

Discovery and Retrieval of Logical Information Units in Web

Keishi Tajima Kenji Hatano Takeshi Matsukura Ryouichi Sano
Katsumi Tanaka

Department of Computer and Systems Engineering
Kobe University

Nada, Kobe 657-8501, Japan

Tel/Fax: +81 78 803-6234

{tajima|hatano|matukura|sano|tanaka}@db.cs.kobe-u.ac.jp

ABSTRACT

In ordinary search engines for Web pages, the data unit for query processing is individual pages. Indexes are produced for each page in accordance with the words appearing in it. In actual Web data, however, a logical document discussing one topic is often organized into a set of pages connected via links provided by the page author as “standard navigation routes.” In such a situation, conjunctive queries with multiple keywords may fail to retrieve an appropriate document if those keywords appear in different pages within that document. Therefore, a data unit for Web data retrieval should not be a page but should be a connected subgraph corresponding to one logical document. In this paper, we develop new techniques for discovering and retrieving the logical information units in Web data. As in some previous researches, we adopt minimal subgraph semantics for conjunctive queries. In our approach, when given a conjunctive query, we try to approximate information units including all the given keywords in the following three steps: (1) we distinguish standard route links from the others, (2) we find minimal subgraphs connected via those links and including all the keywords, and (3) we compute the score of each subgraph based on the locality of the keywords within it in order to examine whether it is really a logical information unit relevant to the query.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Proc. of Workshop on Organizaing Wep Space (WOWS'99) in conjunction with ACM DL'99 Berkeley CA USA, Aug. 1999

KEYWORDS: Web, hypertext, query, data unit, structure discovery, information discovery

1 INTRODUCTION

Web data is a huge hypertext data consisting of a huge number of Web pages and links connecting them. Links are used in various ways. Some links are used to provide a way to jump to Web pages at other (or same) Web sites discussing a related topic. On the other hand, some links are provided by the page author as the suggested navigation routes going through a set of pages which as a whole compose one complete document. For example, a document is sometimes organized into a sequence of pages corresponding to its sections. Links with anchors saying “next” composing such a sequence are typical examples of the latter kind of links. Similarly, pages describing various information on one topic are sometimes organized into a hierarchy. Links going-down within such a hierarchy are also examples of the latter kind of links. In pages organized in those ways, links going back to some prior pages along the routes, e.g. links to the previous page with anchors “back” or links to the top page with anchors “top,” are also often found.

The users can retrieve Web data either by directly accessing Web pages at known URLs, or by navigating from pages to pages via those links. Currently, however, another important way to access Web pages is the content-based access via search engines. Search engines provide facilities to list up URLs of pages including given keywords. When a user want to newly find unknown pages discussing a topic of one's interest, one can submit a query to search engines by specifying keywords that are likely to appear in pages on that topic.

In ordinary search engines for Web pages, the data units for retrieval is individual pages. As mentioned above, however, a complete document discussing one topic is often organized into a sequence or a hierarchy of pages connected via “standard route” links. In this way, a logical information unit in Web data is not a page but is a connected subgraph corresponding to one logical document, and therefore, so should the data unit in Web query be. If we use individual pages as the data units in query processing, conjunctive queries with multiple keywords would fail to retrieve some appropriate document when those keywords happen to appear in different pages of that document.

Recently a couple of researches including ours have proposed frameworks for querying those logical information units in Web [22, 23, 26, 16, 13, 9, 8]. The approaches to discover and retrieve such logical information units are classified in several ways as follows.

static approach v.s. dynamic approach

- Static approach [22, 23, 26, 13, 8] — this approach statically divides the Web graph into fixed subgraphs corresponding to logical documents. When a query is issued, logical information units including all the given keywords are returned. This approach is a natural consequence of our original observation that there exist in Web data logical information units intended by page authors. The advantage of this approach is that we can employ complex analysis for the information unit detection without causing the response latency at query execution because the analysis can be done in advance in the phase of index creation.
- Dynamic approach [16, 9] — this approach dynamically finds a information unit including all the query keywords each time a query is issued. In this approach, partition of the Web graph into units is variable depending on the given query. In the existing researches, [16] and [9] use minimal subgraphs including all the query keywords as approximations of information units matching to the query. Some of such minimal subgraphs, however, may not actually be part of one information unit but may be spanning to multiple documents. To give priority to subgraphs that are more likely to be part of a single document, [16] uses a query relaxation schemes based on the size of the graph and [9] uses a ranking algorithms. The advantage of dynamic approach is that it can always return “best N results” as mentioned in [16].

classification based on the information used for unit discovery

- Page contents — this approach analyzes the contents of pages in order to detect logical information units. For example, in many researches, the term frequencies of neighboring pages (or subgraphs) are compared, and similar neighboring pages (or subgraphs) are merged into one logical information unit [22, 23, 26]. On the other hand, [13] uses the similarity of tagging patterns in neighboring pages. Other techniques to capture the semantics of the documents, such as natural language processing techniques, may also be effective.
- Link structure — another approach is to use information on hypertext structure. For example, graph theoretic properties, such as various kind of connectivity [2, 23, 21, 8] or fun-in/fun-out [2, 23, 8] can be clue to detecting strongly related pages. In dynamic approaches [16, 9], the size of subgraphs are used in query relaxation schemes or in answer ranking.
- Directory structure — another useful information that we can find in Web data is directory structure embedded in URL. It usually reflects the intention of page authors, and is very useful to guess which links are intended to be the standard routes [21], or which set of pages are intended to be composing a single document [23, 8].

In this paper, we develop new techniques for discovering and retrieving logical information units in Web data. In this paper, we adopt the dynamic approach, particularly the minimal subgraph approach. The reason of the choice is, although the static approach is more straightforward solution to the problem “retrieval of logical information units,” it is quite difficult to detect such units perfectly. In some cases, even the author oneself cannot tell which pages are composing a single independent document. Therefore, minimal subgraph approach is more practical.

One problem in dynamic approach is the cost to dynamically traverse Web graphs to find minimal subgraphs. If we examine arbitrary subgraphs, they usually include many subgraphs that are clearly not part of a single logical document. Searching those subgraphs is wasting time. We need some method to eliminate those “clearly wrong” cases in some early phase. The cost grows especially when the average of fun-out, the number of out-going links from a page, is large. Therefore, reducing the number of links to traverse is very effective to reduce the cost. In this research, we eliminate links that are not intended by page authors to be the standard routes, i.e. not part of a single document, by looking at

directory structure in URLs.

In dynamic approach, some query relaxation scheme or some ranking algorithm is also essential as explained above. In this research, we use a ranking method based on the “locality” of query keywords in subgraphs. A locality of a word is the area where the word has its influence. If localities of two words greatly overlap, those two words must be strongly related in that subgraph, and if they does not overlap at all, those words may be unrelated. Therefore, if the locality of query keywords in a subgraph greatly overlaps with each other, we rank that subgraph higher. We examine the locality of words both across pages and within each page.

In summary, when given a conjunctive query, we find (approximations of) information units including all the given keywords in the following three steps: (1) we distinguish route links from the other kind of links, (2) we find minimal subgraphs connected via those links and including all the keywords, and (3) we compute the score of each subgraph based on the locality of the keywords in order to examine whether it is really an information unit relevant to the query. We develop techniques for each of these steps, discuss the rationality of the semantics of conjunctive queries for Web, and show our preliminary experimental results.

The reminder of this paper is organized as follows. Next section discuss more about related work. Section 3 describes the three steps of the retrieval of logical information units. Then Section 4 discusses various semantics of conjunctive queries on Web data, and the relation between our method and them. Finally Section 5 is conclusion.

2 RELATER WORK

An early research on the detection of meaningful connected subgraphs in hypertext data is [2, 1]. They identify connected subgraphs consisting of strongly related nodes based on the graph structure in order to produce a summary or an overview map of the whole structure of huge hypertext data. On the other hand, this paper and [22, 23, 26, 16, 9] have proposed to use such meaningful subgraphs as the logical data unit for Web data retrieval.

[14, 6] also developed a method to extract “communities” of pages, the group of pages that are written by authors having the same interest and are referring to each other, by examining the link structure. On the other

hand, our purpose is to detect a single document written by one author.

There are also many researches on utilizing both content information and link information for retrieval or clustering of hypertext data [5, 27], for ranking Web pages [12, 19, 4, 3], or for classifying Web pages into several page roles [24]. Although those approaches incorporate link information in querying or clustering hypertext data, they still use individual nodes as data units for retrieval, ranking, or classification.

There is a research proposing the detection and the utilization of subtopic structure in large documents [10]. They divide a long text into smaller fragments each of which corresponds to each subtopic, and use both those fragments and the whole document as the data units in comparison with given queries. The concept of subtopic structures and the concept of the logical information units in Web data is very similar. [10], however, deals not with hypertext but only with sequential texts.

Our first research on the detection of meaningful subgraphs in hypertext data is [11]. In that research, we have proposed a method for identifying connected subgraphs discussing one topic in newsgroup articles or mailing list archives, which have hypertext structure induced by “Reference:” or “Reply-to:” header information. We used similarity between neighboring nodes based on the term frequency. In the workshop where we presented that paper, Hiroyuki Kawano of Kyoto University suggested that the same idea can also be applied to Web search. Following that suggestion, we have developed a framework to detect meaningful subgraphs in Web data and to use those subgraphs as the logical data unit in the retrieval [22, 26].

The idea of using minimal subgraphs including all keywords is, as far as we know, first suggested by Yoshinori Hara of NEC USA. In an informal meeting with us, he suggested it as another solution to the problem of conjunctive queries in Web retrieval. Later, that idea was presented in [16] by members of NEC USA, and in [9] by some of us. In [16], query results are progressively produced in the order of the size of the subgraph, which must also be the order of the relevance of the subgraphs to the query. [9] proposed a method of ranking subgraphs based on the locality of keywords across the pages, i.e. how query keywords are distributed to pages in a subgraph. In that framework, subgraphs where

multiple query keywords appear together in many pages is ranked higher than subgraphs where keywords appear disjointly. It is because those keywords seem more strongly related to each other in the former case.

In this paper, we revise the ranking method proposed in [9] so that it reflects the size of subgraphs. In addition, we extend the concept of locality. In our new ranking method, we examine not only the locality of words across pages but also the locality of words within each page. If two query keywords appear at positions far from one another in a very long page, those two keywords may not be related at all. It often happens especially in “link collection” pages. Therefore, we rank these cases lower than those where the positions of keywords are close to each other.

In this paper, we also introduce a technique to distinguish links working as browsing routes within a single document and other kind of links. The technique we use in this paper is a simplified version of the techniques developed in [21]. By this technique, we improve the efficiency and accuracy of the automatic information unit detection. [25] also discussed that all links are not equally useful and that we should use link information selectively by distinguishing various link types. In this paper, we use the concept of “route links” introduced in [21] to improve the detection of logical documents in Web.

3 DISCOVERY OF LOGICAL INFORMATION UNITS

In this section, we explain each step in the detection of logical information units.

3.1 Route Links and Non-route Links

The first thing we do for the detection of information units in Web data is to distinguish links that are meant to be the standard browsing routes through which all readers should navigate. We call those links *route links*. We distinguish them because we consider a set of pages that is meant to be one logical document is always connected by that kind of links.

In this research, we use directory structure encoded in URLs to distinguish route links. We adopt the following heuristics:

- A link going from a page to a page in a subdirectory may be a route link.
- A link going from a page to a page in an upper directory is never a route link.
- A link going from a page to an index page, i.e. a

page named `index.*` or a URL ended by `/`, in the same directory is never a route link. Otherwise, a link between pages in a same directory may be a route link.

- A link going from a page to a page in an incomparable directory can be a route link only when those two directories are siblings, i.e. immediate subdirectories in the same directory.
- A link going from a page to a page on a different Web server may be a route link in some cases [21], e.g. when a logically single Web site is physically divided into multiple servers such as `www.acm.org`, `www1.acm.org`, and so on. In this research, however, we assume that a logical information unit never span multiple Web servers. Therefore, in this research, we assume links across different Web servers are never route links.

These are modified version of the assumptions adopted in [21]. We modified them in two ways. First, we are more conservative here because in this research we want to exclude only links that are clearly not a route link. Second, in this research we want to detect route links in order to reduce the cost of detection of logical information units. As explained in a later section, in our current implementation, we incrementally retrieve pages starting at URLs given by other existing search engines. We distinguish non-route links in order to reduce the time cost by not retrieving pages linked via non-route links. Therefore, we have to decide whether a link can be a route link or not before retrieving its destination page. The heuristics above were simplified so that they use only URLs of source and destination pages of links.

3.2 Extraction of Minimal Subgraphs

Next step is to extract minimal subgraphs including all the given query keywords. In our prototype implementation, we extract them by the following procedure:

1. We transform the query of the form $Q = k_1 \wedge \dots \wedge k_m$ into a disjunctive query $Q' = k_1 \vee \dots \vee k_m$, and submit Q' to an existing search engine. Currently, we are using `goo` [7], which is the most popular search engine in Japan.
2. We traverse links one step deeper starting from every page returned by that query. We traverse only links that are determined as route links by the method explained above.
3. After we have retrieved all pages linked from the pages in the queue, we examine if any subgraph includes all

the query keywords.

4. When we have found enough number of answers, we stop the traverse. Otherwise go back to Step 2.
5. In many cases, there are subgraphs that have exactly the same set of nodes including the query keywords, but differs only in nodes with no query keywords. We exclude those subgraphs but the one with the highest rank. The method to compute rank is explained in the next subsection.

3.3 Ranking Method

Next step is to sort found minimal subgraphs in order of scores calculated based on the localities of query keywords. In this paper, we propose the following function F to calculate the score of a subgraph G :

$$F(G) = \sum_{v \in V} (K(v) + C)^{-1}$$

where V is a set of pages in G , and $K(v)$ is the number of query keywords appearing in the page v . C is a parameter to adjust the cost value, which will be explained later.

This function sums up the cost of every page in the subgraph G . Subgraphs with smaller value of $F(G)$ are ranked higher. The cost of each page is $(K(v) + C)^{-1}$. The larger the number of keywords in the page is, the smaller the cost of that page is.

We can also interpret it as consisting of two factors as below:

$$F(G) = n \cdot \frac{1}{n} \sum_{v \in V} (K(v) + C)^{-1}$$

where n is the number of pages in V . From this interpretation, we can see the following properties of $F(G)$:

- When the second factor is constant, $F(G)$ is proportional to n , i.e. the size of the subgraph.
- When the size of the subgraph is constant, $F(G)$ is proportional to the second factor, the average of the inverse of $(K(v) + C)$, which is, very roughly speaking, inversely proportional to the average of $K(v)$.
- Moreover, the following inequality holds:

$$\frac{1}{n} \sum_{v \in V} (K(v) + C)^{-1} \geq \left\{ \frac{1}{n} \sum_{v \in V} (K(v) + C) \right\}^{-1}$$

That is, the average of the inverse of $(K(v) + C)$ is greater than or equal to the inverse of the average of









Minimal Subgraph	Current Formula ($C=1$)	Current Formula ($C=3$)	Previous Formula
(1) 	0.250	0.167	0.333
(2) 	0.667	0.400	0.500
(3) 	0.833	0.450	0.750
(4) 	1.167	0.650	0.667
(5) 	1.333	0.700	0.833
(6) 	1.667	0.733	1.333
(7) 	1.500	0.750	1.000
(8) 	1.833	0.783	1.500

Figure 1: Examples of $F(G)$

it. Equality holds only when $K(v_i) = K(v_j)$ for all $v_i, v_j \in V$. Generally, when the average of $K(v)$ is constant, the larger the variation of $K(v)$ is, the larger the average of the inverse of $(K(v) + C)$ is, and therefore, so is $F(G)$.

- The larger C is, the stronger the effect of the size of the subgraph on $F(G)$ is. On the contrary, the smaller C is, the stronger the effect of $K(v)$ is. Particularly, the effect of the variation of $K(v)$ becomes more significant. When $C = 0$, subgraphs including a page with no keyword are eliminated even if other pages include many keywords because the cost of a page with no keyword is infinite.

On the other hand, $F(G)$ does not reflect the shape of the graph at all. For example, a subgraph consisting of a root page and its two children and a subgraph consisting of three pages organized into a sequence have the same score as long as $K(v)$ of those three pages are equal. The reason of this design decision is that the shape of the graphs depend on the organization style of authors, and we cannot tell which shape is better. For example, the same document may be organized into one root page and its many children by some author, and be organized into a sequence by other author.

Figure 1 shows the values of $F(G)$ for various subgraphs. Here, we show two cases, $F(G)$ with $C = 1$ in

the second column, and $F(G)$ with $C = 3$ in the third column. In the fourth column, we also show the score of each subgraph computed by the formula proposed in [9]. It was the formula below:

$$F(G) = \frac{1}{n} \sum_{v \in V} (\min(K(v), \frac{1}{n}))^{-1}$$

(In [9], the inverse of the formula above was used, and subgraphs with larger $F(G)$ was ranked higher. In this comparison, however, we use the inverse of it in order to make it easier to compare the result of two formulae.) In the formula proposed in [9], its score does not properly reflect the size of the graph (e.g. see (3) and (4)). In our new function, however, the size of the graph is the primary factor.

In Figure 1, subgraphs are sorted in order of:

1. the number of pages,
2. the average of $K(v)$, and
3. the variation of $K(v)$.

When $C = 3$, the order of the cost is exactly the same order. When $C = 1$, the order of (6) and (7) is reversed. It is because when C is small, the effect of pages with small number of keywords is strong.

$F(G)$ reflects the distribution of the locality of keywords across pages in the subgraph. When keywords are distributed to many pages, n becomes large. When n is constant, the more the localities of keywords overlap, the larger the average number of $K(v)$ is, and therefore, the smaller $F(G)$ is. (See (2) and (3), or (4), (5), and (7) in Figure 1.) When the average of $K(v)$ is constant, subgraphs where keywords are continuously overlapping in every page is ranked higher than subgraphs which can be divided into two (or more) subgraphs with many keywords and pages between them with few keywords. (See (5) and (6), or (7) and (8) in Figure 1.)

3.4 Appearance Density of Words

Next, we examine whether two words appearing in the same page are really related to each other. Even when two words appear in the same page, sometimes these words are not related at all. For example, suppose there is a very long “links page” listing many links and brief explanations for them. If a query keyword appears at

somewhere near the top of that page, and another query keyword appears near the bottom of that page, those two keywords must not be related strongly. We want to rank such pages low.

To measure how strongly words in the same page are related, we use *appearance density* [15] of those words. An appearance density of a word at a position is calculated based on the frequency of the word within a “window” around that position. We regard it as reflecting the importance, or the degree of the influence, of the word at the position.

To calculate appearance density of words, we need to select a window function. We consider that a document is a sequence of words, and suppose L denotes the length of the sequence. Window function defines the weight value from 0 to 1 for each position from 1 to L in the sequence. The most simple window function is a rectangular window function which gives 1 to every position within a window of some fixed width and gives 0 elsewhere. In this paper, we use Hanning window function. $h_l(i)$, Hanning window function centered at l , is defined by the following formula:

$$h_l(i) = \begin{cases} \frac{1}{2}(1 + \cos 2\pi \frac{i-l}{W}) & |i-l| \leq \frac{W}{2} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where W is the width of a window (a range where a non-zero weight is given). $h_l(i)$ gives its maximum value 1 at the center of the window, i.e. at $i = l$, and it gets smaller as i gets far from the center.

$d_t(i)$, the appearance density of a word t at a position i using Hanning window function, is defined as follows:

$$d_t(i) = \sum_{j=1}^L h_i(j) \cdot a_t(j) \quad (2)$$

where $a_t(i)$ is defined as below:

$$a_t(i) = \begin{cases} 1 & \text{the word } t \text{ appears at the position } i \\ 0 & \text{otherwise} \end{cases}$$

$d_t(i)$ sums up the weight values of all positions where the word t occurs. We consider that it expresses the importance the word t has at the position i , and that the area where $d_t(i)$ is high is the *locality* of the word t .

Figure 2 shows examples of $d_t(i)$ for t_1 . In the document at the top, t_1 occurs twice, and the distance between them is longer than the width of the window.

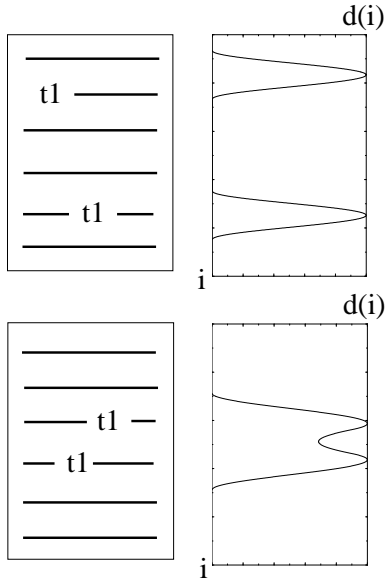


Figure 2: Examples of $d_t(i)$

$d_{t_1}(i)$ in that document has two separate part of the form of the window function as shown at the right of the document. In the document at the bottom, t_1 occurs twice, and the distance between them is shorter than the width of the window. $d_{t_1}(i)$ in that document is shown at the right of the document. In this case, the two parts corresponding to two occurrences overlaps, and the graph is sum of two graphs of the window function.

In order to measure the importance relative to other positions in that document, we normalize $d_t(i)$ so that the peak of $d_t(i)$ within a document be 1. We define relative appearance density $\tilde{d}_t(i)$ as follows:

$$\tilde{d}_t(i) = \frac{d_t(i)}{\max_{1 \leq j \leq L} d_t(j)}$$

Now we define $r(t_1, t_2)$, the degree of the interrelation of two words t_1 and t_2 , by the formula below:

$$r(t_1, t_2) = \max_{1 \leq i \leq L} \min(\tilde{d}_{t_1}(i), \tilde{d}_{t_2}(i))$$

Figure 3 shows an example of $r(t_1, t_2)$. In that graph, the dotted line represents $d_{t_1}(i)$ and the normal line represents $d_{t_2}(i)$. Then $r(t_1, t_2)$ is the value of $d_{t_1}(i)$ at the rightmost peak, which is the highest peak among those that are within $d_{t_2}(i)$.

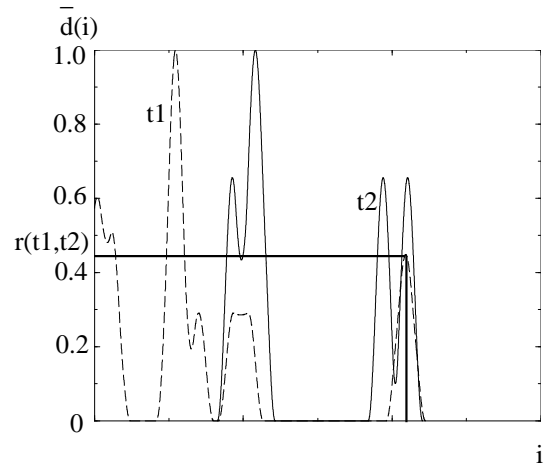


Figure 3: An Example of $r(t_1, t_2)$

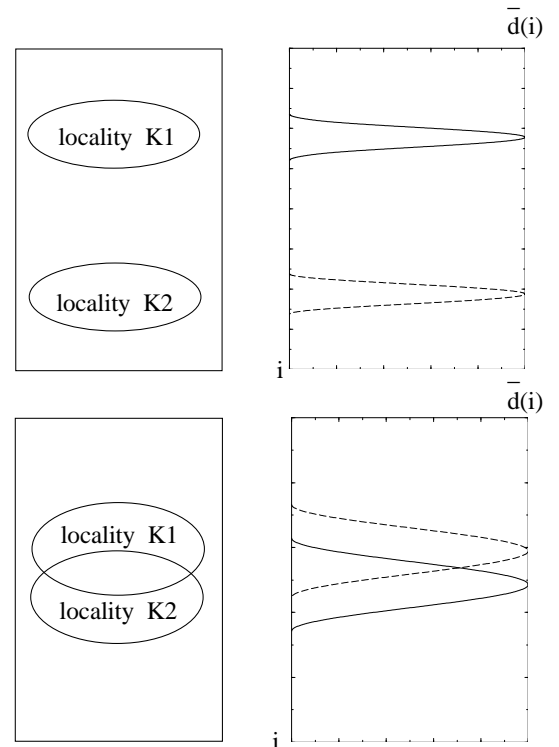


Figure 4: Examples of related words and unrelated words

When the localities of two words do not overlap at all as in the example at the top of Figure 4, $r(t_1, t_2) = 0$. On the other hand, when the localities of two words greatly overlap, $r(t_1, t_2)$ has a large value as in the example at the bottom of Figure 4.

In Web pages, however, words in a title tag, or the main heading at the top of a page have their influence all through the page even if they do not appear in the rest of the page. To give special treatment to these words, we define $\tilde{d}_t(i) = 1$ for any i if t is in a title tag or in a main heading of a page. Title tags can easily be detected. On the other hand, in order to detect the main headings of pages, we use a Perl module HTML::Parse in HTML-Tree, which is a free software library distributed by libwww-perl [18].

In order to make $F(G)$ reflect the locality of keywords within pages, we redefine $F(G)$ as below:

$$F(G) = \sum_{v \in V} (A \cdot K(v) + B \cdot \sum_{t_i, t_j \in T(v)} r(t_i, t_j) + C)^{-1}$$

where $T(v)$ is the set of query keywords appearing in v .

Now the formula has three parameters A , B , and C . Coefficient A and B define the weight given to $K(v)$ and $r(t_1, t_2)$. Currently we are testing this formula by experiment, and finding the best values for A , B , and C . The evaluation of the real effectiveness of this formula is a future issue.

3.5 Preliminary Experimental Results

We tested the following queries with 2 or 3 keywords:

- Q1: notebook, card, catalog — whose intention is to collect all the product catalog pages of PC cards for notebook computers
- Q2: “Ryoichi Sano”, “Kobe University”— whose intention is to find the homepage of Ryoichi Sano, who is a student of Kobe University. (Both “Ryoichi Sano” and “Kobe University” are treated as one word in Japanese.)

The result of Q1 is shown in Table 1. This table shows:

- the number of returned subgraphs of size n ,
- the number of correct subgraphs among the returned ones of size n ,
- the ratio of the number of correct ones of size n to the returned ones of size n ,
- the ratio of the number of correct subgraphs of size less than n to the number of returned ones of size less than n , and

- recall ratio relative to $n \leq 5$, i.e. the ratio of the number of correct subgraphs of size less than n to the number of correct subgraphs of size less than 5.

for n from 1 to 5.

For Q1, very large percentage of correct answers are consisting of one page. Therefore, they can be retrieved by a simple page-based conjunctive query. In Web sites of some companies, however, the word “catalog” does not appear in each catalog page but appear only in the root pages of those catalog pages. This kind of pages are retrieved as subgraphs of size 2 or 3. No subgraphs of size larger than 4 are retrieved.

Although the minimal subgraph approach does not significantly improve the recall ratio for Q1, it does not significantly reduce the precision ratio either. Therefore, when we want to create “complete” list of catalog pages of PC cards, introducing minimal subgraph query is meaningful.

Table 2 shows the result of Q2. The homepage of Ryoichi Sano cannot be retrieved by a simple page-based query because:

- in the member list page of our research group, the word “Ryoichi Sano” appears but the word “Kobe University” does not,
- the homepage of Ryoichi Sano is divided into a main frame and a sub-frame, and in these frames, the word “Ryoichi Sano” appears but the word “Kobe University” does not, and
- there is a separate page describing the profile of Ryoichi Sano, which is linked from the sub-frame of his homepage, and that page includes the word “Kobe University” but not the word “Ryoichi Sano.”

As a result of it, the homepage of Ryoichi Sano is retrieved by Q2 as a subgraph of size 4 (the member list page, two frames of homepages, and the profile page).

As far as we see the result of those example queries, the minimal subgraph approach seems useful for the following kinds of queries:

- recall-oriented queries, i.e. queries where we want “complete” list of relevant pages, and
- queries to retrieve very specific page which must be somewhere but whose URL is unknown, for example the homepage of xxx. In this case, if some keywords which seem very appropriate for query happen to occur only in neighboring pages, minimal subgraph approach

Table 1: Precision and Recall Ratios for Q1

n (the size of subgraphs)	1	2	3	4	5
the number of returned answers	980	27	21	0	0
the number of correct answers	682	17	12	0	0
precision ratio (within each n)	0.696	0.629	0.571	–	–
precision ratio (accumulative)	0.696	0.694	0.691	0.691	0.691
recall ratio (relative to $n \leq 5$)	0.959	0.983	1.000	1.000	1.000

Table 2: Precision and Recall Ratios for Q2

n (the size of subgraphs)	1	2	3	4	5
the number of returned answers	0	0	1	1	0
the number of correct answers	0	0	0	1	0
precision ratio (within each n)	–	–	0.000	1.000	–
precision ratio (accumulative)	–	–	0.000	0.500	0.500
recall ratio (relative to $n \leq 5$)	0.000	0.000	0.000	1.000	1.000

is very useful.

4 DISCUSSION

In this section, we discuss the semantics of conjunctive queries for Web retrieval and the rationality of our approach.

In most search engines, there is only one form of conjunctive queries: a list of keywords in order of importance. Conjunctive queries with multiple keywords t_1 , t_2 , and t_3 are, however, used in various intentions such as:

- **also** query: intending to retrieve pages discussing t_1 , and also t_2 , and also t_3 ,
- **and** query: intending to retrieve pages discussing “ t_1 and t_2 and t_3 ”, and
- **of** query: where there is a clear order among keywords from general terms to specialized terms, and the keywords narrow down a topic stepwise in that order, like “ t_1 of t_2 of t_3 .”

The class of **also** queries is the most general class including all conjunctive queries. A typical example of **also** query is a query with keywords “Aug. 1999” and “ACM”, whose intention is to retrieve homepages for any events in July 1999 sponsored by ACM. In this query, July 1999 and ACM are not directly related to one another. They are independently narrowing down the topic from different aspects.

and queries are the special case of **also** queries, where

the keywords are related to each other, and are cooperatively describing the topic of the interest. An example of **and** query is a query with keywords “Web” and “visualization”, whose intention is to retrieve pages discussing a topic related to both Web and visualization techniques, e.g. visualization of Web data, or visualization of some kind of data using Web environments.

of queries are special cases of **and** queries, where there is a clear order among keywords from general terms to specialized terms. An example of **of** query is a query with keywords “Workshop”, “SIGWEB”, and “ACM”, whose intention is to retrieve homepages of workshops sponsored by SIGWEB of ACM.

There are also combinations of those three kind of conjunctions. In addition, those three kinds of conjunctions cannot always be distinguished clearly. There are mediums of some two.

The appropriate criterion for ranking retrieved query answers varies depending on the type of queries. In this research, we use the size of subgraphs and overlap of locality of query keywords. We use the overlap of locality of query keywords because we mainly focus on **and** queries, where keywords should be related to each other in correct answers. If we focus on **also** queries, the overlap of locality of query keywords are not necessarily important, and the size of subgraphs should be the only important factor for ranking. If we

focus on **of** queries, the shape of graphs and how query keywords appear in the subgraph (e.g. the order) may be additional meaningful information for better ranking. The verification of these idea, the support of more than one kind of conjunctive queries and providing different ranking functions for them, with experiments is a future issue.

As discussed above, when we focus on **and** queries, relation between the occurrences of query keywords is very important. When we use subgraphs as the logical information unit for retrieval, relation between keywords and anchors pointing to other pages where other keywords occur is also very important. For example, if a keyword occurs in the anchor string of some link, and another keyword occurs in the page pointed by that link, then those two keywords must be strongly related in that document. There are a couple of researches that proposed to include words in anchor strings into indexes of pointed pages [20, 3, 17, 21]. When we introduce the minimal subgraph approach, we also need to examine the relation between the pointed page and all the occurrences of query keywords in the source page of the anchor. The method explained in the subsection 3.4 must easily be extended to include the relation between keywords and anchors. This is another future issue.

5 CONCLUSION

In this paper, we developed a couple of new techniques for the retrieval of logical information unit in Web data based on the minimal subgraph approach. First, we proposed the concept of route links, which are links that are meant to be the standard browsing routes within a single information unit, and developed a method to distinguish them from other kinds of links. Second, we designed a new formula for the ranking of minimal subgraphs including all the given query keywords. It ranks subgraphs based on the distribution of query keywords within subgraphs. Third, we proposed a extension to that formula which takes into consideration not only distribution of keywords within a subgraph but also distribution of keywords within a page. For the first and the second one, we showed simple experiments. For the third one, the verification of the idea with experiments is a future issue.

REFERENCES

1. Rodrigo A. Botafogo, Ehud Rivlin, and Ben Shneiderman. Structural analysis of hypertexts:

Identifying hierarchies and useful metrics. *ACM TOIS*, 10(2):142–180, Apr. 1992.

2. Rodrigo A. Botafogo and Ben Shneiderman. Identifying aggregates in hypertext structures. In *Proc. of ACM Hypertext*, pages 63–74, Dec. 1991.
3. Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proc. of 7th Intl. WWW Conf.*, Apr. 1998.
4. Jeromy Carrière and Rick Kazman. WebQuery: Searching and visualizing the web through connectivity. In *Proc. of 6th Intl. WWW Conference*, Apr. 1997.
5. W. Bruce Croft and Howard R. Turtle. A retrieval model for incorporating hypertext links. In *Proc. of ACM Hypertext*, pages 213–224, Nov. 1989.
6. David Gibson, Jon Kleinberg, and Prabhakar Raghavan. Inferring web communities from link topology. In *Proc. of ACM Hypertext*, pages 225–234, Jun. 1998.
7. <http://www.goo.ne.jp>.
8. Masanori Harada, Shin-ya Sato, and Kazuhiro Kazama. Estimation of hierarchical structure of Web for search engine. *IPSJ SIG Notes 99-DBS-118/98-FI-54*, pages 105–112, May 1999. In Japanese.
9. Kenji Hatano, Ryouichi Sano, Yiewi Duan, and Katsumi Tanaka. An interactive classification of Web documents by self-organizing maps and search engine. In *Proc. of 6th Intl. Conf. on Database Systems for Advanced Applications (DASFAA'99)*, pages 35–42. IEEE Computer Society Press, Apr. 1999.
10. Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. In *Proc. of ACM SIGIR*, pages 59–68, Jun. 1993.
11. Masatsugu Kitagawa, Yoshiaki Mizuuchi, Keishi Tajima, and Katsumi Tanaka. Clustering and cut detection for information organization of mailing list. In *Proc. of Data Engineering Workshop (DEWS'97)*, pages 221–226. IEICE, Mar. 1997. In Japanese.

12. Jon Kleinberg. Authoritative sources in a hyper-linked environment. In *Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 668–677, Jan. 1998.
13. Nobuyuki Kobayashi and Fumio Kitagawa. Finding a page-set in the WWW document and its application to search engines. In *Proc. of Data Engineering Workshop (DEWS'99)*. IEICE, Mar. 1999. In Japanese.
14. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the web for emerging cyber-communities. In *Proc. of 7th Intl. WWW Conf.*, Apr. 1998.
15. Sadao Kurohashi, Nobuyuki Shiraki, and Makoto Nagao. A method for detecting important descriptions of a word based on its density distribution in text. *Transactions of Information Processing Society of Japan*, 38(4):845–854, Apr. 1997. In Japanese.
16. Wen-Syan Li and Yi-Leh Wu. Query relaxation by structure for Web document retrieval with progressive processing (extended abstract). In *Proc. of Advanced Database Symposium*, pages 19–25. IPSJ, Dec. 1998.
17. Yanhong Li. Toward a qualitative search engine. *IEEE Internet Computing*, 2(4):24–29, July/August 1998.
18. <http://www.linpro.no/lwp/>.
19. Massimo Marchiori. The quest for correct information on the web: Hyper search engines. In *Proc. of 6th Intl. WWW Conference*, Apr. 1997.
20. Oliver A. McBryan. GENVL and WWW: Tools for taming the web. In *Proc. of 1st Intl. WWW Conf.*, may 1994.
21. Yoshiaki Mizuuchi and Keishi Tajima. Finding context paths for Web pages. In *Proc. of ACM Hypertext*, pages 13–22, Feb. 1999.
22. Yoshiaki Mizuuchi, Keishi Tajima, and Katsumi Tanaka. Retrieval of graph structured data based on cut partitioning. *IPSJ SIG Notes 97-DBS-113*, pages 281–286, Jul. 1997. In Japanese.
23. Takuhiro Nagafuji and Motomichi Toyama. A WWW search engine based on disjoint page groups in dividing WWW hyper-text space. In *Proc. of Data Engineering Workshop (DEWS'98)*. IEICE, Mar. 1998. In Japanese.
24. Peter Pirolli, James Pitkow, and Ramana Rao. Silk from a Sow's ear: Extracting usable structures from the web. In *Proc. of ACM SIGCHI Conf.*, pages 118–125, Apr. 1996.
25. Ellen Spertus. ParaSite: Mining structural information on the web. In *Proc. of 6th Intl. WWW Conference*, Apr. 1997.
26. Keishi Tajima, Yoshiaki Mizuuchi, Masatsugu Kitagawa, and Katsumi Tanaka. Cut as a querying unit for WWW, Netnews, and E-mail. In *Proc. of ACM Hypertext*, pages 235–244, Jun. 1998.
27. Ron Weiss, Bienvenido Vélez, Mark A. Sheldon, Chanatip Namprempre, Peter Szilagy, Andrzej Duda, and David K. Gifford. HyPursuit: A hierarchical network search engine that exploits content-link hypertext clustering. In *Proc. of ACM Hypertext*, pages 180–193, 1996.