http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 8 (4): 427-440, 2009
ISSN 1812-5638

Finding Related Web Pages in Parallel by Using Grouped Link Structures

Shen Xiaoyan, Chen Junliang, Meng Xiangwu and Zhang Yujie
State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing, 100876, China

Abstract: In this study, a block co-citation algorithm is proposed to find related pages for a given web page in two steps. First, all hyperlinks in a web page are segmented into several blocks according to the HTML structure and text style information. Second, for each page, the similarity between every two hyperlinks in the same block is computed. Then the total similarity from one page to the other is obtained after all web pages are processed. For a given page u, the pages which have the highest total similarity to u are selected as the related pages of u. The block co-citation algorithm was implemented in parallel to analyze a corpus of 37, 482, 913 pages sampled from a commercial search engine and demonstrate its feasibility and efficiency. Experimental results for 28 pages pertaining to 7 topics indicated that the performance of the block co-citation algorithm is superior to traditional co-citation algorithm. This method is very suitable for application in commercial search engines.

Key words: Related pages, co-citation algorithm, HTML segmentation, parallel, scalable

INTRODUCTION

With the rapid growth of Internet resources, it is increasingly difficult to efficiently find information on one topic. Traditional methods use keywords as queries to search for information, but once a person find a web page that contains needed information, he would like to search for more pages that address the same topic as the original page. For this purpose, finding more pages related to the original page is often more effective than trying to modify the query. Therefore, searching for related pages has become an important function in some commercial search engines via., links called find-similar or similar pages.

Since several factors can affect the similarity of web pages and different aspects of web pages affect similarities between pages in different manners (Tombros and Ali, 2005), finding related pages is not an easy job. Many studies have been conducted to evaluate the similarity of two pages efficiently and accurately; the current algorithms for finding related pages are divided into three categories.

The first category is content-based algorithms which compute page similarities based completely on page contents (Loia *et al*., 2002; Fan *et al*., 2004). However, these algorithms ignore the available hyperlink data and are subject to infection by spam pages. Content-based methods generally require large storage and long computing time, so they are not easily applied to large corpora. Progress in this area has been slow.

The second category is link-based algorithms, which use connectivity information to evaluate the similarity between web pages (Dean and Henzinger, 1999; Tsuyoshi, 2001; Asano *et al*., 2004; Hou and Zhang, 2003; Ubaldo and Huang, 2003; Huang *et al*., 2004; Chirita *et al*., 2004; Brin and Page, 1998; Ollivier and Senellart, 2007; Jeh and Widom, 2002; Fogaras and Racz, 2007; Lin *et al*., 2006; Liben-Nowell and Kleinberg, 2003). Although link-based algorithms are generally more resistant to scamming, pages that are related to other pages which were published later with few in-links may not have sufficient information to be discovered by these algorithms. Furthermore, they ignore the differences between hyperlinks within a page and treat a web page as a single node. However, web pages can be subdivided into several smaller sections, each of which may contain some useful semantic information on a common topic. We refer to such sections as blocks. Since, blocks in a page may cover multiple topics that are not relevant to one another, the relationship between links in the same block may be stronger than the relationship between links in different blocks. Moreover, pages in the same domain may have several template blocks; these blocks are totally identical and links in them may also drastically distort the link analysis result.

The most famous related pages finding service is provided by Google, as shown in Fig. 1. The method they adopted is a revised co-citation algorithm. Two web pages are similar if they are linked from the same website. Factors that increase the similarity of two pages include the popularity of the site as well as the positions, sizes and proximities of the links.

Corresponding Author: Shen Xiaoyan, State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing, 100876, China Tel/Fax: 86-10-62247244

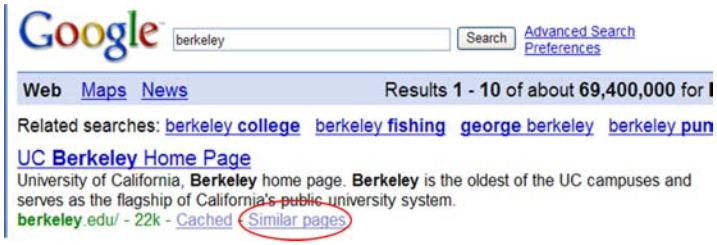

Fig. 1: Related pages finding service of Google

Fig. 2: A poor result from the Google similar pages service

It is very interesting that www.miibeian.gov.cn is selected as a similar page result for several famous websites by Google as Fig. 2 shows. Since, almost all websites in China put this link at the bottom of pages to identify copyright, www.miibeian.gov.cn can easily be selected as a related page for its neighbor links when using a traditional co-citation algorithm. If we split the copyright link and its neighbors into different blocks and limit the similarity contributed by template links in the same domain, these bad cases should not appear in related page results any more.

The third category is anchor-based algorithms which use the similarity between texts near anchors, denoted as the anchor window, to find related pages (Chakrabarti *et al*., 1998a, b). Using the anchor window, which often has a topic summary of the linked target document, can avoid time and space costs in the content analysis process.

Some research has focused on combining multiple types of algorithms to improve the precision (Haveliwala *et al*., 2002; Zhu *et al*., 2003; Sadi *et al*., 2006). Since, these methods can combine the advantages of several algorithms, we think that hybrid methods may be more effective to find related pages.

In this study, we propose a block co-citation algorithm to find related pages. This is also a hybrid method and considers several factors. After segmenting a page into blocks, we revise the co-citation algorithm to only consider link pairs within the same block. In addition, we also take the anchors of each link into account to modify the similarity of each pair. Furthermore, although parallelization is a key property of related pages algorithms for application, few research studies take it into account.

THE HTML SEGMENTATION FLOW

Several HTML segmentation algorithms have been proposed; they can be divided into two categories

(Xin *et al*., 2006): vision-based methods and non-vision based methods.

Cai *et al*. (2003) introduced a vision-based algorithm called VIPS. The algorithm segmented a page based on its visual characteristics, identified horizontal spaces and vertical spaces and delimited blocks much as a human being would visually identify semantic blocks in a page. Although, the precision of the algorithm is very high, it needs to use the pixel information of separators which can only be obtained with the assistance of web browsers and may not be easily applied to large corpora. We abandon vision-based methods for this reason.

The simplest yet most powerful non-vision based method was proposed by Debnath *et al*. (2005). A list of partitioning tags was used to identify the boundary between HTML sections. Although, their method is clear and performs well, we could not apply it directly for the following reasons:

- Since only the hyperlinks and anchor texts in pages are useful for block co-citation algorithm at present, our only concern in segmentation is how to group together the links belonging to the same topic or type and split the links which are unrelated to one another. The text contents of a page are only clues to help us split links, so the precision of text segmentation is not very important
- Obviously, the granularity of blocks needs to be sufficiently large to get enough co-citation link pairs; using only HTML tag to segment pages may generate too many small blocks for us

For these reasons, we the segmenting algorithm in (Debnath *et al*., 2005) as a first step and further process its results, which we will call sections. A block is defined as a list of links from a page. All links in a block should pertain to the same topic. The links in a block are indexed by their order in the original page. The size of a block is the number of links it contains. Present segmenting algorithm includes 2 stages:

Scanning stage: Given a page, we scan the content and trace its original DOM structure. The DOM is a W3C (World Wide Web Consortium) standard (Philippe, 2002) and defines a standard for accessing documents like HTML and XML. The DOM presents an HTML document as a tree-structure (a node tree), with elements, attributes and text. To cope with common errors in HTML documents, we add missing parent elements, close elements with optional end tags and handle mismatched in line element tags in this stage. Then we use the GetBlocks routine (Debnath *et al*., 2005) to segment the DOM tree into sections. Besides the texts and links in each section, we also save some extra information such as the font and text tags (<h1>, <h2>, etc.) the text background color, etc., to help the adjusting stage recognize similar neighbor sections.

Adjusting stage: In order to avoid generating many trivial blocks, we merge neighboring sections according to heuristic rules derived by examining a few thousand web pages from various sources and formats. Furthermore, sections which contain invalid links are removed automatically. Finally, all links in each section are extracted to construct a block.

For two arbitrary neighbor sections S and T, the following rules are used sequentially to decide whether they can be merged. Each rule is used only if no earlier rule applied.

- If the ratio of anchor words to total words in the section exceeds a threshold T_1 and the number of links in the section is larger than another threshold T_2, the section type is link; otherwise it is non-link. If S and T are link sections and have the same background color, they are merged
- S and T cannot be merged if either section contains boldfaced text
- S and T cannot be merged if their text font, font size, or font color are different. We combine these factors into a formula to quantify this difference. For a sections, the average font score afs (s) is defined as:

$$afs(s) = \frac{\sum_i n_i \times b_i \times edis_i}{\sum_i n_i} \quad (1)$$

where, n_i is the number of non-white space characters of the ith HTML text element here; when characters are in bold type, b_i is set to 1.2, otherwise, b_i is set to 1; $edis_i$ is the normalized euclidean distance (Jain, 1989) between the font color and the background color in CIELAB color space

Based on this observations, when the difference between the average font score of two sections is larger than 0.5, the sections should not be merged.

- S and T cannot be merged if both blocks contain more than 10 words.
- When the types of S and T are different, but they can be merged into a section with the same type as the larger of S and T, they are merged.

```
Algorithm 1: adjust
    Input: List S of sections after scanning for page p
    Output: List B of blocks for page p
    begin
        s1 = first(S), s2 = next(S);
        while s2 ≠ null do
            if canMerge(s1, s2) then
                s1 = merge(s1, s2);
            else
                if is Valid(s1) then
                    B.add(getBlock(s1));
                end
                s1 = s2;
            end
            s2 = next(S);
    end
end
```

Fig. 3: The adjust algorithm

- If S and T do not match any of the previous rules, they cannot be merged

Figure 3 shows the flow of the adjusting algorithm. The first function returns the first element of a list and the next function returns the subsequent element. The can Merge routine takes two sections as input and judges whether the sections can be merged according to the above rules. The merge routine merges both text and links from two sections into a new section. The is valid routine discards all sections which have only one link. The get block routine extracts all links in a section.

In order to validate the proposed segmenting strategy, we randomly sampled 56 pages from the experimental corpus. The links in each page were manually segmented into blocks according to topic. For each page P_i, let BA_i be the set of blocks generated using the algorithm; define BM_i as the set of blocks generated manually. The total number of blocks is expressed as:

$$NB = \sum_i |BA_i| \quad (2)$$

Two measures were used to evaluate the segmentation precision:

- E-precision is the fraction of the blocks in BA_i identical to a block in BM_i, format:

$$E-precision = \frac{\sum_i |\{b \mid b \in BA_i \cap \exists b'(b' \in BM_i \cap \forall l(\in b \leftrightarrow l \in b'))\}|}{NB} \quad (3)$$

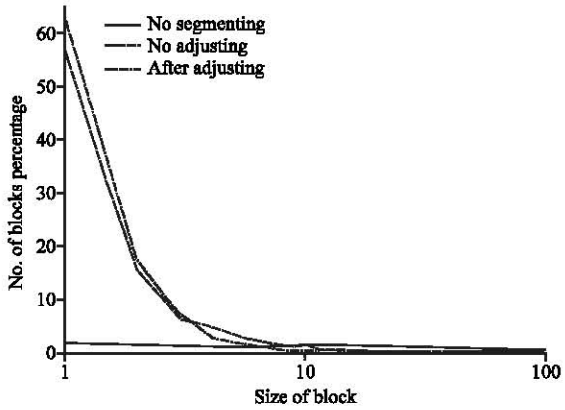

Fig. 4: Comparison of block size

Table 1: Comparison of segmenting precision

Algorithm	No. of blocks	E-precision (%)	I-precision (%)
By human	1194	100.0	100.0
Before adjusting	6628	9.6	98.9
After adjusting	3317	29.7	93.3

where, b is a block and l is a link in a block

- I-precision is the fraction of the blocks in BA_i included in a block in BM_i

$$I-precision = \frac{\sum_i |\{b \mid b \in BA_i \cap \exists b'(b' \in BM_i \cap \forall l(\in b \rightarrow l \in b'))\}|}{NB} \quad (4)$$

The statistical results are shown in Table 1. Both the original algorithm and this algorithm achieved high I-precision, so most links in each block belong to the same topic. The original algorithm generated more blocks; examination showed that several adjacent blocks generated by the original algorithm often corresponded to one block in the manual results. The adjusting stage merged many adjacent blocks and improved the E-precision. Besides larger blocks, we can find more related page results per block.

Figure 4 compares the block size distributions. If we treat each page as a block (no segmentation), the block sizes are approximately uniformly distributed. The links in large blocks almost always belong to more than one topic. By adjusting after segmenting, many trivial blocks can be suppressed. We use the adjusting strategy to try to find blocks with links pertaining to the same topic, without generating too many trivial blocks.

FINDING RELATED PAGES

Before describing how to generate related pages, we define the following terms.

Definition 1: Two pages p_1 and p_2 are block co-cited once if they are both targets of two hyperlinks in the same block b of the same page p. The pair $<p_1, p_2>_b$ is called a related pair in block b. The similarity of $<p_1, p_2>_b$ is denoted by S_b ($<p_1, p_2>$). If more than one block in p cites both p_1 and $p_{,2}$ only one is counted to avoid p contributing too much to the pair similarity. A related pair generated from page p is denoted by $<p_1, p_2>_p$ and the similarity of $<p_1, p_2>_p$ is S_p ($<p_1, p_2>$). Let S_p ($<p_1, p_2>$) = 0 if there is no valid related pair between p_1 and p_2 from p.

Definition 2: The domain name of a page means the first level in the URL string associated with the page. For example, the domain name of the page en.wikipedia.org/wiki is wikipedia.org. In reality, almost all domain names end with one or more top-level domains governed by ICANN or another domain name registry, such as .com, .cn.com and so on. The top sub-domain name of page p is defined as the domain name directly under the leftmost top-level domain, denoted by TSD (p). For example, TSD (www.yahoo.com) is yahoo.com, since.com is a top-level domain governed by ICANN.

The flow of finding related pages takes all blocks as input and includes three key steps:

Filtering hyperlinks: So, as to avoid meaningless related pairs generated by insignificant links, all hyperlinks in a block are sequentially processed by the following filters. Each filter can remove or change links in the block. Only links remaining after all filter procedures generate valid related pairs:

- **Large block filter:** Discard blocks which contain more than S (block size threshold) links. When a block has too many links, the links are likely to pertain to several topics. For example, home pages of hubs may have some large blocks to recommend many famous web sites belonging to different topics.
- **File filter:** Discard all links pointing to a file, such as mp3 files, doc files, etc. All file links are removed because we only consider the relationship between two pages.
- **Dynamic filter:** Replace all dynamic links with their domain name, because too many dynamic links would introduce trivial noise and data sparsity problems.
- **Normalization filter:** Since, two syntactically different URLs may represent the same resource, such as www.sina.com/index.html and www.sina.com, the URL normalization technologies (Pant *et al*., 2004; Lee *et al*., 2005) are used to transform all links into their canonical forms.
- **Singleton filter:** Consider all links in a block at most once. Any link with the same URL format as a previous link will be deleted. This avoids inflating link similarity because of repeated citation within a block.
- **Empty anchor filter:** Consider all links with no anchor as a bookmark link or spam link and discard them.
- **Common domain name filter:** Remove any links in a block which have the same top sub-domain name as the parent page. This can avoid some web sites contributing too much similarity between their own pages.

Generating related pairs: Here, the similarity between every pair of links remaining in a block is computed and then related pairs are generated, along with their similarity in the current page and other useful information. Furthermore, after filtering the links in blocks, we only generate a related pair between two links with different top sub-domain names, because we do not want to include many related pairs within the same top sub-domain. If a block contains n links, it can be seen that at most C_n^2 related pairs will be generated.

For two pages p_1 and p_2 in block b of page p, two factors are considered in computing S_b ($<p_1, p_2>$).

- **Proximity factor:** Based on observation, a link l is more related to its close neighbors than to distant neighbors and the nearest neighbors of l may have equal similarity to l. With this assumption, the proximity of p_1 and p_2 in b is defined as:

$$P_b(< p_1, p_2 >) = \begin{cases} 1 & |i-j| <= \lambda \\ e^{\frac{-(|i-j|-\lambda)}{2}} & |i-j| > \lambda \end{cases} \quad (5)$$

  where i, j are the indices of p_1 and p_2 in b; the near neighbor threshold λ is set to differentiate near neighbors from distant neighbors
- **Anchor similarity factor:** We think two links sharing more matched words in their anchor texts are likely to be more related to one another. Based on this assumption, the anchor similarity of p_1 and p_2 in b is defined as the Jaccard coefficient between the words set of their anchor texts:

$$A_b(<p_1,p_2>)=\frac{|words(A_{b,p_1})\bigcap words(A_{b,p_2})|}{|words(A_{b,p_1})\bigcup words(A_{b,p_2})|} \quad (6)$$

where, $A_{b,p1}$ ($A_{b,p2}$) is the anchor text of p_1 (p_2) in block b and words(s) is the set of words in text s

For English anchor texts, all words are first normalized with the Porter Stemming algorithm (http://www.tartarus.org/~martin/PorterStemmer); the words set contains all distinct words after normalization. For Chinese anchor texts, the words set contains all Chinese words in the anchor output by a Chinese word segment algorithm, such as ICTCLAS (ictclas.org/index.html). All stop words with highest IDF in each language are removed before computing.

The similarity between p_1 and p_2 in b is computed as follows:

$$S_b(<p_1,p_2>)=P_b(<p_1,p_2>)*(1+A_b(<p_1,p_2>)) \quad (7)$$

Finally, S_p ($<p_1, p_2>$) equals the S_b ($<p_1, p_2>$) computed from the first block b which can generate $<p_1, p_2>_b$. In order to record the anchor texts used to calculate the similarity, we let $A_{p,p1}$ ($A_{p,p2}$) = $A_{b,p1}$ ($A_{b,p2}$).

In order to suppress template noise in the next step, for each related pair $<p_1, p_2>_p$, $A_{p,p1}$, $A_{p,p2}$ and TSD (p) are recorded along with S_p ($<p_1, p_2>$). The output of this step is a vector:

$$V_{p,<p_1,p_2>}=<p_1,p_2,TSD(p),A_{p,p_1},A_{p,p_2},S_p(<p_1,p_2>)> \quad (8)$$

Merging related pairs: For clear illustration, we first define the following concepts.

Definition 3: For a given top sub-domain name d and two arbitrary pages p_1 and p_2, the set of related pairs generated from pages belong to d is denoted by P_d ($<p_1, p_2>$) namely,

$$P_d(<p_1,p_2>)=\{<p_1,p_2>_p | TSD(p)=d \text{ and } S_p(p_1,p_2)>0\} \quad (9)$$

The set of top sub-domain names whose P_d ($<p_1, p_2>$) is not an empty set is denoted by D (p_1, p_2).

As shown in Eq. 8, each related pair in P_d ($<p_1, p_2>$) has an $A_{p,p1}$ and an $A_{p,p2}$. If we consider the values of $A_{p,p1}$ and $A_{p,p2}$ and group all related pairs with the same value of $A_{p,p1}$ or $A_{p,p2}$ together, we can construct two partitions of P_d ($<p_1, p_2>$):

$$Pl_d^n=\{Pl_d(A_{p_1}^1),Pl_d(A_{p_1}^2),...,Pl_d(A_{p_1}^n)\}$$
$$<p_1,p_2>_p\in Pl_d(A_{p_1}^i)\Leftrightarrow A_{p,p_1}=A_{p_1}^i \quad (10)$$

$$Pr_d^m=\{Pr_d(A_{p_2}^1),Pr_d(A_{p_2}^2),..,Pr_d(A_{p_2}^m)\}$$
$$<p_1,p_2>_p\in Pr_d(A_{p_2}^j)\Leftrightarrow A_{p,p_2}=A_{p_2}^j \quad (11)$$

where, A_{p1}^i and A_{p2}^j represent a value of $A_{p,p1}$ and $A_{p,p2}$, respectively.

Definition 4: For two arbitrary pages p_1 and p_2, the domain similarity from p_1 to p_2 is denoted by Ds_d ($p_1 \rightarrow p_2$) and vice versa.

Ds_d ($p_1 \rightarrow p_2$) is computed by summing up the similarities contributed by each page in P_d ($<p_1,p_2>$) with the following limitations:

- Page templates in a top sub-domain always have the same format and the anchor text of each link in these templates is a constant value. To avoid template links introducing noise to present results, for an anchor text of p_2 belong to d, A_{p2}^j, no more than the top r similarities of related pairs belonged to $Pr_d(A_{p2}^j)$ will be added to Ds_d ($p_1 \rightarrow p_2$) and so as Ds_d ($p_2 \rightarrow p_1$). r is called the anchor repetition frequency.
- Since, some malicious top sub-domain names provide too many similarities between two pages, one page may appear very related to another, when actually these two pages are not so related. Hence, we limit the maximum value of Ds_d ($p_1 \rightarrow p_2$) or Ds_d ($p_2 \rightarrow p_1$) to N, which is called the domain similarity threshold.

As described above,

$$DS_d(p_1 \rightarrow p_2)=\min(N,\sum_j \sum_{top\ r}(S_p(<p_1,p_2>) \text{ in } Pr_d(A_{p_2}^j)) \quad (12)$$

Definition 5: For two arbitrary pages p_1 and p_2, the total similarity from p_1 to p_2 is calculated as:

$$TS(p_1 \rightarrow p_2)=\frac{1}{1+\ln(in_{p_2})}\sum_{d\in D(p_1,p_2)} DS_d(p_1 \rightarrow p_2) \quad (13)$$

where, in_{p2} is the number of blocks which contain page p_2. Some generic pages such as www.google.com may have too many co-citation links with other pages. In order to avoid generic pages seeming very similar to other non-generic pages, in_{p2} is used to adjust the total similarity.

Finally, for two arbitrary pages p_1 and p_2, all related pairs containing them are grouped to compute TS ($p_1 \rightarrow p_2$) and TS ($p_2 \rightarrow p_1$), respectively. Then for page p_1, we choose no more than M pages with the highest TS ($p_1 \rightarrow p_i$) as the related pages of p_1. Basically, when TS ($p_1 \rightarrow p_i$) is higher, p_i is more related to p_1. We also limit the minimum value of TS ($p_1 \rightarrow p_i$) to T, which is called the total similarity threshold.

PARALLEL IMPLEMENTATION

Here, we discuss a parallel implementation of the steps in the previous section, inspired by the distributed computing model MapReduce (Dean and Ghemawat, 2004). MapReduce can break input data sets into several small partitions to be handled on several machines. Both the input and output data sets are several partitions of key/value pairs. The process of MapReduce is divided into two steps:

- The first step is a Map operation that transforms a partition of input pairs into an intermediate key/value pair list presentation:

$$\text{map } (<\text{key}_1, \text{value}_1>) \rightarrow \text{list } (<\text{key}_2, \text{value}_2>)$$

- The second step is a Reduce operation that recombines all the intermediate values corresponding to the same intermediate key and generates the final key/value pair list:

$$\text{reduce } (<\text{key}_2, \text{list } (\text{value}_2>) \rightarrow \text{list } (<\text{key}_3, \text{value}_3>)$$

Between the map and reduce procedures, a partition function is designed to divide the intermediate <key, values> pairs into several splits and ensure that all pairs with the same key are in the same split. Then key and value comparison functions are designed to sort the pairs in each split by the intermediate key and value. Finally all occurrences of the same key and the corresponding values are grouped together.

Figure 5 shows the MapReduce flow for finding related pages from page blocks. Before this step, there was another step that parsed web pages and extracted blocks. This step can also be implemented by the MapReduce model, but it is not shown in Fig. 5 in order to limit the paper length.

In the Mapper1 operation, we finish the task of filtering links and generating related pairs. The input key: p is the URL of a page, The input value: lb is the list of blocks segmented from p. The filter block routine filters all links in blocks using the filters described above. The cal Sim routine calculates the similarity of each related pair according to Eq. 7. The output routine outputs a <key, values> pair for the following steps. For each <key, value> pair output by Mapper1, the key p_{m1} is the URL of a link in page p, the value is v_{m1} a tetrad. In fact, the $V_{p,<p1, p2>}$ in Eq. 8 is divided into two key-value pairs in the output of Mapper1, as line 9~10 showed:

$$<p_1, <p_2, \text{TSD } (p), A_{p, p1}, S_p (<p_1, p_2>) >>,$$

$$<p_2, <p_1, \text{TSD } (p), A_{p, p2}, S_p (<p_1, p_2>) >>$$

Besides these items, we also output an auxiliary value <null, null, null, 0.0> for each key page p_i as line 5 showed, to help us to compute in_{pi} in the Reducer1 operation.

In the Sort1 operation, the comparison method of the URL keys simply compares URLs by alphabetic order and the null string is the smallest value in alphabetic order. The method to compare the output tetrad values of Mapper1 is sequential comparison: if the values of the first components are not equal, we output their comparison result; otherwise, we compare the second components and so on. We compare the 4th component reversely, so that tetrad value with higher similarity are processed earlier in Reducer1.

After the sort process is finished, all values for the same key will be grouped together. In these values, the auxiliary tetrad value must be ahead of all normal values; the normal values following the auxiliary value are grouped by their four components in order. This order can help Reducer1 compute the total similarity.

In Reducer1 operation, the number of auxiliary tetrad corresponding to each key page p_{m1} is first counted to compute in_{pm1} as line 3~6 showed. Then for each page last p related to p_{m1}, the TS (lastp→p_{m1}) is computed according to Eq. 12, 13 as line 8~30 showed. Finally, lastp is the key of a pair <p_{r1}, v_{r1}> output by Reducer1, <p_{m1}, TS (lastp→p_{m1})> is corresponding value as line 24 and 30 showed.

The Mapper2 and Reducer2 operations finish generating the relevant pages set for each page p_{r2}. Their implementation is quite simple: Mapper2 reads <key, value> pairs and does nothing but writes them correspondingly. After sorting the output from Mapper1 by key, Reducer2 get the page p_{r1} and its related pages list list(v_{r1}) as input. Then Reducer2 uses a min-heap with maximum capacity M to pick at most M pages with highest total similarity to p_{r1} as the related pages of p_{r1}.

This implementation of finding related pages based on the MapReduce model has the following properties:

- If we regard pages as vertices and co-citation links between pages as edges in the web graph, all Mapper1 operations will visit all edges in the graph and require O (V^2) time, where V denotes the number of vertices. In reality, the graph processed is a sparse graph. The co-citation edges are generated between links in a block, so the number of edges in the graphs is O (V) and all Map and Reduce operations in our method can be finished within O (V) time. Since, the

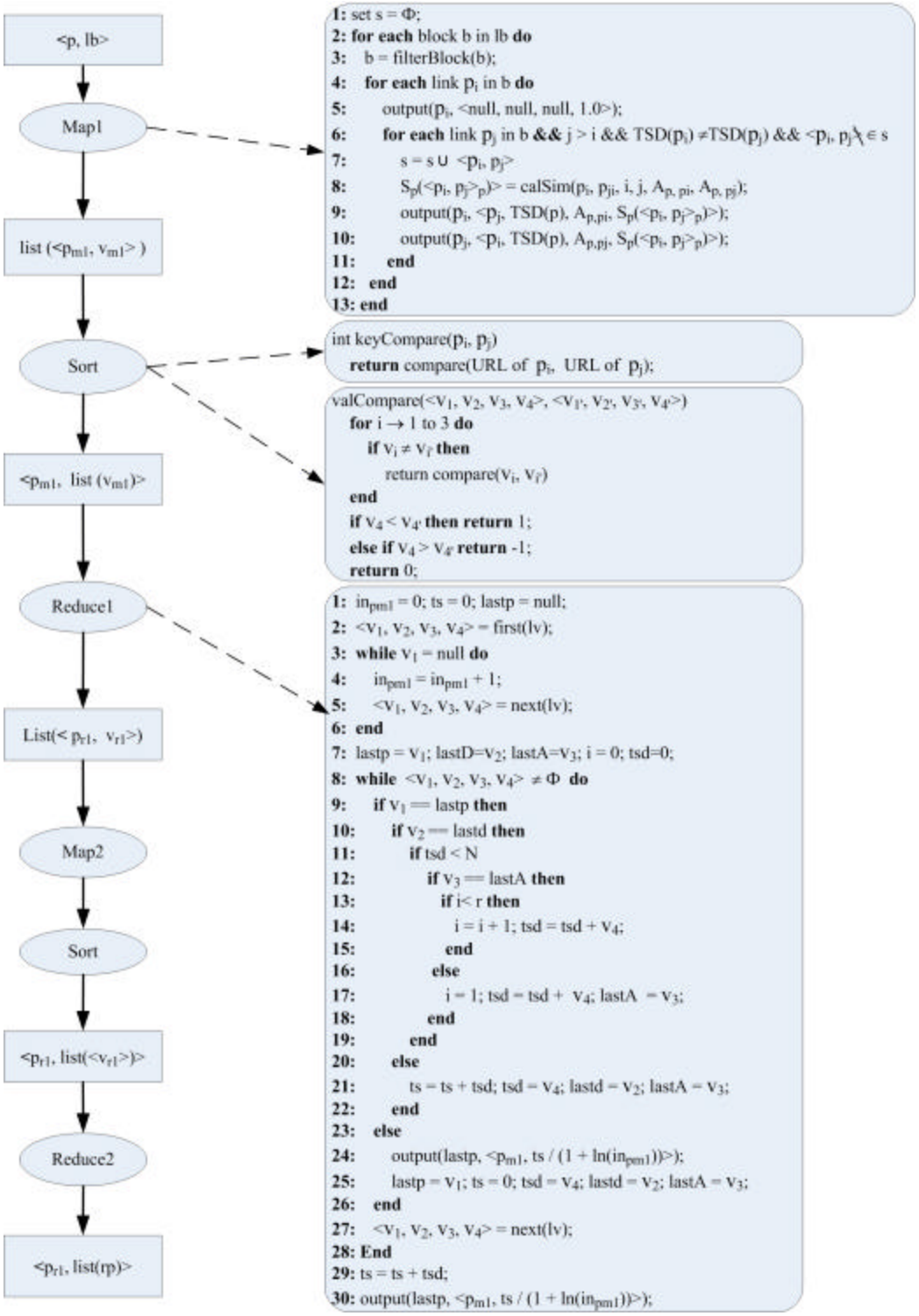

Fig. 5: MapReduce flow of finding related pages

sort operation between Mapper1 and Reducer1 requires sorting all edges, the time complexity is O (E log E)≈ O (V log V). Therefore, the block co-citation algorithm can process a corpus with V pages within O (V log V) time

- After building the related pages data for each page, finding related pages is a simple database lookup. Therefore, our algorithm is very suitable for commercial search engines
- Since, all Reduce operations process the values for a given key in order and have no backtracking steps, the main memory required by Reduce operations is limited to a constant size. Hence, the total memory requirement is no more than the memory needed by the sort process between Map and Reduce operations
- Since, all Map and Reduce operations are mutually independent and the sort process can be executed in parallel for each partition, we can use several machines to achieve parallelization

As Fogaras proposed (Fogaras and Racz, 2007), a related page algorithm is scalable if it has the following three properties: the index database must be generated within the time of a sorting operation, up to a constant factor and the results for a query can be retrieved within a constant access time; the algorithm can be executed within a constant available main memory, no matter how much input data need to be computed and both pre-computation and query operations can be implemented in parallel by using several servers. Based on the discussion above, the block co-citation algorithm meets the scalability requirements.

EXPERIMENTAL EVALUATION

Algorithm parameters: The values of the algorithm parameters mentioned earlier were determined from statistical results or experience:

- **Block size threshold S:** According to the statistical information in Fig. 4, there are very few blocks with size larger than 80. We sampled 50 such large blocks; 42 of them were generated from spam pages. Hence, our choice for S was 80
- **Near neighbor threshold λ:** Several Chinese hub pages were sampled and segmented. We found that the size of blocks used to recommend web sites about the same topic was usually less than 9, so we set λ to 8
- **Anchor repetition frequency r:** Figure 6 shows the distribution of anchor repetition frequency for links in the same top sub-domain. For a given top sub-domain name d, almost 75% of links appeared only once with the same anchor. Less than 5% of the links appeared more than 9 times with the same anchor; we suspected these to be template links. We chose r = 9 for the experiment

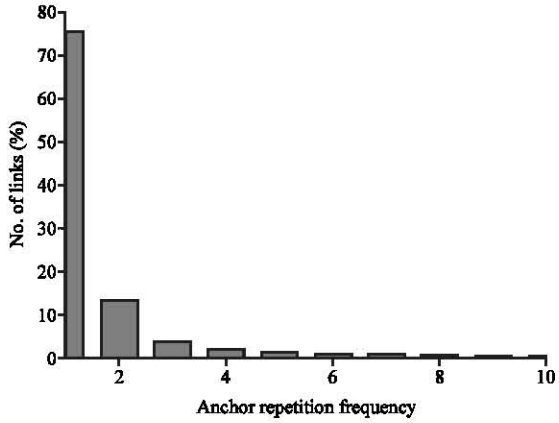

Fig. 6: Distribution of anchor repetition frequency for links in the same top sub-domain

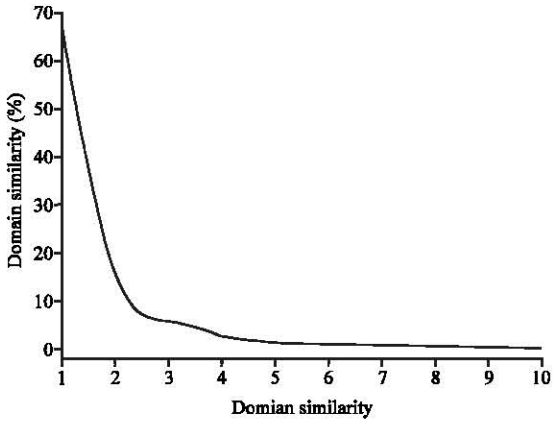

Fig. 7: Distribution of domain similarity values

- **Domain similarity threshold N:** According to the statistics of domain similarity shown in Fig. 7, less than 1% of domain similarities ($Ds_d(p_1 \neg p_2)$) are larger than 10, so we chose N = 10
- **Number of related pages M:** Considering the size of the experimental pages corpus, only 15 related pages were generated and evaluated for each query page if possible, i.e., M = 15
- **Total similarity threshold T:** Many non-famous pages may not have enough neighbor links to generate 15 valid related pages. In order to reduce the number of non-related results, for arbitrary page p_1, the related pages p_i are discarded when $TS(p \neg p)_i$ is lower than 4.0, i.e., T = 4.0

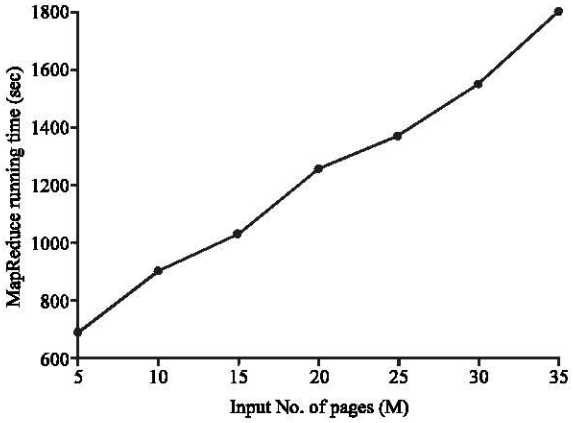

Fig. 8: Running time performance for input page counts from 5 to 35 M

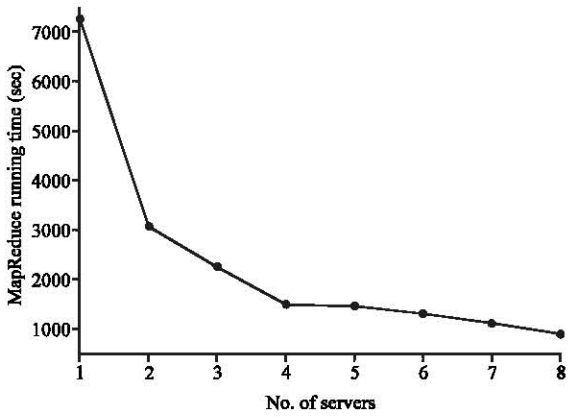

Fig. 9: Running time performance with different numbers of servers

Experimental implementation and running time performance: To implement the algorithm, we selected Hadoop (http://lucene.apache.org/hadoop/) as the MapReduce platform. As the pages corpus we used CWT200G (http://www.cwirf.org), which was sampled from tianwang (http://www.tianwang.com/). It is one of the most famous search engines in China.

The experiment was conducted on 8 HP servers each with 4 GB memory and two dual 1.8 GHz AMD processors. All 37482913 pages in CWT200G were processed and a related pages database of approximately 270 MB was generated. The database was divided into 64 partitions according to the hash code of the page URLs and stored on disk. For a given query page, we need only choose the corresponding partition and use binary search on this partition to get its related pages; the average time for finding the related pages for a given page is 165 ms.

Dean and Ghemawat (2004) proposed that both the number of Map pieces and Reduce pieces should be much larger than the number of machines in order to improve dynamic load balancing and speed up the sorting operations between map phases and reduce phases; hence, we set these two numbers to 64. The time required to find related pages on 8 servers was measured by using input sets with different sizes from 5 to 35 M pages. Figure 8 shows that the total running time was linear with the number of pages. The reason is that the I/O time cost is significant in our parallel implementation.

In addition, the time required with different numbers of servers, measured on using a fixed input set of 10 M pages is depicted in Fig. 9. The running time was significantly reduced when the number of severs increased from 1 to 4. This result can be explained as follows: some partitions may have pages with more valid links than other partitions, since, the partition function fails to uniformly divide pages. Consequently, the process time required in map or reduce operations may be very long with only one server. By using two or more servers, several partitions need less time to process in parallel and at the same time other servers can work on the partition which needs a long time. When the number of servers was larger than 4, the time reduction became much less. This was because the I/O load between servers reached a balance.

Evaluation of performance results: We also implemented traditional co-citation algorithms using Hadoop. The parameter BF was set to 8, the same value as (Dean and Henzinger, 1999). For an arbitrary page p, since all parent pages of p in CWT200 g must be checked to generate co-citation links, the parameter B in traditional co-citation algorithm is equal to the number of parent pages of p in CWT200 g.

In order to evaluate the precision of the proposed block co-citation algorithm and traditional co-citation algorithm, a user feedback study was performed. We carried out a blind evaluation of the results of two algorithms on 28 different pages. These pages belonged to 7 topics and each topic had 4 pages, as listed in Table 2. All pages selected are home pages of famous web sites in China to ensure many neighbor links in CWT200 g to generate enough related pages results for evaluation.

For each test page, we combined and shuffled the top 15 related page results of both algorithms before presenting them to our human evaluators. Each of the 7 evaluators, who are students in our lab, was instructed to mark all the results based on the following criteria:

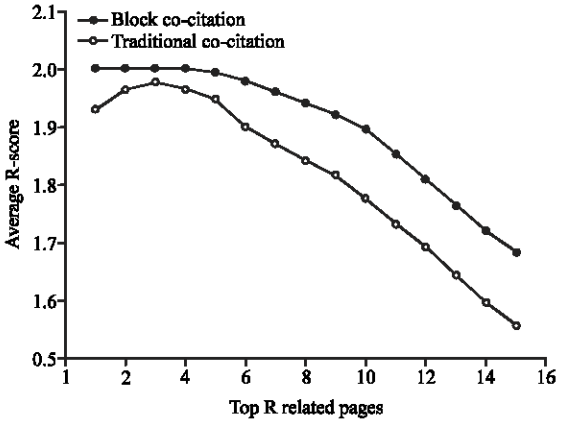

Fig. 10: Comparison of the average R-scores

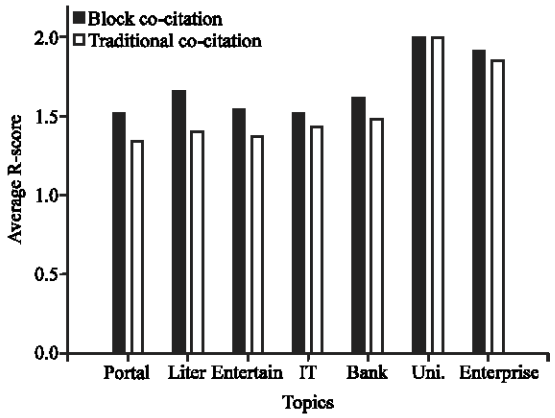

Fig. 11: R-score comparisons by topic

Table 2: Pages used for evaluation

Topic	Selected pages
Portal	www.sina.com.cn;www.sohu.com; www.163.com;cn.yahoo.com
Literature	www.xxsy.net;www.hongxiu.com; www.rongshuxia.com;hjsm.tom.com
Entertainment	www.ourgame.com;www.cga.com.cn; www.the9.com;www.17173.com
IT	www.it168.com;www.enet.com.cn; www.pconline.com.cn;www.yesky.com
Bank	www.ccb.cn;www.icbc.com.cn; www.95599.cn;www.cmbchina.com
University	www.tsinghua.edu.cn;www.pku.edu.cn; www.ruc.edu.cn;www.bupt.edu.cn
Enterprise	www.ibm.com.cn;www.chinamobile.com; www.haier.com;ww.lenovo.com.cn

- 0: Page could not be accessed or is totally unrelated to the test page.
- 1: Page belongs to a similar but not identical topic to the test page.
- 2: Page belongs to the same topic or type as the test page.

The performance in retrieving related pages was investigated through the average R-score. First, the average score of the top R results for every test page was calculated and then the sum of the average scores was divided by the number of test pages.

Figure 10 shows the average R-score comparison of the two algorithms. It shows that the block co-citation algorithm had better average R-scores than the traditional co-citation algorithm over all R values. Although the traditional co-citation algorithm has very similar scores for the top 3 related pages, the difference was significant for R above 3.

Figure 11 shows the average R-scores for different topics. The two algorithms had identical scores for some topics, such as university, denoted as Uni in Fig. 11. Links belong to those topics were more concentrated in the web pages and the advantage of the block co-citation algorithm was not very evident in this case. For many other topics, the R-score was improved by using the block co-citation algorithm.

DISCUSSION

The experimental results revealed that by way of segmenting pages into blocks and limiting the contribution of template links, block co-citation algorithm can avoid some poor results generated by traditional co-citation algorithm. Since treating all links in a page as the same, www.miibeian.gov.cn was also selected as a related page of test URLs by traditional co-citation algorithm as the results of Google shown in Fig. 2. But the poor result did not generated by block co-citation algorithm. Another typical bad case is showed in Fig. 12. 红袖添香 (www.hongxiu.com) and 17173 (www.17173.com) were two links about literature and entertainment, respectively. Traditional co-citation algorithm regards them as neighbors. In our test corpus, there were some pages who had similar HTML structure as shown in Fig. 12, so that 红袖添香 was selected as a related page of 17173. While using block co-citation algorithm, these two links are divided into two different blocks in our experiment, hence the poor result was removed.

For some topic, e.g., University, links belonging to this topic were seldom mixed with links belonging to other topics in our test corpus. There were also many pages which contain the large block of links belonging to this topic. In this case, both block and traditional co-citation algorithm can generate perfect results within this topic, as shown in Fig. 11. Therefore, when links belonging to the same topic were already concentrated into a number of pages, the results of traditional co-citation algorithm approximated those of block co-citation algorithm.

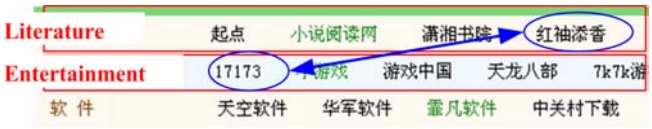

Fig. 12: A typical case which traditional co-citation algorithm can not solve well

As a whole, block co-citation algorithm can be implemented in parallel easily and exhibits good running time performance as show in Fig. 8, 9. It can also generate better results than traditional co-citation algorithm. The algorithm is very suitable for application in commercial search engines.

CONCLUSIONS

In this study, a block co-citation algorithm based on HTML segmentation was introduced to find related pages for a given web page; we demonstrated an implementation of this method using the MapReduce framework to achieve scalability and parallelization. By limiting the vote counts of links with the same anchor text in the same top sub-domain name and the value of domain similarity as shown in Eq. 12, the influence of template links was reduced to a reasonable level. This also prevented the distortion of results by malicious hyperlinks in malicious domains. Experiments conducted on a large corpus suggested that the block co-citation algorithm outperforms traditional co-citation algorithm in the scenarios we tested. Thus the block co-citation algorithm may be a practical choice for commercial search engines to provide a related pages service.

Future extensions of our algorithm will concentrate on the following aspects:

- Currently the links in all pages are treated equally, but we believe that links from a good hub or authority generate more reliable pairs than links in secondary pages. Using hub and authority scores (Kleinberg, 1998) or PageRank (Brin and Page, 1998) to regulate the score of the pairs could keep malicious pages from introducing noise. We will explore this improvement.
- Comparing the results of Google with our algorithm, we find our results for a few test pages may contain mirror pages. For example, given www.sina.com.cn, the experiment returns two Yahoo mirror site: www.yahoo.com.cn and cn.yahoo.com. Too many duplicate results may annoy users and affect the recall of our algorithm. Furthermore, the results returned by our algorithm may not be identical for different mirrors of the same site. We will bring mirror site filter technology into our algorithm to fix these problems. At present, this work is ongoing and produces fairly good performance using the technologies proposed by Broder *et al*. (1997).

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments for this study. They would also like to thank all their colleagues who took their time to evaluate present approach. This study is sponsored by the National Natural Science Foundation of China with grant No. 60432010 and 60872051, the National Basic Research Program of China (973 Program) with grant No. 2007CB307100. The state science and technology support projects with grant No. 2006BAH02A11.

REFERENCES

Asano, Y., H. Imai, M. Toyoda and M. Kitsuregawa, 2004. Finding neighbor communities in the web using an inter-site graph. IEICE Trans. Inform. Syst., E87-D: 2163-2170.

Brin, S. and L. Page, 1998. The anatomy of a large-scale hypertextual web search engine. Proceedings of 7th International World Wide Web Conference, Apr. 1998, Brisbane, Australia, Elsevier Science, pp: 107-117.

Broder, A.Z., S.C. Glassman, M.S. Manasse and G. Zweig, 1997. Syntactic clustering of the web. Comput. Networks Syst., 29: 1157-1166.

Cai, D., S. Yu, J.R. Wen and W.Y. Ma, 2003. Extracting content structure for web pages based on visual representation. Proceedings of the 5th Asia Pacific Web Conference (APWeb2003), Apr. 2003, Xi'an China, Springer, pp: 406-417.

Chakrabarti, S., B. Dom and P. Indyk, 1998a. Enhanced hypertext categorization using hyperlinks. Proceedings of 1998 ACM SIGMOD International Conference on Management of Data, Jun 1998, Seattle, Washington, United States, ACM., pp: 307-318.

Chakrabarti, S., B. D om, P. Raghavan, S. Rajagopalan, D. Gibson and J. Kleinberg, 1998b. Automatic resource compilation by analyzing hyperlink structure and associated text. Proceedings of the 7th International Conference on World Wide Web, Apr. 14-18, Brisbane, Australia, pp: 65-74.

Chirita, P.A., D. Olmedilla and W. Nejdl, 2004. Finding related pages using the link structure of the www.. Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI 2004), Sept. 20-24, IEEE Computer Society, Beijing, China, pp: 632-635.

Dean, J. and J. Ghemawat, 2004. MapReduce simplified data processing on large clusters. Commun. ACM., 51: 107-113.

Dean, J. and M.R. Henzinger, 1999. Finding related pages in the world wide web. Comput. Networks, 31: 1467-1479.

Debnath, S., P. Mitra, N. Pal and C.L. Giles, 2005. Automatic identification of informative sections of Web pages. IEEE Trans. Knowledge Data Eng., 17: 1233-1246.

Fan, W.B., S.F. Wang, H. Jin and J.G. Pan, 2004. Recognition of the topic-oriented Web page relations based on ontology. J. South China Univ. Technol., 32: 37-41.

Fogaras, D. and B. Racz, 2007. Practical algorithms and lower bounds for similarity search in massive graphs. IEEE Trans. Knowledge Data Eng., 19: 585-598.

Haveliwala, T.H., A. Gionis, D. Klein and P. Indyk, 2002. Evaluating strategies for similarity search on the Web. Proceedings of the 11th International Conference on World Wide Web, May 7-11, Honolulu, Hawaii, USA., ACM., pp: 432-442.

Hou, J. and Y. Zhang, 2003. Effectively finding relevant web pages from linkage information. IEEE Trans. Knowledge Data Eng., 15: 940-951.

Huang, S.H.S., C.H. Molina-Rodriguez, J.U. Quevedo-Torrero and M.F. Fonseca-Lozada, 2004. Exploring similarity among web pages using the hyperlink structure. Proceedings of the International Conference on Information Technology: Coding and Computing, Las Vegas, Nevada, Apr. 05–07, IEEE Computer Society, USA., pp: 344-348.

Jain, A.K., 1989. Fundamentals of Digital Image Processing. Prentice Hall, New Jersey, USA., ISBN-10: 0133361659.

Jeh, G. and J. Widom, 2002. SimRank: A measure of structural-context similarity. Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, Edmonton, Alberta, Canada, ACM., pp: 538-543.

Kleinberg, J.M., 1998. Authoritative sources in a hyperlinked environment. Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, Jan. 25-27, San Francisco, California, United States, Society for Industrial and Applied Mathematics, pp: 668-677.

Lee, S.H., S.J. Kim and S.H. Hong, 2005. On URL normalization. Proceedings of the International Conference on Computational Science and its Applications (ICCSA 2005), May 9-12, Singapore, Springer Berlin, pp: 1076-1085.

Liben-Nowell, D. and J. Kleinberg, 2003. The link prediction problem for social networks. Proceedings of the 12th International Conference on Information and Knowledge Management, Nov. 03-28, New Orleans LA USA., ACM., pp: 556-559.

Lin, Z., I. King and M.R. Lyu, 2006. PageSim: A novel link-based measure of web page similarity. Proceedings of the 15th International Conference on World Wide Web, Dec. 18-22, Hong Kong, IEEE Computer Society, pp: 1019-1020.

Loia, V., S. Senatore and M.I. Sessa, 2002. Discovering related web pages through fuzzy-context reasoning. Proceedings of the 2002 IEEE International Conference on Plasma Science, May 12-17,: Institute of Electrical and Electronics Engineer, Honolulu, HI, USA.,-pp: 150.

Ollivier, Y. and P. Senellart, 2007. Finding related pages using green measures: An illustration with wikipedia. Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 07), July 22-26, Vancouver, British Columbia, Pierre Senellart, pp: 1427-1433.

Pant, G., P. Srinivasan and F. Menczer, 2004. Crawling the Web. In: Web Dynamics: Adapting to Change in Content, Size, Topology and Use, Levene, M. and A. Poulovassilis (Eds.). Springer-Verlag, Berlin, pp: 153-178.

Philippe, L.H., 2002. The W3C document object model (DOM). http://www.w3.org/2002/07/26-dom-article.html.

Sadi, M.S., M.M.H. Rahman and S. Horiguchi, 2006. A new algorithm to measure relevance among web pages. Proceedings of the 7th International Conference on Data Mining and Information Engineering, July 11-13, WIT Press, Prague, Czech Republic, pp: 243-251.

Tombros, A. and Z. Ali, 2005. Factors affecting Web page similarity. Proceedings of the 27th European Conference on IR Research (ECIR 2005), Mar. 21-23, Santiago de Compostela, Spain, Springer Berlin, pp: 487-501.

Tsuyoshi, M., 2001. Finding related web pages based on connectivity information from a search engine. Proceedings of the 10th International World Wide Web Conference, May 01-05, Hong Kong, pp: 18-19.

Ubaldo, Q.J. and S.H.S. Huang, 2003. Similarity among web pages based on their link structure. Proceedings of the International Conference on Information and Knowledge Engineering, June 23-26,: CSREA Press, Las Vegas, NV, United States,-pp: 232.

Xin, Y., X. Peifeng and S. Yuanchun, 2006. Semantic HTML page segmentation using type analysis. Proceedings of the 1st International Symposium on Pervasive Computing and Applications, Aug. 03-05, Urumqi, pp: 669-674.

Zhu, X., S. Huang and Y. Yu, 2003. Recognizing the relations between web pages using artificial neural network. Proceedings of the ACM Symposium on Applied Computing, Mar. 09-13, Melbourne, Florida, ACM., pp: 1217-1221.