse than the Random method and the LinkFrequency method. The remaining nine methods, which use anchor-related text for identifying important links, obtain better results than the four methods (HITS, BHITS, Random, and LinkFrequency) which do not identify important links.

From this result, we examine the following issues: (i) Why do the HITS algorithm and BHITS method achieve very bad results? (ii) Why does the StpHITS method achieve the best result?

Table 4
Performance scores of three methods for each query

	AA	AL	***AR***	***BI***	CH	CR	GA	HI	SH	TE	Average
LinkFrequency	0.57	0.63	***2.45***	***4.53***	1.72	1.10	2.02	5.07	4.27	0.60	2.30
HITS	0.48	0.47	***0.13***	***5.17***	0.82	0.10	1.67	3.33	0.37	0.35	1.29
BHITS	0.50	0.55	***0.15***	***7.18***	1.27	0.12	1.75	7.47	0.35	0.38	1.97

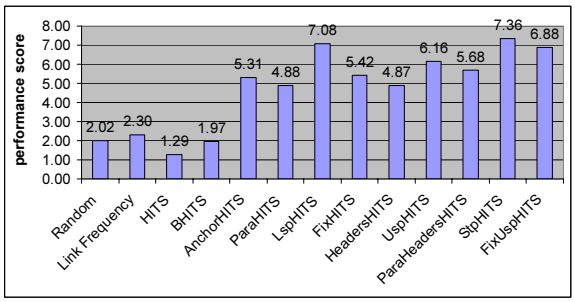
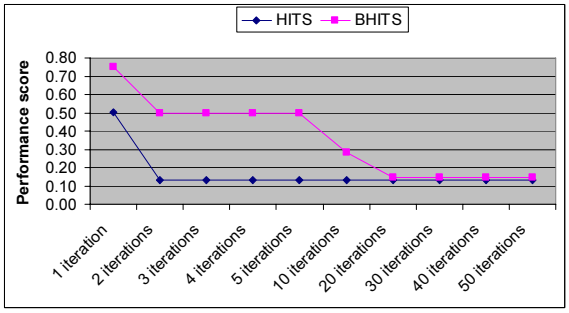
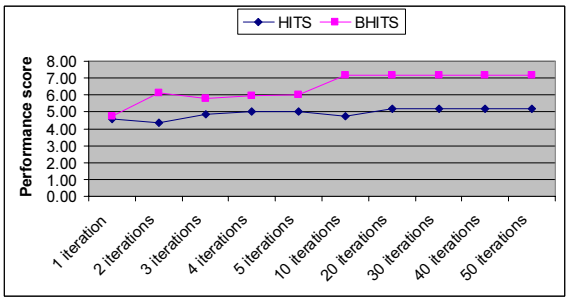


Fig. 5. Performance scores of the 13 methods.



a) Results for the *architecture* query



b) Results for the *bicycling* query

Fig. 6. Performance scores of the HITS algorithm and the BHITS method when we change the number of iterations

5.3. Investigation of the weakness of the HITS algorithm and the BHITS method

In this section, we investigate the weakness of the HITS algorithm and the BHITS method. Surprisingly, the HITS algorithm and the BHITS method present worse results than the LinkFrequency method, which is the most basic method for exploiting the link structure of the Web for ranking web pages. The LinkFrequency method considers a page as an authority if the page has numerous incoming links, and considers a page as a hub if the page has many outgoing links. Table 4 presents the performance scores of the three methods: LinkFrequency, HITS, and BHITS.

From the data presented in Table 4, we realize that the performance of the LinkFrequency method is higher than the performance of the HITS algorithm and the BHITS method for most of the 10 queries. Two special queries are *architecture* and *bicycling*. For the *architecture* query, the LinkFrequency method performance is much higher than the performance of either the HITS algorithm or the BHITS method. For the *bicycling* query, the LinkFrequency method performance is worse than the performance of the HITS algorithm or the BHITS method. We expect that by examining these two queries we can understand why the LinkFrequency method is better than the HITS algorithm and the BHITS method.

Tables 5 and 6 show the top 10 authorities and the top 10 hubs ranked using the LinkFrequency method, the HITS algorithm, and the BHITS method, respectively, for the two queries of *architecture* and *bicycling*. The results in Table 5 show that the HITS algorithm is influenced by the link-spamming problem. The BHITS method is influenced by the topic drift problem. For the result ranked by the HITS algorithm, almost all of the top 10 authorities are pages on the same host *howstuffworks.com*. All the top 10 hubs are pages on the same host *howstuffworks.com.* Because the pages in the host *howstuffworks.com* are well connected to receive high rankings by Web search engines, these pages mutually receive high authority scores and hub scores and they become the top authorities and the top hubs. From the evaluators' judgments, these pages are not related to the *architecture* query. Therefore, the performance score of the HITS algorithm for the *architecture* query is very low (0.13). For the result ranked using the BHITS method, almost all the top 10 authorities and the top 10 hubs pages are popular pages but they are not re-

Table 5
Ranking results for the *architecture* query

a) Top 10 authorities

	Link frequency	HITS algorithm	BHITS method
1	www.usa.gov	products.howstuffworks.com	www.usa.gov
2	www.whitehouse.gov	www.hsw.com.br	www.whitehouse.gov
3	www.buffalo.edu	mobiltravelguide.howstuffworks.com	www.buffalo.edu
4	www.google.com	Consumerguideauto.howstuffworks.com+B5	www.google.com
5	mobiltravelguide.howstuffworks.com	videos.howstuffworks.com	www.hsw.com.br
6	www.hsw.com.br	Communication.howstuffworks.com	academic.oreilly.com
7	Products.howstuffworks.com	auto.howstuffworks.com	Consumerguideauto.howstuffworks.com
8	consumerguideauto.howstuffworks.com	home.howstuffworks.com	videos.howstuffworks.com
9	Videos.howstuffworks.com	people.howstuffworks.com	reminders.barnesandnoble.com/?z=y
10	Auto.howstuffworks.com	health.howstuffworks.com	www.sun.com

b) Top 10 hubs

	Link frequency	HITS algorithm	BHITS method
1	www.lib.utexas.edu/apl/internet_ resources.html	www.howstuffworks.com/file-sharing.htm	www.educationworld.com/contact/
2	lanic.utexas.edu/la/region/architecture	computer.howstuffworks.com/ myspace5.htm	www.howstuffworks.com/ file-sharing.htm
3	www.library.yale.edu/art/ subjectguides/architecture.html	computer.howstuffworks.com/ hardware-channel.htm	www.research.ibm.com/cell/ cell_compilation.html
4	wings.buffalo.edu/ap/	computer.howstuffworks.com/ peripherals-channel.htm	www.worldbank.org/ifa/
5	www.education-world.com/awards/ past/r1297-02.shtml	computer.howstuffworks.com/ software-channel.htm	www.os.dhhs.gov/fedhealtharch/ index.html
6	directory.google.com/Top/Reference/ Museums/Arts_and_Entertainment/ Architecture/	computer.howstuffworks.com/ internet-channel.htm	www.firstgov.gov
7	www.academicinfo.net/archorg.html	computer.howstuffworks.com/ security-channel.htm	www.arquiperu.com
8	www.sc.edu/beaufort/library/pages/ links/fineart.shtml	Media.howstuffworks.com	www.sas.upenn.edu/ealc/faculty/ steinhardt.htm
9	dmoz.org/Arts/Crafts/Origami/ Origamic_Architecture/	electronics.howstuffworks.com/ question313.htm	Fisher.lib.virginia.edu/collections/ cities_main.html
10	vos.ucsb.edu/browse.asp?id=2705	money.howstuffworks.com/ cutting-your-own-cd4.htm	www.oreillylearning.com

lated to the *architecture* query. For example www.google.com offers search services and www.sun.com offers information about workstation and Java language. Therefore the performance score of the BHITS method for the *architecture* query is also very low (0.15). The HITS algorithm and the BHITS method performance scores are lower than the LinkFrequency method performance score (2.37), and much lower than the StpHITS method performance score (7.02), which is the best method for the *architecture* query.

Conversely, Table 6 shows that the HITS algorithm and the BHITS method achieve quite good results for the *bicycling* query (5.17 and 7.18). The HITS algorithm and the BHITS method are not influenced by the link-spamming problem and the topic drift problem. Most of the top 10 authorities and the top 10 hubs ranked using the HITS algorithm or the BHITS method are related to the *bicycling* query (in Table 6, the pages in gray cells receive low ratings from evaluators; other pages receive a high rating from the evaluators). Especially, the BHITS method presents a good performance score (7.18). This score is not so much lower than the performance score of the FixUspHITS method (7.68), which is the best method for the *bicycling* query.

Table 6
Ranking results for the *bicycling* query

a) Top 10 authorities

	Link frequency	HITS algorithm	BHITS method
1	www.nps.gov/sagu/	www.bicyclinglife.com	www.bicyclinglife.com
2	www.nps.gov/sagu/index.htm	www.bicycling.com	www.cyclery.com
3	www.wsdot.wa.gov/Bike/	www.adobe.com/prodindex/acrobat/readstep.html	www.bikexchange.com
4	www.adobe.com/products/acrobat/readstep2.html	www.co.oconto.wi.us	www.bicyclecolo.org
5	wsdot.wa.gov/traffic	www.exploratorium.edu/cycling/index.html	www.shimano.com
6	www.wsdot.wa.gov/bike/default.htm	Bicycling.about.com/mbody.htm	www.adv-cycling.org
7	www.wsdot.wa.gov/bike	www.sheldonbrown.com/tooltips/index.html	www.kenkifer.com/bikepages
8	www.usa.gov	Infosource.uwex.edu/index.cfm?countyid=72	www.bikecolorado.com
9	www.bicyclinginfo.org	cecommerce.uwex.edu	www.adventuresports.com
10	Find.metrokc.gov	www.flyfisherman.com	www.specialized.com

b) Top 10 hubs

	Link frequency	HITS algorithm	BHITS method
1	www.uwex.edu/ces/cty/oconto/4h/NaturalResources.html	www.uwex.edu/ces/cty/oconto/4h/NaturalResources.html	bcn.boulder.co.us/transportation/bike.page.html
2	bcn.boulder.co.us/transportation/bike.page.html	www.uwex.edu/ces/cty/oconto/4h/MechanicalScience.html	www.uwex.edu/ces/cty/oconto/4h/NaturalResources.html
3	www.uwex.edu/ces/cty/oconto/4h/MechanicalScience.html	bcn.boulder.co.us/transportation/bike.page.html	www.sneakeasysjoint.com/thecyclingdude/creative_writing/index.html
4	www.cs.indiana.edu/~robh/	www.sneakeasysjoint.com/thecyclingdude/creative_writing/index.html	www.genesbmx.com/BMXLINKS1.html
5	www.noah-health.org/en/healthy/exercise/specific/bicycling.html	probicycle.com/mainnet.html	www.cs.indiana.edu/~robh/
6	www.fhwa.dot.gov/environment/bikeped/publications.htm	www.wisconsinsportsmanmag.com	probicycle.com/mainnet.html
7	www.dot.ca.gov/hq/tpp/offices/bike/bikesites.htm	www1.umn.edu/pts/links.htm	www.heartcycle.org/Pages/hclinks.htm
8	www.december.com/places/msp/sports.html	www.friendsofsaguaro.org/links.html	rex.skyline.net/html/Bicycling.html
9	www.sneakeasysjoint.com/thecyclingdude/creative_writing/index.html	www.succulent-plant.com/botanic.html	www.uwex.edu/ces/cty/oconto/4h/MechanicalScience.html
10	www.cs.wisc.edu/~wenger/personal_links.html	www.dot.state.mn.us/library/bike_peds.html	www.bikingbis.com/blog/_archives/2007/4/15/2882355.html

Tables 5 and 6 presented above show the two results for the two queries related to *architecture* and *bicycling*. Recall that the HITS algorithm and the BHITS method iteratively calculate authority scores and hub scores of pages in the base set *I* until they do not change anymore. We conducted the following experiment for the *architecture* and *bicycling* queries. We gradually increased the number of iterations of the HITS algorithm and the BHITS method. We calculated the performance scores of the two methods after each iteration.

Figure 6 shows the experimental results. The *y*-axes in the two charts respectively present the performance scores of the methods. For the *architecture* query, the results worsen when the number of iterations is increased gradually. After the first iteration, the HITS algorithm performance score (0.50) and the BHITS method performance score (0.75) are less than the LinkFrequence method performance score (2.45). The performance scores of the HITS algorithm and the BHITS method decrease after many iterations. These performance scores are also lower than the performance scores of the LinkFrequency method.

Conversely, for the *bicycling* query, the results improve when the number of iterations is gradually increased. The performance score of the HITS algorithm (4.58) and the performance score of the BHITS

Table 7
Percentage of meaningless links in the base set

Query	*architecture*	*bicycling*
Percentage of meaningless links	62%	30%

method (4.73) are not much better than the performance score of the LinkFrequency method (4.53) after the first iteration. The performance scores of the HITS algorithm (5.17) and the BHITS method (7.18) improve considerably compared to the performance score of the LinkFrequency method after many iterations.

Based on the results described above, we realize that the HITS algorithm and the BHITS method present two opposite results for the two queries. Recall that the HITS algorithm and the BHITS method are link-based ranking algorithms. Their performances are easily influenced by meaningless links among the pages in the base set *I* (see Section 1.2 for a detailed explanation about meaningless links). We expect that the percentages of meaningless links among all the links between the pages in the base set *I* for each query were able to help us understand the two opposite results for the two queries described above. The calculation of the percentage of meaningless links is a time-consuming task because the number of links among the pages in the base set *I* is quite large for each query (1639 links for the bicycling query, and 3005 links for the architecture query). Therefore, we decided to calculate the percentage of meaningless links by sampling. We randomly selected 100 links from all the links among the pages in the base set *I*. We asked three evaluators to judge meaningless links. Table 7 shows the experimental result.

Table 7 shows that the percentage of meaningless links for the *architecture* query (62%) is larger than that for the *bicycling* query (30%). We think that when the percentage of meaningless links among all the links between the pages in the base set *I* becomes greater than 50%, it strengthen the influences of the meaningless links by repeating the score calculation. When it becomes less than 50%, the mutual influences of good pages in the base set *I* become large.

5.4. Investigation of why the StpHITS method achieves the best result

The experimental results described in the previous sections demonstrate that the methods using anchor-related text for identifying important links yield better results than the methods using only the link structure of the Web. Among the methods using anchor-related text, the StpHITS method achieves the best result. Apparently, the StpHITS method is the best for identifying important links. We presume for this study that when people consider whether or not a link to a target page is important for finding good information in their searches, they verify whether or not the link is related to their queries. Based on that supposition, it is highly likely that its anchor-related text includes the queries. Consequently, we propose the hypothesis that the text portions in which the queries occurred the most frequently are STPs. We expect that the average number of occurrences of the queries in each kind of anchor-related text helps us understand the differences in performances of the methods using anchor-related text. We clarify our hypothesis by conducting the following experiment. We calculate the average number of occurrences of the 10 queries in each kind of anchor-related text. Figure 7 shows the average number of occurrences of the queries in each kind of anchor-related text. We also show performance scores of nine methods that use anchor-related text in Fig. 8.

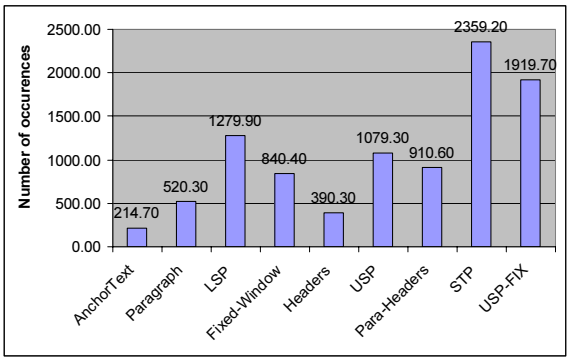


Fig.7. Average quantities of occurrences of queries in each kind of anchor-related text.

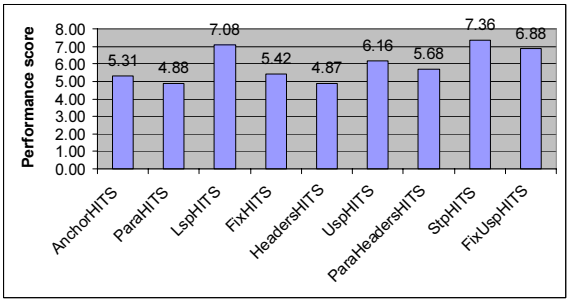


Fig. 8. Performances of the nine methods which use anchor-related text for identifying important links.

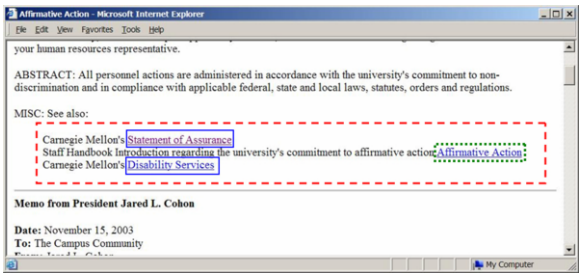


Fig. 9. An example of misidentification of important links.

Among anchor-related texts of four kinds that exist directly around the anchor (anchor-text, text in the paragraph which directly includes the anchor, LSPs, and text in the fixed-window of 50 words around the anchor), the queries occur most frequently in LSPs. From this result, we can understand why the LspHITS method achieves the best result among the four methods: AnchorHITS method, ParaHITS method, LspHITS, and FixHITS method.

The queries occur more frequently in USPs of the two kinds of anchor-related text that exist in the upper-level structure of the original page (text in all upper-level headers of the anchor and USPs). Consequently, the UspHITS method achieves a better result than the HeadersHITS method.

We also realize that the queries occur more frequently in LSPs than in USPs. Therefore, the LspHITS method outperforms the UspHITS method.

It is interesting that the average number of occurrences of the queries in the anchor text is the smallest among anchor-related text of all kinds, but the AnchorHITS method achieves better results than either the ParaHITS method or the HeadersHITS method. Similarly, the queries occur more frequently in the combination of the text in the fixed-window and the USPs than in the LSPs, but the LspHITS method achieves a better result than the FixUspHITS method. The reason for this interesting phenomenon is that the extraction method's precision[4] of anchor-related text influences the ranking method's performance. If the extraction method extracts text portions that are not semantically related to the anchors of the links, but the given query occurs in these text portions, the ranking method determines that these links are important. This misidentification of important links reduces the ranking method's performance. An example of this situation is presented in Fig. 9. The given query in this example is *affirmative action*. The page in this figure is a page in the root set R of the given query. We specifically examine the paragraph that is marked in the dashed red rectangle. Three anchors (one in the dotted green rectangle and two in the blue rectangles) exist in this paragraph. They link to three target pages in the base set I of the given query. Among these three target pages, only the second target page, which has the anchor in the dotted green rectangle, has contents related to the given query *affirmative action*. Three links from the page in the figure to these three target pages are also in the base set I. Ideally, only the link from the page to the second target page is an important link. If the extraction method extracts the whole text in the paragraph, which is represented in the dashed red rectangle, as anchor-related text, the ranking method considers all three links as important links because this text portion includes the given query *affirmative action*.

In our previously reported study [15], the precision of the Anchor-text method, which extracts the anchor-text of the anchor, is 100%. The precision of the Paragraph-based method, which extracts the text in the paragraph which directly includes the anchor, is 71.23%. We did not examine the Headers-based method, which extracts text in all upper-level headers of the anchor, but it is apparent that the precision of this method is smaller than 100%. Consequently, the AnchorHITS method outperforms the ParaHITS method and the HeadersHITS method.

In our previously reported study [15], the precision of our LSP extraction method, which extracts LSPs, was 97.01%. The precision of our USP extraction method, which extracts USP, was 89.43%. The precision of the Fixed-window method, which extracts text in the fixed-window of 50 words around the anchor, is 29.52%. We did not examine the precision of a method that extracts USPs and text in fixed-window, but it is apparent that it is lower than the precision of our LSP extraction method. Therefore, the LspHITS method outperforms the FixUspHITS method, even when queries occur more frequently in the combination of the text in fixed-window and the USPs than in the LSPs.

In Fig. 8, we compare the method using paragraph and the method using anchor text (or the method using LSP). Here we concern the performance when we use the full text of the original page instead of paragraph, anchor text, or LSP. However using the full text also causes the same problem as using the paragraph. When we consider a case in which the target page is related to the query and valuable and the original page also includes the query, using the full

[4] The precision of an extraction method of anchor-related text is the percentage of extracted text portions that are related semantically to the anchor evaluated by human evaluators [15].

text of the original page is not a problem because we just check the text includes the query. However when we consider a case in which the target page is not related to the query and the original page also includes the query, using the full text of the original page causes a problem. The full text of the original page of the unrelated target page also includes the query. Therefore HITS algorithm gives a high authority score for the target page. In this case, we want to lower the authority score of the target page.

The analyses described above prove that the precision of the extraction method of anchor-related text and the number of occurrences of the queries in the anchor-related text affect the performance of the ranking method. Fig. 6 shows that the queries occur most frequently in STPs. Results of our earlier study [15] demonstrated that our STP extraction method, which extracts STPs, achieves high precision (94.08%). The StpHITS method achieves the best result among all the methods that use anchor-related text for identifying important links.

6. Summary and future work

This paper presented an investigation of the effectiveness of using *Semantic Text Portions* (STPs) for improving the HITS algorithm. We compared STPs with anchor-related texts of other kinds from the viewpoint of improving the HITS algorithm. In detail, we compared STPs with: (i) anchor text, (ii) text in the paragraph which directly includes the anchor, (iii) text in a fixed window of 50 words around the anchor, and (iv) text in all upper-level headers of the anchor. We developed a system for conducting experiments to compare STPs with such anchor-related texts.

We used our experimental system and 10 queries for the evaluation. Three human evaluators were invited to participate in our experiments to judge the ranking result yielded by the experimental system. Experimental results demonstrate that the use of STPs is best for improving the HITS algorithm. The STPs are best because (i) the queries occur most frequently in STPs. Furthermore, (ii) STPs are more related semantically to the anchor of the link than anchor-related texts of other kinds.

In the experiment, we found that methods using the text portion which seems to be relevant to the target page (LspHITS, UspHITS, STP and Fix-UspHITS) are better than other methods. However, we do not find the statistical significance. We have to increase the number of queries for achieving the statistical significance. We want to try more queries in the future.

Through the experiments we realized that two types of semantic exist in STPs. One is facts about the target page. Another is people's opinions about the target page. A fact is information about what content is written in the target page or what service is offered in the target page. An opinion is information about how people think the target page's content or service. In this study, we have not distinguished the above two types and not exploited the difference. Especially there are two types in people's opinions: (i) positive ones and (ii) negative ones. Links with positive opinions improve the value of the target page, but links with negative opinions decrease it. We surmise that a link becomes important if its STPs include a positive opinion and the given query. We will study this idea in future work.

References

[1] J. Kleinberg, Authoritative Sources in a Hyperlinked Environment, Journal of ACM, Vol. 46, No. 5, pp. 604–632, 1999.
[2] G. Salton and M.J. McGill, Introduction to Modern Information Retrieval, 1983.
[3] M.R. Henzinger, Hyperlink Analysis for the Web, IEEE Internet Computing, Vol. 5, No. 1, pp. 45–50, 2001.
[4] S. Brin and L. Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine, in: Proc. Seventh WWW Conf., pp. 107–117, 1998.
[5] http://code.google.com/apis/.
[6] D. Hawking, N. Craswell, and P. Thistlewaste, Overview of the TREC-7 very large collection track, in: Proc. TREC-7, pp. 1–24, 1998.
[7] http://www.google.com.
[8] http://www.ask.com.
[9] J. Carriere and R. Kazman, Webquery: Searching and Visualizing the Web through Connectivity, in: Proc. Sixth WWW Conf., pp. 1257–1267, 1997.
[10] K. Bharat and M.R. Henzinger, Improved Algorithms for Topic Distillation in a Hyperlinked Environment, in: Proc. 21st ACM SIGIR, pp. 104–111, 1998.
[11] L. Li, Y. Shang and W. Zhang, Improvement of HITS-based Algorithms on Web Documents, in: Proc. WWW 2002, pp. 527–535, 2002.
[12] J. Hirai, S. Raghavan, A. Paepcke, and H. Garcia-Molina, WebBase: A repository of Web pages, in: Proc. 9th International World Wide Web Conference (WWW9), Amsterdam, May 2000.
[13] E. Hatcher and O. Gospodnetic, "Lucene in Action", Manning, ISBN 1932394281
[14] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson and J. Kleinberg, Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text, Proc. Seventh WWW Conf., pp. 65–74, 1998.
[15] B.Q. Hung, M. Otsubo, Y. Hijikata and S. Nishida, Extraction of Semantic Text Portion Related to Anchor Link, IEICE Trans. on Information and System, 2006.

[16] B.D. Davison, Topic locality in the Web, in: Proc. 23rd ACM SIGIR Conf., pp. 272–279, 2000.
[17] E. Amitay, Using common hypertext links to identify the best phrasal description of target web document, in: Proc. SIGIR'98 Post-Conference Workshop on Hypertext Information Retrieval for the Web, pp. 271–276, 1998.
[18] A. Blum and T. Mitchell, Combining labeled and unlabeled data with co-training, in: Proc. COLT 1998.
[19] E. Amitay and C. Paris, Automatically summarizing Web sites: Is there a way around it?, in: Proc. CIKM 2000, pp. 173–179, 2000.
[20] J. Furnkranz, Exploiting structural information for text classification on the WWW, in: Proc. IDA'99, pp. 487–498, 1999.
[21] E.J. Glover, K. Tsioutsiouliklis, S. Lawrence, D.M. Pennock, G.W. Flake, Using Web structure for classifying and describing web pages, in: Proc. WWW 2002, pp. 562–569, 2002.
[22] http://www.live.com.
[23] http://www.yahoo.com.
[24] http://www.altavista.com.
[25] http://hotbot.com.
[26] S. Nomura, S. Oyama, H. Tetsuo, T. Ishida, Analysis and Improvement of HITS Algorithm for Detecting Web Communities, IEICE Transactions on Information and Systems, Vol. J85-D-I, No. 8, pp. 741–750, 2002.
[27] Y. Nishimura, A New Algorithm for Analyzing the Link Structure of WWW Using Semi-structured Data, Bachelor Thesis for Nihon University, 2005.
[28] I. Kang and G. Kim, Query type classification for web document retrieval, in: Proc. 27th ACM SIGIR Conf., pp. 64–71, 2003.
[29] X. Geng, T. Liu, T. Qin, A. Arnold, H. Li, H. Shum, Query dependent ranking using K-nearest neighbor, in: Proc. 31st ACM SIGIR Conf., pp. 115–122, 2008.
[30] A. Broder, A taxonomy of web search, SIGIR Forum, 36(2), pp. 3–10, 2002.