Expert Systems with Applications 38 (2011) 3944–3958

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Automatic sitemaps generation: Exploring website structures using block extraction and hyperlink analysis

Shian-Hua Lin *, Kuan-Pak Chu, Chun-Ming Chiu

Department of Computer Science and Information Engineering, National Chi Nan University, Taiwan

ARTICLE INFO

Keywords:
Web mining
Sitemap
Hyperlink analysis
Block extraction
Sequence analysis

ABSTRACT

Sitemaps designed by webmasters are not only presenting the main usage flows for users, but also organizing the hierarchical concept of the website. However, websites seldom provide sitemap pages to facilitate users to browse pages easily. Even provided, these sitemaps are not for machine-understanding, although few websites provide sitemaps with the XML format. In this paper, we develop a system, SiteMap Generator (SMG), to automatically generate the hierarchical sitemap for a website. SMG consists of five components. *Sequence Translator* translates a page's HTML source into a long sequence and then *Page Partitioner* splits the page into blocks based on analyzing the sequence complexity. *Block Identifier* categorizes each block into one of three block types: *content*, *structure* or *redundant*. Using the popular sequence searching tool, BLAST, *Block Cluster* calculates similarities between blocks so that blocks with similar functionalities are grouped and considered as candidate blocks for the sitemap. Finally, Hyperlink Analyzer transforms page-to-page links into block-to-block links and applies Kleinberg's HITS algorithm to estimate authority and hub values of each block. Block entropy value derived from features entropies is also used to improve the HITS. Several experiments on three websites: Mozilla, CNN and Yahoo! News, show that SMG is useful to partition a page into blocks (F1 = 86%), identify the block type (F1 = 85%), and generate the sitemap for the website (F1 = 63%).

1. Introduction

With the explosive growth of the Web, huge numbers of websites create and publish contents as static pages with HTML and related formats like CSS, XML and XSL. As static pages rapidly increased, maintaining numerous static pages becomes a tedious and hard work. Considering issues of flexibility and scalability while maintaining pages, web publishing techniques continuously migrate from writing static pages to deploying dynamic application programs (CGI programs), which dynamically create pages to serve requests based on predefined schemas or templates stored in back-end databases or storages, respectively. Most commercial websites, such as portals, search engines, e-commerce stores, and news publishers, apply this dynamic technique to adapt various requests from numerous web users. These websites are called *systematic websites* (Lin & Ho, 2002). Based on predefined schemas or templates, systematic websites efficiently organize numerous pages with dynamic hyperlinks so that users can easily read related pages through few clicks. Some websites also build sitemap pages to summarize the major content with tree-like hierarchical links to pages. Actually, the sitemap usually presents the knowledge structure of a website in forms of hierarchically conceptual categories. Also, the sitemap can be regarded as major usage flows for users. An example of sitemap presented by Mozilla.org[1] is shown in Fig. 1, in which contents are organized into different usage flows: "Products," "Support," "Store," "Developers" and "About".

The sitemap represents an explicit specification of the design concept and knowledge organization of a website and is therefore considered as the website's basic ontology (Gruber, 1993). While performing mining tasks on a website, the sitemap information facilitates web mining processes with the initial background knowledge. For example, wrappers of information extraction (Ashish & Knoblock, 1997; Kushmerick et al., 1997) would be able to roughly identify categories of a website according to the guidance of sitemap, different wrappers templates can therefore be efficiently applied to extract structured metadata from pages of individual categories. Focused crawlers (Chakrabarti, Van Den Berg, & Dom, 1999b; Diligenti, Coetzee, Lawrence, Giles, & Gori, 2000) need the pre-defined set of topics organized as a tree to selectively and efficiently seek out relevant pages.

Although sitemaps are useful to improve web applications; unfortunately, websites seldom provide sitemaps. Even provided,

* Corresponding author.
E-mail address: shlin@ncnu.edu.tw (S.-H. Lin).

[1] "http://www.mozilla.org/sitemap.html" is a popular website of open source software. Example pages of Mozilla used in this paper are visited at June 6, 2006.

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.09.056

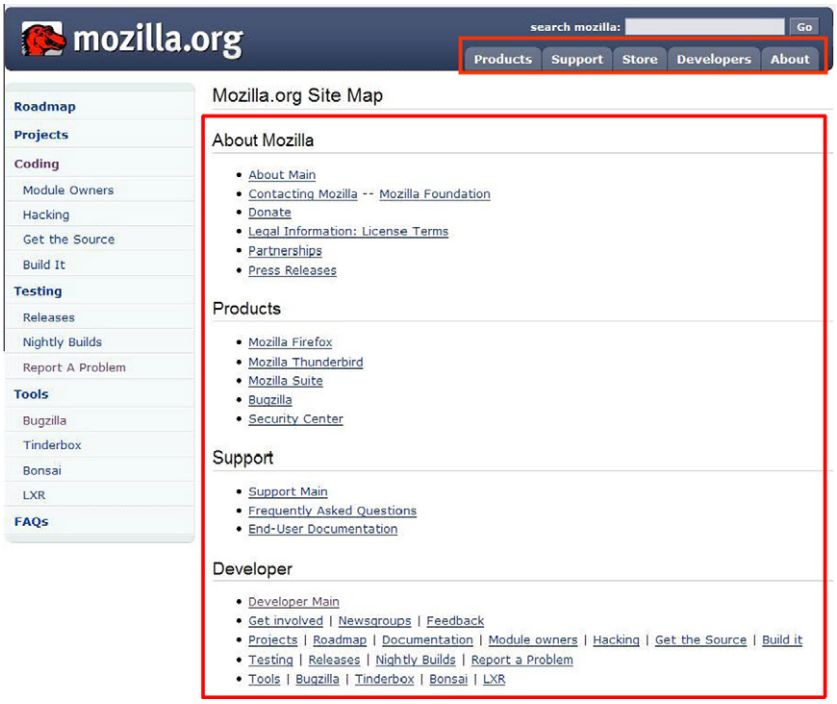

Fig. 1. The sitemap page of "http://www.mozilla.org/".

sitemaps are only for human browsing rather than computer understanding; only few websites provide XML-based sitemap data (in RSS format). There are rich knowledge structures and relationships hidden behind the website's obscure hyperlinks and pages, the sitemap is the clue to extract the hidden knowledge. Therefore, automatically mining sitemap structures from websites is significant to various web applications as illustrated in following examples.

- The sitemap generator would facilitate to mine the basic ontology of the website by distilling its complex hyperlink structures (Kao, Lin, Ho, & Chen, 2002, 2004).
- Automatically generated sitemaps can be applied to provide user-friendly browsing environments for mobile devices through distilling and simplifying website structures that are originally designed for desktop PC users.
- Dynamically generated sitemaps may contribute to design factors of web sites, such as "customized design", "functional service", "information service", "information usefulness", and "ease of use", defined in (Lee & Lee, 2003) for serving requirements of web users.
- Following links appearing in the sitemap, web crawlers concentrate on pages pointed by the sitemap to efficiently gather major contents of the site. Similar to focused crawling (Chakrabarti et al., 1999b; Diligenti et al., 2000), the sitemap would improve the performance of web applications like agents or search engines.
- Sitemaps can also be utilized to integrate directories from domain portal sites. For example, a global news portal can be integrated by gathering news categories appearing in news website maps.

In this paper, several methods are proposed and seamlessly integrated to develop the SiteMap Generator (SMG). The major function of SMG is to distill the complex structure of a website into a concise sitemap as shown in Fig. 2. Considering hyperlinked web pages written in the HTML format, blocks and links specified by HTML tags are intrinsic data to the problem. Therefore, SMG mainly consists of four processes: *block extraction*, *block identification*, *block clustering*, and *hyperlink analysis*. Former three processes partition a page into different types of clustered blocks; the last one analyzes hyperlinks among blocks of pages to evaluate the contribution of a block to the sitemap.

This paper is organized as follows. Section 2 presents studies related to the problem and our proposed methods. Section 3 illustrates concepts and ideas of the SMG system. The implementation of SMG is described in Section 4. Section 5 shows several experiments to evaluate the performance of SMG. Finally, we conclude contributions and point out several issues as future works in Section 6.

2. Related work

Although extracting the sitemap is an important preprocess for many web applications, few studies are proposed to solve the problem. Thinking the design of pages, blocks and links specified by HTML tags are intrinsic data since they are corresponding to the page layouts and relations (hyperlinks), respectively. Accordingly, extracting blocks and analyzing link relations are trivial approaches while analyzing the website structure to automatically extract the sitemap. The research problem is therefore divided into following sub problems with corresponding to related studies.

- *Block extraction*: An HTML page is partitioned into fine-grained blocks so that following analytical processes focus on significant content and links located in few blocks.
- *Block identification*: The process recognizes content and structure blocks by removing irrelevant and redundant blocks.
- *Hyperlink analysis*: Obviously, link analysis methods based on the Hypertext-Induced Topic Search (HITS) (Kleinberg, 1998) algorithm are possible solutions to extract link structures for generating the sitemap.

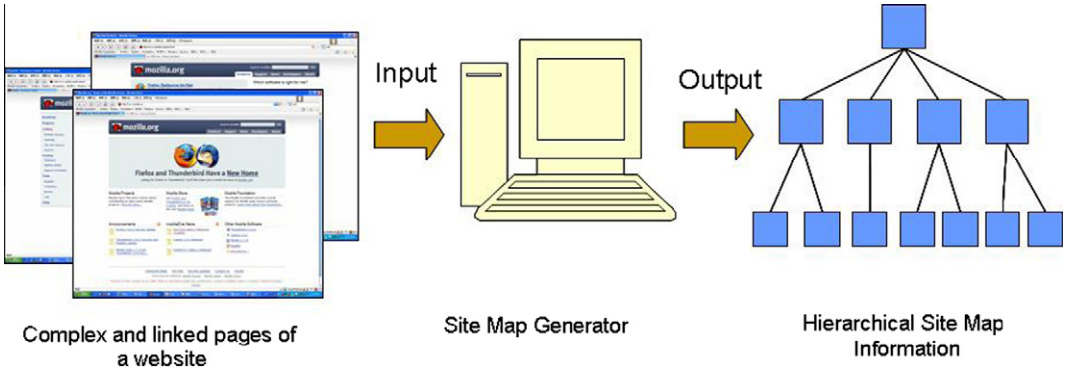

Fig. 2. The major function of SMG.

2.1. Block extraction

Block extraction[2] is also known as "page segmentation" mentioned in (Yu, Cai, Wen, & Ma, 2003). Based on the Document Object Model (DOM)[3], an HTML page is represented as a tree in which nodes and edges are corresponding to tags and embedded relations between tags. In this paper, block extraction methods are roughly categorized into *tag-based* and *vision-based* segmentation methods. Both kinds of methods extract page blocks from perspectives of webmasters and users, respectively.

2.1.1. Tag-based method

According to specific HTML tags, tag-based methods partition a page into blocks based on common heuristics of designing websites and pages. For example, tags like <table>, <ul> and are widely used to present the skeleton of page layout. Contents embedded in these tags are usually regarded as blocks of the page; the tag-based method is therefore regarded as a method from the perspective of webmasters. The tag-based approach proposed in (Lin & Ho, 2002), called InfoDiscover, attempts to partition a page into content blocks based on the <table> tag since <table> is widely used to design the page layout based on statistics on millions pages. Experiments on several news websites show that InfoDiscoverer is able to precisely extract informative content blocks. Debnath, Mitra, Pal, and Giles (2005) also propose a tag-based approach that partitions a page not only according to <table> tag, but also other tags based on a pre-defined tag-list. By parsing a page into the DOM tree, Chakrabarti, Joshi, and Tawde (2001) addresses the fine-grained topic distillation and disaggregates hubs into regions (blocks) by analyzing link structure as well as intra-page text distribution.

Tag-based methods are easy to implement by reusing mature search engine modules like crawlers and HTML parsers. However, tags are sometimes ambiguous to present page blocks since webmasters seldom obey suggestions of the HTML validator specified in the W3C QA[4]. Moreover, if blocks are not presented by these specified tags, the performance of tag-based methods may be unacceptable. Therefore, tag-based approaches still have rooms to be improved.

2.1.2. Vision-based method

VIsion-based Page Segmentation (VIPS) algorithm (Cai, He, Wen, & Ma, 2004; Yu et al., 2003) applies visual features to partition a page into blocks. VIPS aims to extract the semantic structure of a page based on visual characteristics, i.e., from the user's perspective. Given a page, VIPS first extracts all candidate blocks from the DOM tree. By identifying horizontal and vertical spaces, it delimits blocks as a human being would visually identify semantic blocks in a page. The extracted semantic structure is a tree structure; each node in the tree corresponds to a block. Each node will be assigned a value denoted by Degree of Coherence (DoC) to indicate how coherent of the content in the block based on visual perception.

Compared with tag-based methods, VIPS segmentation scheme utilizes useful visual clues to obtain a better partition of a page at the semantic level. From the user perspective, vision-based method seems more intuitive and robust than tag-based methods. However, different users may read pages with different views; visualized blocks are therefore not uniform. Also, the DoC threshold is hard to be determined to find the optimal extraction of blocks. In this paper, both kinds of methods are considered to improve the block extraction performance.

2.2. Block identification

Block identification methods attempt to categorize page blocks into pre-defined types, such as *redundant*, *informative* or *structured* blocks so that various web applications are able to focus on relevant block types (Lin & Ho, 2002). For example, search engines focus on informative (content-rich) blocks to create indices for significant content; link analyzers merely consider structured blocks without processing links appearing in redundant blocks. Block identification methods can be roughly classified as follows: content-based and classification-based methods.

2.2.1. Content-based method

Content-based methods analyze the textual content of a block to identify the block property. InfoDiscoverer (Lin & Ho, 2002) extracts blocks with terms extracted as features. The feature entropy is estimated based on the term distribution in all pages. The block entropy is derived from entropy values of terms within the block, then the entropy threshold is automatically determined to categorize blocks into informative or redundant with corresponding to lower or higher entropy values. In this way, redundant blocks, such as menus, navigation bars, advertisements, and copy right statements, are removed from pages. Hence, the performance of web applications like link analyzers and search engines is improved. Based on the observation "In a given website, noisy blocks usually share some common contents and presentation styles, while the main content blocks of the pages are often diverse in their actual contents and/or presentation styles.", Yi and Liu (2003) propose methods to analyze contents and presentation styles of pages for eliminating a page's non-content blocks, referred to as "noisy"

[2] "Page segmentation" is widely used in research fields like computer vision, image processing or pattern recognition so that "block extraction" is used in this paper.
[3] W3C DOM, "Document Object Model (DOM)," http://www.w3.org/DOM/.
[4] W3C Quality Assurance Tools: http://www.w3.org/QA/Tools/.

blocks. The tree structure, called Style Tree, is designed to capture common presentation styles and actual contents of pages within the website.

2.2.2. Classification-based method

Classification-based methods treat the block identification problem as classifications of blocks. Song et al. (2004) propose a learning algorithm to measure the importance of different blocks. The VIPS technique (Yu et al., 2003) is employed to partition a page into hierarchical blocks. Then, spatial features (such as position and size) and content features (such as the number of images and links) are extracted to construct a feature vector for each block. According to block feature vectors of pages in the training set, learning algorithms, such as Support Vector Machines (SVM) (Hsu & Lin, 2002) and Neural Networks are used to train a model so that the system is able to assign importance values to blocks of new pages.

2.3. Hyperlink analysis

The web hyperlink environment can be regarded as a directed graph, in which nodes and edges are pages and links, respectively. Based on analyzing links in the directed graph, the importance of each node can be estimated. Kleinberg's HITS (Kleinberg, 1998) (Section 2.3.1) and Google's PageRank (Brin & Page, 1998; Page, Brin, Motwani, & Winograd, 1998) (Section 2.3.2) are two representative link analysis algorithms. Many link analysis algorithms are derived from both algorithms. The basic idea of link analysis algorithms is that if a link points to page v from page u, then some relation may exist between both pages. However, a link from page u to page v usually denotes the semantic relation between certain content parts of both pages rather than the whole page content. Therefore, many studies revise HITS and PageRank with different link-weighting schemes (Section 2.3.3) and link relations between fine grain blocks rather than the whole page (Section 2.3.4).

2.3.1. Hyperlink induced topics search (HITS)

The HITS algorithm analyzes hyperlinks on the Web and draws two conclusions. First, different pages are not of equal importance, authoritative and hub pages are more valuable. Second, authoritative and hub pages generally reinforce each other; better authority pages come from incoming links from good hubs and better hub pages come from outgoing links to good authorities. To offer better search result for queries, based on "seed pages" of the initial search result, HITS includes the "pointing-to" or "pointed-by" pages of the "seed pages". Based on the expanded page set, authority and hub values of pages are calculated to re-rank the search result based on authority or hub values of the expanded pages.

2.3.2. Google's PageRank

PageRank is a random walk model to simulate a user who at some page, and decides which page to visit on the next step as follows. With probability $1 - q$, the user randomly picks one of the hyperlinks on the current page and jumps to the page it links to; with probability q, the user "resets" by jumping to a page picked uniformly and at random from the collection. Based on the idea, Google gets much better ranking results than those returned by other text-based search engines.

2.3.3. Entropy-based link analysis

Based on Kleinberg's HITS algorithm, the "Link Analysis for Mining web Informative Structures (LAMIS)" system attempts to explore the informative structure of a website (Kao et al. 2002, 2004). Given a website, all pages of the website are parsed to extract terms from textual information. According to the term distribution, the entropy measure is applied to estimate the significance of anchor texts as links weights while applying HITS to analyze the website structure. Experiments show that LAMIS obtains more important links than the original HITS algorithm.

Several kinds of link-weighting schemes are also proposed to enhance the significance of links. In the Clever system (Chakrabarti et al., 1999a), weights tuned empirically are added to distinguish same-site links and others. In (Bharat & Henzinger, 1998), the similarity between the pointing-to and the pointed-by pages is taken as the link weight. Another study that uses the similarity between the surrounding text of a link and the pointed-by page to determine the link weight is proposed in (Chakrabarti et al., 1998). Considering the distribution of terms in pages, the TFIDF-weighted model is applied to measure the similarity.

2.3.4. Block-level link analysis

Most of link analysis algorithms treat a page as a single node in the Web graph. However, in most cases, a page contains rich semantics and hence the page might not be considered as an atomic node. Cai, He, Wen and Ma (2004) propose a block-level link analysis algorithm in which a page is partitioned into blocks by using their VIPS algorithm (Yu et al., 2003). By extracting page-to-block and block-to-page relationships according to the link structure and page layout analysis, block-to-block relationships can be obtained to form a fine-grained graph. Therefore, blocks can be considered as the atomic node while applying HITS- or PageRank-based link analysis algorithms. Based on block-to-block relationships, they propose two algorithms: Block Level PageRank (BLPR) and Block Level HITS (BLHITS). Experiments show that both BLPR and BLHITS are better than original PageRank and HITS algorithms.

Several ideas of above methods are employed in this study to solve the problem of sitemap extraction. The performance of above methods is not compared since these methods did not use the same data set or measure in accordance with the common baseline method. Also, the information on the Web is regarded as an infinite set; different methods are therefore hard to be evaluated based on a tiny data set. Consequently, these methods are hard to be compared through standard evaluation criteria.

3. Concepts and ideas

The sitemap consists of links to important pages of the website to indicate the usage flows. To facilitate the access to pages, webmasters usually put these important links in fixed blocks and disseminate these blocks to most pages of the website. Intuitively, these links are located to be rapidly focused and clicked. Based on the DOM tree of an HTML page, a block is regarded as a subtree that may be recursively partitioned into smaller subtrees as fine-grained blocks. The problem of extracting sitemaps is therefore reduced to "starting from the website homepage and finding important blocks and links that probably form pages and links of the sitemap." Following sections define concepts used in this paper and illustrate details of ideas to solve the problem.

3.1. Concepts and definitions

Given the homepage URL, the web crawler widely used in search engines can efficiently fetches all pages within the website. The sitemap generator attempts to analyze page layout and link structures to generate the tree-like sitemap efficiently and effectively. According to observations on designing pages, the page layout is conceptually organized as blocks to deploy various types of contents and links. A page is therefore partitioned into blocks while analyzing page structures. In this paper, blocks are assigned with attributes to denote different characteristics and meanings:

Content Block (CB) and Structure Block (SB). Details of concepts and definitions are described as followings:

- *Website*: A website, indicated by the homepage URL, denotes a set of pages that form a complicated directed graph with "page" nodes and "link" edges. In general, a website can be regarded as a set of URLs sharing the same domain name. To simplify the problem in this paper, the website indicates a host that contains pages whose URLs share the same hostname.
- *SiteMap (SM)*: A sitemap, manually built by webmasters, is a tree-like graph that consists of major content pages of the website. SM is regarded as a concise view of the website. By taking a glimpse at the sitemap, users can efficiently realize contents and knowledge organization of the website. However, websites seldom provide sitemaps for users.
- *SiteMap Generator (SMG)*: SMG is an intelligent web system that analyzes pages of a website to builds the sitemap automatically.
- *Block*: A page's layout is conceptually organized as visualized (form webmaster's or reader's perspectives) blocks that are usually embedded in HTML tags like <table>, , etc. A block belongs one of the following block types as shown in Fig. 3.
- *Content Block (CB)*: CB is the block that contains the major content (concept) of the page. For example, blocks containing texts of news articles are regarded as CB of the news page. If the content of the CB is not relevant to the page's topic, the CB is regarded as a redundant block (RB).
- *Link Block (LB)*: LB is a block in which the major information is dominated by links. LB can be regarded as a candidate of Structure Block (SB). However, not all LBs are finally identified as SBs.
- *Structure Block (SB)*: SB is defined as the block that probably contributes links to construct the sitemap structure. For example, websites usually put links of content categories in navigation or menu bars that are regarded as SBs. Most pages of the website contain SBs so that users can easily browse important categories or relevant pages.
- *Redundant Block (RB)*: RB is the block that contains irrelevant contents and links or appears in less important visualization areas of a page. In commercial websites, advertisements are examples of the former case; page footers and copyright announcements are examples of the later.

3.2. Ideas and methods

In order to automatically collect pages of a website, the web crawler is employed to fetch all pages of the site. Then theses pages are parsed into DOM trees and stored in the pre-defined relational database. Based on a page's DOM tree, partitioning the page into blocks is equivalent to dividing the DOM tree into several subtrees. By translating a page's tags and texts into a symbolic sequence, *Page Partitioner* is proposed to partition a page into several blocks by analyzing the sequence complexity and performing generalization/specialization operations on the DOM tree. Obviously, sitemap links often appear in SB. Therefore, *Block Identifier* is then applied to determine each block type, especially for SB, according to tag, text and link features of the block.

Usually, pages of a systematic website are generated by CGI programs for easy management. Many pages usually share the same template and present similar styles. For example, to facilitate users navigate the website easily, webmasters usually put catalogs in the navigation menu (identified as SB) that are located in most pages for easily accessing. The occurrence frequency of "similar" SB sequences (blocks) is therefore high. The frequency is estimated according to "similarity" rather than "equality" since block sequences may not be equal while presenting the same block (like navigation menu) with different clicked links. *Block Cluster* is therefore designed to group a set of very similar blocks into the same cluster so that the "similar-block" frequency can be correctly estimated according to similarities between block sequences. Intuitively, SM should be generated from links in high-frequency blocks. SM represents the simplified hierarchical structure of a website; the depth of the sitemap is seldom large (seldom greater than three levels without considering the root). Consequently, SM is generated by top-down approach; i.e., finding important links from the homepage (root), then following linked pages (SM pages

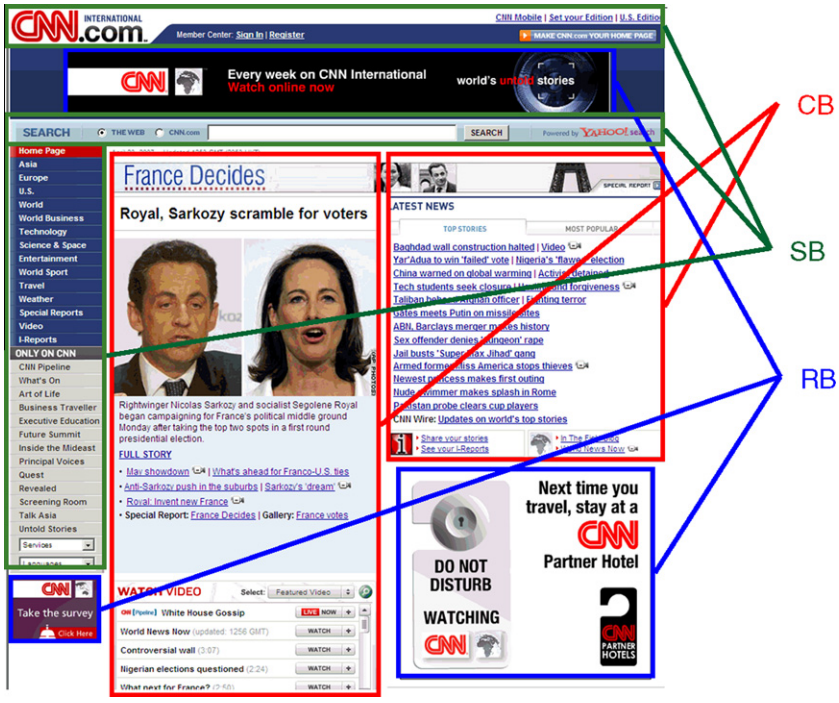

Fig. 3. Categories of blocks (CB, SB and RB) and manually identified blocks of the CNN homepage (visited at April 23, 2007).

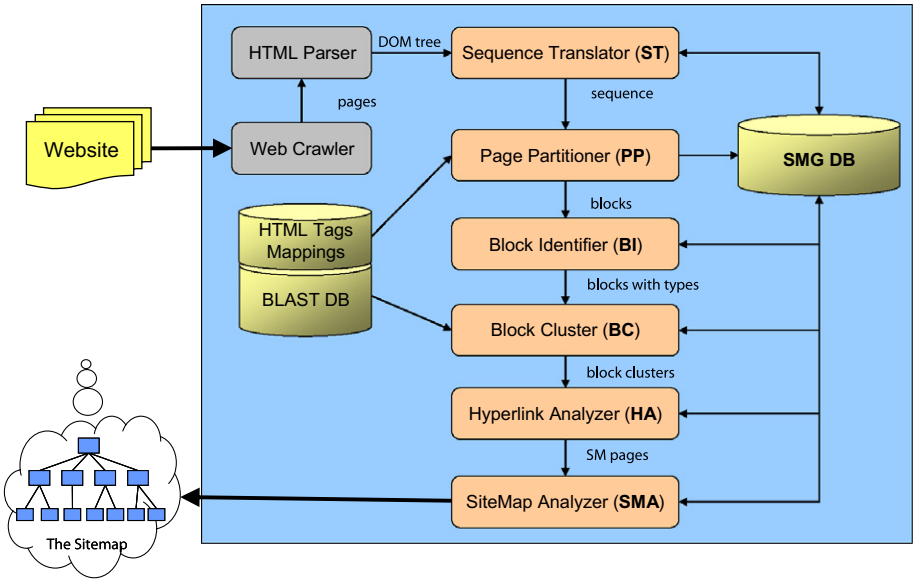

Fig. 4. The system architecture of SMG.

candidates) in the next level, and vice versa. To determine the link importance, HITS is usually applied to calculated hub and authority of a page. By partitioning a page into blocks, the original page-to-page HITS algorithm is revised to block-to-block algorithm. Based on the block-level HITS algorithm, *SiteMap Analyzer* estimates hub and authority values of blocks and combines with block frequencies to determine the sitemap.

According to these ideas and concepts mentioned above, major procedures of SMG are summarized as followings. The detail of each procedure is described in next sections.

- *Web Crawler* fetches HTML pages from a website. In the SMG system, regular expressions for URLs are employed to define the crawling range of a website.
- *HTML Parser* parses an HTML page into the DOM tree stored in the database.
- *Sequence Translator (ST)* translates a subtree into a linear sequence (like a protein sequence) so that the sequence complexity is analyzed as the criterion of partition.
- *Page Partitioner (PP)* divides a page into blocks by running generalization/specialization operations on the tree recursively.
- *Block Identifier (BI)* determines the type of each block. In this paper, SMG concentrates on SB that effectively contributes important links to SM.
- *Block Cluster (BC)* groups a set of similar blocks into a cluster to estimate the block occurrence frequency as an importance measure for generating SM.
- *Hyperlink Analyzer (HA)* runs block-level HITS algorithm according to results of above processes.
- *SiteMap Analyzer (SMA)* focuses on SB to construct the sitemap based on results of BC and HA.

4. Methods and the implementation

The architecture of SiteMap Generator (SMG) system consists of several components as shown in Fig. 4. In order to gather pages of a website, Web Crawler fetches all pages transitively linked by the homepage and builds the link graph for later analyzes. Without losing generality, we simplify the crawling complexity by assuming that "the whole website pages within the same host or domain, like {www.ncbi.nlm.nih.gov} or {*.yahoo.com}"; although some websites deploy content pages in various hosts or domains. Then, HTML Parser is employed to parse HTML pages into DOM tree structures that consist of hierarchical nodes, relations and content texts stored in the database. Non-HTML files, such as images, Microsoft Office or PDF documents, are not considered by the HTML Parser since these documents can be linked as leaves but never contribute to the sitemap structure (internal nodes). Web Crawler and HTML Parser are well-known techniques so that details are omitted in this paper since coding both components is a tedious work. Remaining components are described in following subsections.

4.1. Sequence Translator (ST)

The basic idea to partition a page into blocks is analyzing the DOM tree structure, in which the tree is divided into several subtrees recursively. Each subtree is considered as the candidate of a block. Considering a block containing links or texts embedded in HTML tags, SB contains more links and CB includes more texts. By translating a block (subtree) into a sequence, different types of blocks should have different regularities so that the problem of page partition is transformed into analyzing the sequence complexity of a block. Consequently, we translate an HTML DOM tree into a long sequence according to pre-defined HTML-to-Symbol (H2S) table that maps HTML tags and texts into character symbols. For example, a protein sequence is composed of 20 amino acids and three stop codon symbols (B, X, and Z) for ambiguous amino acids (Agris, 2004; Sethi, O'donoghue & Luthey-Schulten, 2005). However, there are over one hundred tags specified in the W3C HTML[5]; several tags with similar presentation properties should be mapped into the same common symbol. According to heuristics of webmasters on designing pages, some tags seldom used are ignored in the H2S table. Also, some tags having similar effectiveness on page presentation are grouped into the same symbol. For example, tags used to adjust the styles of text, such as <b> and <i>, are grouped into the same type since these tags have no influence on the page layout. As for significant tags widely used to present the page layout, such as , <table>, and <hr>, the mapping is one-to-one as shown in Table 1. Remaining tags not specified in the table are ignored. Obviously, the H2S table is customizable to adapt for web applications

[5] W3C HTML, "HyperText Markup Language (HTML)," http://www.w3.org/MarkUp/.

Table 1
H2S table: Mapping HTML tags to symbols.

Symbol	Tags or Text	Descriptions
E	<table>	Page layout
C	<caption>	Annotations of table
K	<tr>	Page layout
N	<td>, <th>	Page layout
D		Page layout
S		Page layout
A	<a>	Link information
Z	<a ...>	<a> Contains attributes, such as "name" or "target", within the tag
H	<h1>–<h6>	Headings
L		List including and <dl>
I	, <dt>	List items
V	<dd>	Indents on list items
T	text	Indicates the text within the nearest tag
Q	<font>	Including <b>, , , <i>, <u>, etc.
R	<hr>	Visualized horizontal line
Y	<pre>	Considered as a separated block
P	<p>	Textual segmentations
W	 	Textual segmentations
F	<form>	Input information
M	<img>	Without "alt" attribute
G	<img ...>	With "alt" attribute
B	<a><img>	Image as the anchor text
X	<object>	Special objects including <applet>, <iframe>, <map>, etc.

and websites. Optimizing the H2S table for applications is another research topic and is out of the scope in this paper.

Based on the H2S table mappings, any DOM trees or subtrees are translated into long sequences. Since rendering HTML pages is the pre-order traversal in the DOM tree, Sequence Translator (ST) performs the depth-first search (DFS) traversal from the root to leaves. Tags and texts are corresponding to internal nodes and leaves, respectively. Visiting and backtracking to the internal node in the DFS traversal is with respect to beginning and ending tags, so that translating the HTML source into a sequence can be completed by one-pass scan with stack structures. The sequence is recursively generated by individually translating each subtree. An example of translating a subtree into a sequence is illustrated in Fig. 5. Consequently, designing heuristics of webmasters are integrated into the ST to facilitate the following block extraction process (Page Partitioner).

4.2. Page Partitioner (PP)

The major functionality of Page Partitioner (PP) is determining boundaries of blocks within a page for splitting a page into several blocks. A block is conceptually corresponding to a subtree of the page's DOM tree; i.e., a sequence *segment*. Accordingly, the problem of PP is transformed to partition a sequence into several segments. Considering the biological meanings of protein sequences that usually contains two kinds of segments: motifs and domains. Relationships among protein sequences can be revealed by the common occurrence of particular clusters of typically around 10 to 20 amino acids (symbols), which is known as a motif (Sigrist et al., 2002). A motif is usually a simple combination of a few secondary structures that appears in several different proteins, for example the helix-loop-helix motif structure. A domain is a more complex combination of secondary structures that by definition has a very specific function. A protein may contain only one domain, or may contain several (Setubal & Meidanis, 1997). Thinking an HTML page as a protein sequence, the page sequence also has meanings similar to the protein sequence. For example, tables and lists within a page often contain repetitive row/column structures and list items, respectively; like motifs in a protein. The page content, likes the protein domain, often contains more different styles of texts and images so that more distinct tags (symbols) are used to present the content (sequence segment).

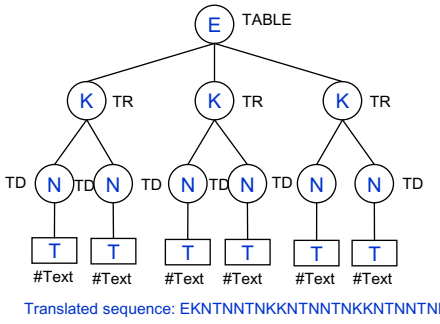

Fig. 5. Translate a DOM tree into an H2S-based sequence by ST.

The idea of PP is splitting the page sequence into low- and high-complex segments that are corresponding to different types of blocks within the page's DOM tree. Wootton and Federhen (1993,1996) developed the SEG algorithm that is widely applied to automatically partition a sequence into low- and high-complexity segments based on the local complexity of the sequence itself. As the example shown in Fig. 6, SEG divides a sequence into low- and high-complexity segments.

From the perspective of user visualization on the page, LB (may be SB) contains many links so that the sequence, with anchor tags and texts, consists of repeated segments; the sequence complexity is therefore low. On the contrary, CB contains rich texts decorated with various tags so that the complexity is high. Using SEG to partition a block sequence into low- and high-complexity segments are conceptually mapping both kinds of segments into LB and CB, respectively. Obviously, some segments may be ambiguous to be mapped into specific blocks. Accordingly, given a sequence S (i.e., a subtree or a block candidate), the *SEG-rate* (R_{SEG}) of the sequence is defined as the ratio of "the length difference between low- and high-complexity segments" to "the length of S (i.e., all segments)" as shown in (1). L and H denote sets of low- and high-complexity segments, respectively. The denominator is trivially equal to the sequence length of the current node (a subtree).

$$R_{SEG}(S) = \frac{\left|\sum_i |L_i| - \sum_j |H_j|\right|}{|S|} \quad (1)$$

Given a threshold for the ratio, says TH_R, if R_{SEG} passes the threshold (i.e., $R_{SEG} \geqslant TH_R$), the sequence is dominated by L-or-H complexity segments and can be recognized as a block. Otherwise, the sequence node is ambiguous and each child node (smaller subtree) will be processed recursively. Based on experiments, the threshold is set to 0.2 to reach the best effectiveness. Consequently, the algorithm of PP is summarized as the following steps.

1. *Initial state*: Given a page's DOM tree, PP starts from the <body> tag as the root node sequence and performs top-down analysis based on the hierarchy of the page's DOM tree.
2. *Recursive analysis*: SEG is applied to partition the sequence into low- and high-complexity segments. If "$R_{SEG} \geqslant TH_R$" is true, the current node is regarded as a block. Otherwise, each subtree of the current node becomes a new sequence and will be analyzed recursively until the sequence can not be partitioned by SEG.
3. *Merging state*: Finally, unsolved "neighboring segments" are merged to form a new block. In other words, identified blocks are applied to separate remaining segments as isolated blocks.

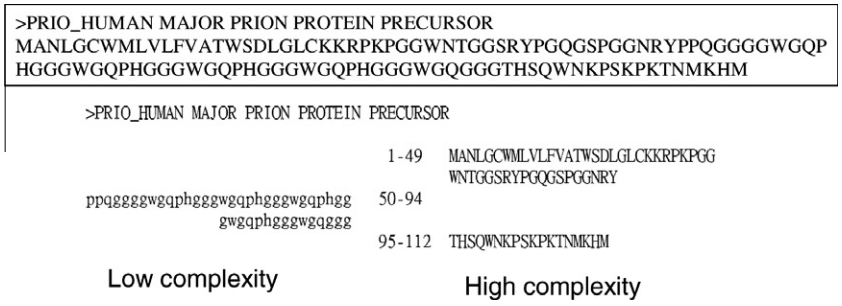

Fig. 6. Low- and high-complexity segments extracted by SEG (Wootton & Federhen, 1993).

An example of PP is shown in Fig. 7. The page is partitioned into nine blocks as shown on the left-hand side with corresponding to nine subtrees of the DOM tree on the right-hand side. Consequently, the block extraction method consists of ST and PP that deal with page segmentation according to perspectives of webmasters (H2S table used in ST) and readers (visualization generalized/specialized based on the DOM tree), respectively.

4.3. Block Identifier (BI)

PP "roughly" divides a page into blocks; then, BI attempts to identify each block type: CB or SB. Although our proposed method, InfoDiscoverer (Lin & Ho, 2002), successfully categorizes page blocks into informative or redundant. In this paper, the goal of BI is extracting SB from those blocks for generating the sitemap. It is not trivial to map informative/redundant blocks into CB/SB since some an informative block may be considered as SB. For example, a block consisting of rich content-link records (e.g., title-anchor with long description) is identified as an informative block, but it is hard to be assigned to CB or SB. Also, a redundant block frequently appearing in most pages, such as navigation bars, may contribute structured links to the sitemap and should be regarded as SB. Consequently, InfoDiscoverer does not work well while "*blocks are mixed with rich contents and links*" and can not elicit SB from redundant blocks. Therefore, BI is proposed to "precisely" classify blocks into: CB or SB. Let's consider following tow extreme cases:

- *Rich-content block*: A block mainly contains textual content with few anchor texts.
- *Rich-anchor block*: A block consists of many links with less textual content.

Obviously, the former block is absolutely assigned to CB and the later is SB or RB (if the block is for advertisements). Therefore, given a block, lengths of textual contents and anchors are clues to identify the block type. Following features are employed to measure a block and analyze the block type.

- *Anchor text length* ($|T_A|$) is the total bytes of all anchors text (the visible anchor text of a link). If a link is denoted by an image or other objects, the "alt= text" attribute is used to measure the length. Otherwise, the average length of textual anchors within the page is used while no information for measuring the length of an anchor.
- *Content text length* ($|T_C|$) is the total bytes of texts within the block, i.e., the block content length.
- *Anchor content rate* (*ACR*) means the ratio between the length of all anchors and the block content length, as shown in (2).

$$ACR = \frac{|T_A|}{|T_C|} \tag{2}$$

Consequently, a strict threshold "$ACR \leqslant 0.1$" is used to recognize the rich-content block as CB. On the other hand, the condition "$ACR \geqslant 0.5$" is applied to identify the rich-anchor block as LB (SB or RB) since anchor texts contribute over a half of texts to the block. Outside above both cases, i.e., "$0.1 < ACR < 0.5$", our proposed InfoDiscoverer is applied to roughly classify informative and redundant blocks as CB and LB, respectively. The remaining task is to revise false-positive CB to SB and extract SB from LB (separate SB and RB). Considering the R_{SEG} measure used in PP, the partitioned block is dominated by high- or low-complexity segments. According to this hint, the "low-complexity dominant" CB (identified by InfoDiscoverer) is updated to SB since the block may contain several links to make the sequence complexity low. As for LB, if the block is "high-complexity dominant" and contains "intra-site links", it is identified as SB since "content-rich LB" may contain structured links. Consequently, BI proposes three measures and integrates PP and InfoDiscoverer to categorize blocks into CB and SB (Note: RBs are removed in BI), as the flow chart shown in Fig. 8.

4.4. Block Cluster (BC)

As pages of systematic websites are usually generated by CGI programs or organized into hierarchical structures, these pages

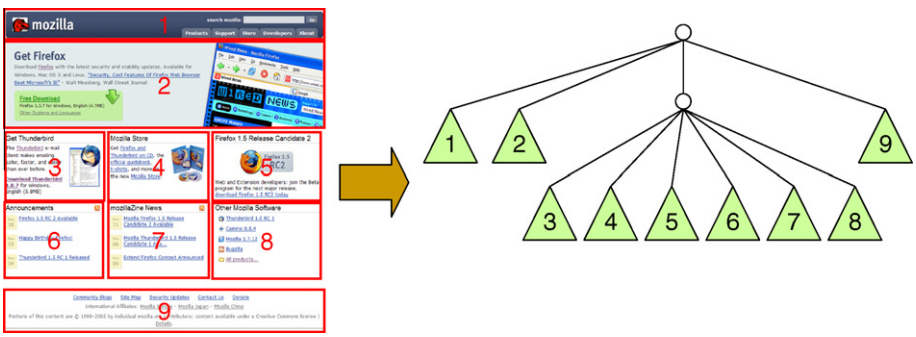

Fig. 7. An example of Page Partitioner.

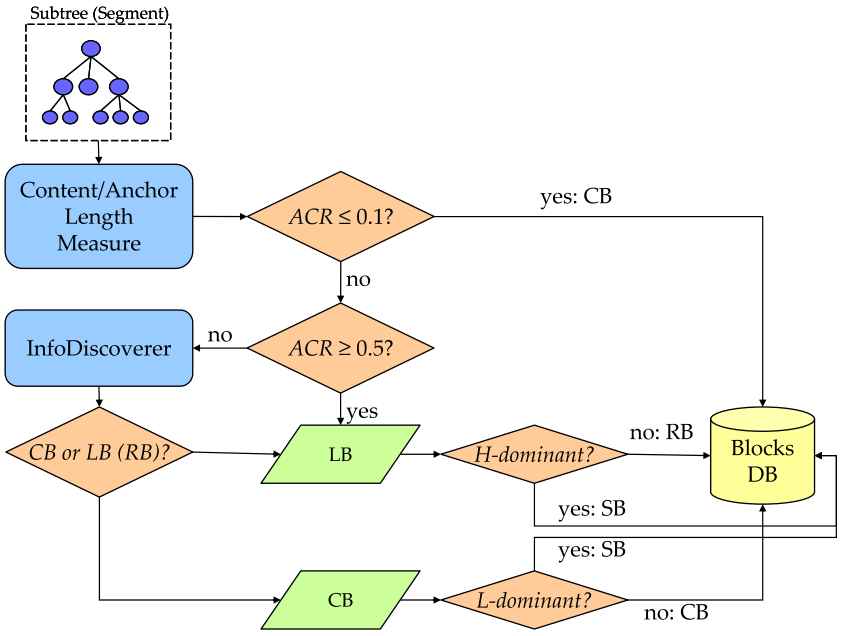

Fig. 8. The Data Flow Diagram (DFD) of BI.

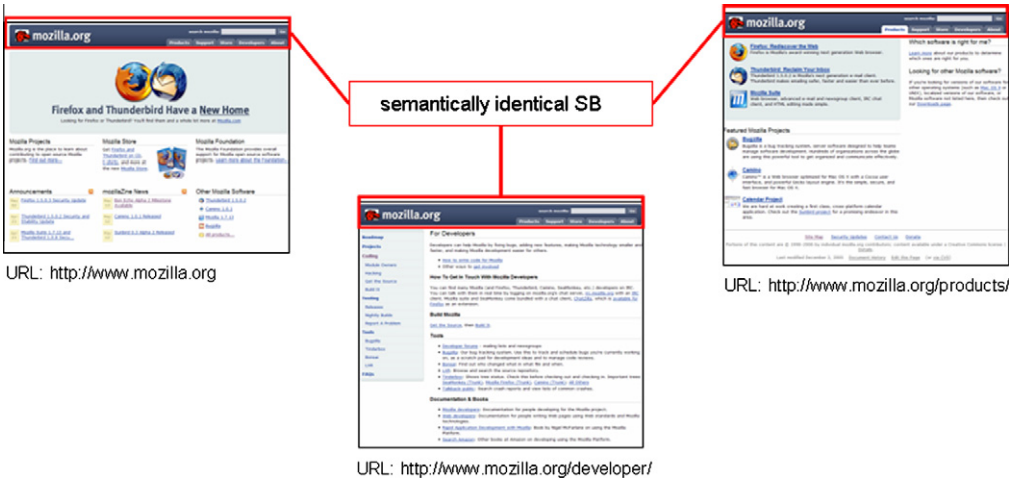

Fig. 9. Semantically identical SBs of the root and second-level pages (“products” and “developer”) in the Mozilla sitemap.

usually share same or similar templates to present common layout styles. For example, to help users navigate the website easily, webmasters usually put catalogs in the navigation menu. The navigation menu recognized as SB by the BI is obviously an important block to the sitemap and should appear in most pages. To hierarchically generate the sitemap from the website homepage (the root), the root’s SBs must be extracted first. Following links within those SBs, pages of the second-level sitemap are collected and SBs of each second-level page are then identified for further analysis, and vice versa. Intuitively, SBs applied in the first-level sitemap should not be used in the second-level. However, “semantically identical” SBs are hard to be identified by exact matching on HTML tags and texts. As the example shown in Fig. 9, the root and second-level pages of Mozilla’s sitemap contain the “semantically identical” SBs that are written in “different but similar” HTML codes for highlighting different “clicked” categories. Moreover, in some websites, items in the same navigation menu may be changed according to different clicks. Hence, Block Cluster (BC) is proposed to find semantically identical blocks so that the system is able to avoid applying semantically identical SBs in different levels of the sitemap.

Obviously, semantically identical blocks usually share similar tags and texts so that corresponding sequences are highly similar. With the benefit of translating an HTML source into a sequence through ST, applying Basic Local Alignment Search Tool (BLAST) (Altschul, Gish, Miller, Myers, & Lipman, 1990) or Smith-Waterman’s sequence alignment algorithm (Smith & Waterman, 1981) to estimate similarities among block sequences is a simple and good solution. In this paper, BLAST was selected as the sequence alignment method due to its fast response with acceptable sensitivity. Moreover, BLAST, likes search engines, creates sequence indices for indexing once and searching unlimited times so that further analysis may get benefits from the BLAST service. The detail of BLAST service and software is available on the BLAST Tutorial of NCBI website[6].

[6] NCBI, "BLAST: Basic Local Alignment Search Tool," available at http://www.ncbi.nlm.nih.gov/BLAST/.

Table 2
Substitution matrix used in Block Cluster.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	7	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−4
R	−1	6	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	4	−1	−1	2	4	−1	−1	−1	−1	−4
N	−1	−1	9	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−4
D	−1	−1	−1	9	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	2	−1	−1	−1	−1	−1	−1	−1	−4
C	−1	−1	−1	−1	7	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−4
Q	−1	−1	−1	−1	−1	4	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−4
E	−1	−1	−1	−1	−1	−1	9	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−4
G	−1	−1	−1	−1	−1	−1	−1	7	−1	−1	−1	−1	5	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−4
H	−1	−1	−1	−1	−1	−1	−1	−1	7	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	4	−1	−1	−1	−4
I	−1	−1	−1	−1	−1	−1	−1	−1	−1	7	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−4
L	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	7	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−4
K	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	9	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−4
M	−1	−1	−1	−1	−1	−1	−1	5	−1	−1	−1	−1	7	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−4
F	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	4	−1	−1	−1	−1	−1	−1	−1	−1	−1	−4
P	−1	4	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	6	−1	−1	2	4	−1	−1	−1	−1	−4
S	−1	−1	−1	2	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	7	−1	−1	−1	−1	−1	−1	−1	−4
T	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	7	−1	−1	−1	−1	−1	−1	1
W	−1	2	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	2	−1	−1	6	2	−1	−1	−1	−1	−4
Y	−1	4	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	4	−1	−1	2	6	−1	−1	−1	−1	−4
V	−1	−1	−1	−1	−1	−1	−1	−1	4	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	7	−1	−1	−1	−4
B	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	4	−1	−1	−1	7	−1	−1	1
Z	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	4	−1	−1	−1	−1	7	−1	1
X	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	6	−4
*	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	1	−4	−4	−4	1	1	−4	1

By running the stand-alone BLAST software, all block sequences assigned with identifiers (Block ID) are formatted (indexed) into the BLAST database. Then, any sequence can be selected as a query sequence to match those sequences in the database. Block sequences similar to the query are retrieved and ranked to form the BLAST result. To align a query sequence with target sequences in the database, the substitution scoring matrix like BLOSUM62 (Henikoff & Henikoff, 1992), must be employed to evaluate substitution scores between amino acids. The substitution matrix of HTML tag mappings must reflect the meanings like amino acids; i.e., tags with similar presentation effects should be assigned with higher "substitution scores". In the system, the matrix is manually defined based on heuristics of designing web pages as shown in Table 2. BLAST default settings are used as arguments, such as blastp program and gap penalty. This part of research is not the kernel of the paper, we omit the detail. Related problems and studies can be referred to CLUSTAL W (Thompson, Higgins, & Gibson, 1994).

The sitemap is usually constructed from the homepage following links to lower-level pages. Therefore, BC applies the top-down approach to recursively match block groups so that semantically identical SBs of each page can be retrieved level-by-level to avoid being reused to form the hierarchical sitemap. The detail algorithm of BC is shown in following steps.

1. Set the similarity threshold θ to 0.9 (with high similarity). All SBs are initially marked "unused."
2. Add the homepage to the queue Q that is initially empty and set the tree root to the homepage.
3. While Q is no empty, get and remove the first page candidate (p_c) from Q and perform step 4–step 7.
4. For each "unused" SB b_i of p_c, get the block sequence as the query q. Run step 4–step 7.
5. *Find SB clusters as the candidates of links for the sitemap*. Match SB sequences of pages that are directly linked by the query SB

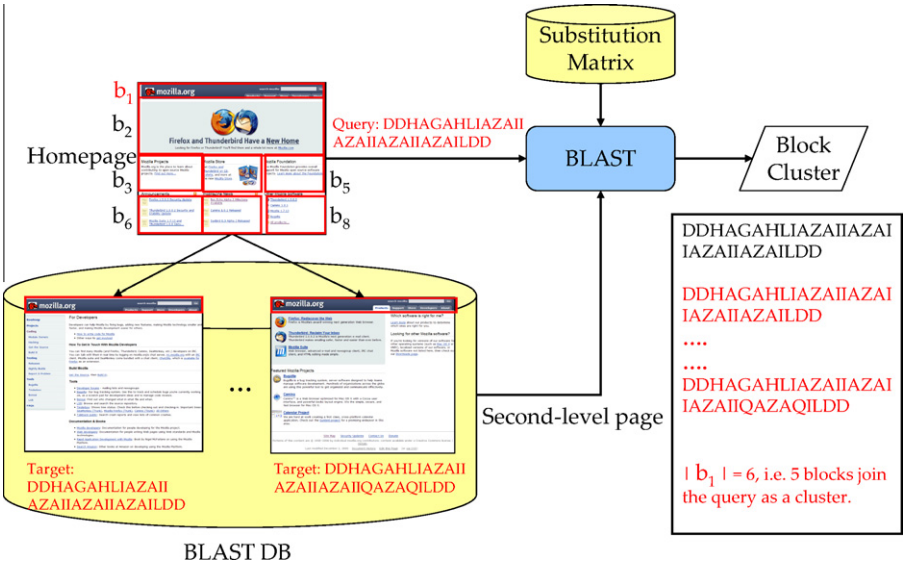

Fig. 10. An example of searching semantically identical blocks (SBs) from the root (homepage) to second-level pages in the Mozilla.

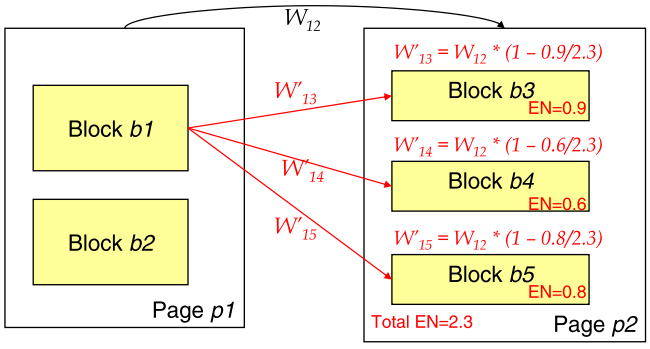

Fig. 11. An example of transforming page-page links weights to block-block links weights.

(b_i) of p_c, i.e., perform BLAST on the sequence database and filter off the search result based on the link relations. Normalize scores with the maximal one to fit the range [0, 1]. Similar SB sequences (with score $\geqslant \theta$) are grouped into a cluster of the query and the cluster size is denoted by $|b_i|$.

6. *Update the cluster information.* Match all block sequences, i.e., perform BLAST on the whole database. Update the cluster triple-information of b_i: ({blocks}, $|b_i|$, average score). Therefore, $|b_i|$ was updated to denote the block's total "*occurrence frequency*" within the website based on the "semantically identical measure".
7. As for a page with more than one SB, the largest two SB clusters (according to the updated $|b_i|$) are selected as SiteMap Blocks (SMB) to construct the sitemap "prototype". Pages pointed by links within SMB are added into Q and set as child pages of p_c; mark applied semantically identical blocks (SMBs) as "used", i.e., these blocks were already employed to the sitemap.
8. Output the tree started from the homepage (root).

An example of BC is shown in Fig. 10, in which blocks ($b_1 - b_8$) sequences of the root page are individual queries submitted to BLAST. Setting the BLAST DB to block (only SB) sequences within child pages of the root, similar block sequences are retrieved to form a block cluster of the query block. Consequently, only b_1 can be considered as a block cluster ($|b_1| = 6$, the query and five semantically identical blocks) since blocks $b_2 - b_8$ can not find any similar blocks, i.e., $|b_i| = 1$. Therefore, the block cluster of b_1 is selected as the SMB to generate the sitemap prototype.

4.5. Hyperlink Analyzer (HA)

BC builds a tree prototype for the sitemap by clustering similar SBs and applies links within these SBs to expand next-level pages transitively. However, expanding the sitemap by the only one parameter "occurrence frequency of SB" tends to bias while RB was incorrectly identified as SB. For example, the textual advertisement block may be identified as SB instead of RB. Usually, ad-links point to irrelevant pages to the website, the HITS algorithm may be useful by assigning each page of the site with authority and hub values. Intuitively, the "hub" value defined in HITS is also an important measure to identify sitemap pages since sitemap pages are probably linking to significant pages of the website. It is feasible to applied HITS-based algorithms to determine more qualified SBs and refine links within the sitemap prototype. However, most HITS-based algorithms are designed to aim at pages rather than blocks, the page-to-page relationships must be transformed into block-to-block relationships while measuring authority and hub values for each block.

Hyperlink Analyzer (HA) employs the Entropy-based HITS (EN-HITS) algorithm (Kao et al., 2002, 2004) to refine link weights as defined in (3). Using α_{uv} as the link weight of anchor ($u \to v$), in which the anchor text consists of k terms by using t_j to denote the jth term. The entropy value of t_j, $EN(t_j)$, is estimated according to the term distribution on the website pages (Lin & Ho, 2002). The entropy value is an inverse proportion to the information quantity, the term weight is therefore "one minus term entropy"). The link weight α_{uv} is therefore defined as the average weights of terms within the anchor text. The authority and hub values of page v are denoted by $A(v)$ and $H(v)$, respectively. S denotes the set of pages in the website. Consequently, the modified link weight of EN-HITS is better than HITS in identifying the important of the page (Kao et al., 2004). The remaining task is transforming page-to-page EN-HITS into the block-to-block mode.

$$A(v) = \sum_{v \in S | u \to v} H(u) * \alpha_{uv}, \quad H(v) = \sum_{u \in S | v \to u} A(u) * \alpha_{uv}$$
$$\alpha_{uv} = \frac{\sum_{j=1}^{k}(1 - EN(t_j))}{k} \tag{3}$$

By partition a page into blocks, block-to-page link structures can be constructed by the HTML parser since the block contains anchors pointing-to URLs of pages. However, we don't know which one is the "pointed-by" block of the "pointing-to" block's link. A trivial solution is "equally" sharing the link weight to all blocks of the pointed-by page. Since the block entropy is already measured in the BI (refer to the InfoDiscoverer method (Lin & Ho, 2002)) process, HA reuses the block entropy to adjust the link weight shared by blocks as the example shown in Fig. 11. Block *b1* of page *p1* contains one link (W_{12}) pointing to *p2*; the link is expanded to 3 block-to-block links with different weights based on blocks entropy values: W'_{13}, W'_{14} and W'_{15}.

Consequently, the block-to-block EN-HITS is defined in (4). Block p and block q belong to page v and page u, respectively. According to the idea of EN-HITS, blocks with high/low entropy

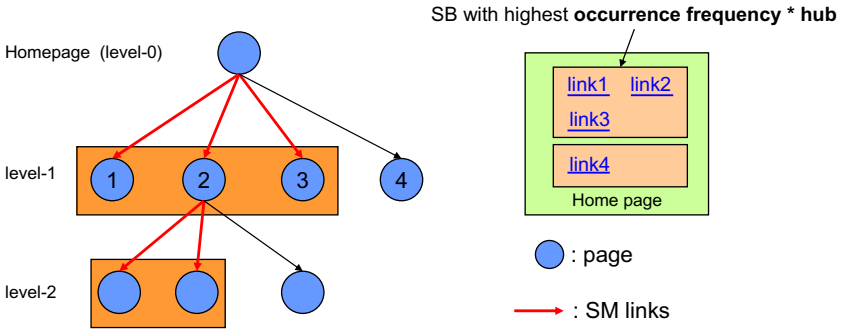

Fig. 12. An example of the sitemap generated by SMA of SMG.

Table 3
The statistical information of SM pages and labeled blocks for three tested websites.

Website	All pages	Crawl depth	SMP[a]	SM level-1	SM level-2	All blocks	CBs	SBs
Mozilla (www.mozilla.org)	5,242	6	25	4	21	213	67	146
CNN (edition.cnn.com)	6,070	5	30	13	17	116	10	106
Y!News (news.yahoo.com)	9,606	4	132	18	114	718[b]	301	417

[a] SMP (SiteMap Page): We merely gather pages that URLs are restricted by the hostname.
[b] We only label blocks from 34 (18 + 16) pages randomly selected from 132 SM pages due to its complex block structure and the time limitation.

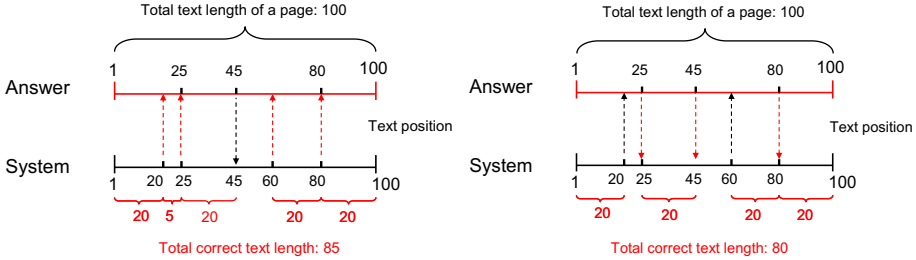

Fig. 13. Two criteria for measuring the effectiveness of PP.

values are often considered as redundant/informative. Thus a link weight is shared according to entropy values of blocks within the pointed-by page. As a block with lower entropy value is more informative, the block should obtain more link weight from the pointing-to page. Therefore, the modified link weight of block-to-block EN-HITS, β_{pq}, is defined in (5). The authority and hub values of each block can be converged after running about ten iterations.

$$\begin{cases} A(p) = \sum_{p \in v | q \to p} H(q) * \beta_{qp} \\ H(p) = \sum_{q \in u | p \to q} A(q) * \beta_{pq} \end{cases}, \text{where } u, v \in S \tag{4}$$

$$\beta_{pq} = \alpha_{uv} * \left(1 - \frac{EN(Block_q)}{\sum_{b \in v} EN(Block_b)}\right) \tag{5}$$

4.6. SiteMap Analyzer (SMA)

BC applies BLAST to search similar block sequences and group high-frequent semantically identical blocks as SMB candidates of the sitemap. HA transforms the page-to-page EN-HITS to the block-to-block mode and refines authority and hub measures of each "block". In this section, SiteMap Analyzer (SMA) summarizes these results to generate the sitemap automatically.

Obviously, SM links are usually and repetitively located in certain blocks of website pages. That is the reason to partition a page into blocks and identify each block type in the beginning. Also, SM links tend to appear in structure blocks so that SMA only focuses on SBs to find the SM links. Usually, SM links appear on top two or three levels of pages of the website's link structures; the hierarchically top-down analysis approach is trivially applied to find SM links started from the homepage. Assuming the root level is level-0, level-1 SM links are extracted from SB of the homepage. Following these level-1 SM links, level-2 SM links are expanded, and vice versa.

As sitemaps are used to facilitate users to browse websites easily, sitemaps usually contain the major categories of the website. Therefore, SM links probably appear on most pages of the website. Based on this characteristic, a block with high occurrence frequency is likely to contain SM links. The occurrence frequency of a block is denoted by the size of the block cluster extracted by BC. However, some blocks never contain SM links but occur frequently in the website, such as page footers and copyright announcements. Such kinds of blocks may be identified as SB since they contain anchors and rich-text information. Fortunately, these blocks, with high occurrence frequency, usually appear in the bottom of pages. Therefore, blocks located in the last k% part of a page should be redundant, where k, set to 20, is an adjustable system parameter. Consequently, SMA can effectively and efficiently filter out blocks like "copyright announcements" or "page footers". On the other hand, SM links usually point to the important pages that have relatively high authority values. These important pages will contribute higher hub values of blocks pointing to them; therefore, the block containing SM links gets higher hub value.

According to ideas mentioned above, the SM block should have high "occurrence frequency" and "hub values". Consequently, a block with the highest product of the occurrence frequency and the hub value within a page is finally regarded as SMB; links included in this block are used to construct the sitemap. The level-1 SM links are extracted from the homepage; SMB of level-1 pages are then analyzed to generate level-2 SM links by ignoring level-1 SMBs. Finally, the two-level sitemap is automatically constructed as shown in Fig. 12. Of course, building n-level ($n \geqslant 2$) sitemap is not a problem for SMG.

5. Experiments and evaluations

To evaluate the performance of SMG, three systematic websites with sitemaps provided by their webmasters, Mozilla, CNN and Yahoo! News (Y!News), are collected as the benchmark data. Only three websites are involved since labeling blocks and assigning types to these blocks for all pages within sitemaps is a hard work. Also, it is not easy to find many popular websites with webmasters' sitemaps as the objective answer set. The statistical information of

Table 4
Average precision/recall/F1 of PP measured from three websites.

Website	SMP	Ans. B	SMG B	Precision	Recall	F1
Mozilla	25	213	476	0.85	0.88	0.86
CNN	30	116	79	0.84	0.86	0.85
Y!News	132	718	839	0.88	0.86	0.87
Average	62.3	349	464	0.86	0.86	0.86

SMP (pages within the sitemap); Ans. B (blocks labeled in the answer set); SMG B (blocks extracted from these pages by PP).

Table 5
Average precision/recall/F1 of BI measured from three websites.

Website	CB	SB	P_{CB}	R_{CB}	$F1_{CB}$	P_{SB}	R_{SB}	$F1_{SB}$
Mozilla	279	197	0.79	0.98	0.88	0.91	0.77	0.83
CNN	9	70	0.93	0.76	0.84	0.75	0.95	0.84
Y!News	595	244	0.78	0.95	0.86	0.97	0.76	0.85
Average	294.3	170.3	0.83	0.90	0.86	0.88	0.83	0.84

Total numbers of SMP and Blocks (CB and SB) can be referred to Table 2.
First two columns "CB" and "SB" are numbers of blocks identified by BI. Remaining six columns are recall/precision/F1 measures for CB and SB.

the answer set corresponding to three websites is shown in Table 3. By focused crawling on pages within those websites, the number of crawled pages for each website and the crawl depth are listed in the first two columns. Following columns are details of the sitemap pages: all sitemap, level-1 and level-2 pages, respectively. These data will be applied to evaluate the overall performance of SGM. Assessors carefully inspect the content of each page and label blocks with CB or SB (don't care the RB). The last three columns are counts of blocks in the answer set corresponding to all blocks, CB and SB, respectively. The Last three columns are employed to evaluate the performance of "Block extraction" (PP and BI).

5.1. Evaluation results of Page Partitioner (PP)

Recall and precision measures widely used in Information Retrieval is employed to evaluate the overall performance of PP. However, both measures are based on the idea of matched objects between system predictions and answer sets. That is the "object matching" is binary: true or false. In this way, it is hard to evaluate the correctness of block matching since there may be overlaps between answer and predicted blocks. The overlap rate between blocks partitioned by PP and blocks labeled by human is used to replace the binary matching. Since blocks are contiguous, a predicted block may be matched with more than one block. Hence, the maximal matched block is used to estimate the ratio based on the overlapped text length. Given a page p, from system/answer perspective, both criteria denoted by $P_{S\to A}(p)$ and $P_{A\to S}(p)$ are corresponding to precision rates of "system matching answer" and "answer matching system". Both rates are corresponding to *precision* and *recall* rates of the traditional evaluation used in Information Retrieval. The former ($P_{S\to A}(p)$) is from the system perspective to evaluate the precision; the later ($P_{A\to S}(p)$) is from the human perspective to measure the recall. Let $\{a_1, a_2, \ldots, a_m\} \in A$ to be the answer block set of a page labeled by human; and let $\{s_1, s_2, \ldots, s_n\} \in S$ to be the block set of the page partitioned by PP. $P_{S\to A}(p)$ and $P_{A\to S}(p)$ are defined in (6) and (7), respectively.

$$P_{S\to A}(p) = \frac{\sum_{i=1}^{n} \max(|a_1 \cap s_i|, |a_2 \cap s_i|, \ldots, |a_m \cap s_i|)}{|p|} \quad (6)$$

$$P_{A\to S}(p) = \frac{\sum_{i=1}^{m} \max(|a_i \cap s_1|, |a_i \cap s_2|, \ldots, |a_i \cap s_n|)}{|p|} \quad (7)$$

For the instance shown in Fig. 13, assume that the total text length of the page is 100, in which four blocks are labeled by human as the answer set and five blocks are extracted by PP. Left and right figures illustrate the assessments of $P_{S\to A}(p)$ and $P_{A\to S}(p)$, respectively. In the evaluation of $P_{S\to A}(p)$, there are 5 overlaps, in which the third system block (length = 35) matches two answer blocks with lengths 20 and 15. According to (6), only the maximal overlap (length = 20) is counted. Therefore, match lengths of five system blocks are: 20, 5, 20, 20, 20. Obviously, the value of $P_{S\to A}(p)$ is 0.85. As for $P_{A\to S}(p)$, the first and third answer blocks match with two system blocks, length (20, 5) and length (15, 20), respectively. Hence, four match lengths are: 20, 20, 20, 20. Consequently, $P_{A\to S}(p)$ is 0.8. In this example, $P_{S\to A}$ (the precision) seems to be an optimistic evaluation while the system tends to extract smaller blocks in par with the answer. However, in this case, the evaluation of $P_{A\to S}$ (the recall) gets worse. On the other hand, if the system extracts larger blocks, the evaluation of $P_{S\to A}$ will become pessimistic, but $P_{A\to S}$ gets compensates. Therefore, both precision and recall rates are considered in the evaluation method. Consequently, the F-measure (F1) of PP, as shown in (8), is equal to 0.824 in this example.

$$F1(p) = \frac{2 \times P_{S\to A}(p) \times P_{A\to S}(p)}{P_{S\to A}(p) + P_{A\to S}(p)} = \frac{2PR}{P+R} \quad (8)$$

Based on above precision and recall measures, the performance of PP is evaluated by the average F of all SM pages of each website. The result is shown in Table 4. The average F1 of three websites is 0.86 that concludes PP is an applicable module. Also, the stability of PP is good without biases to precision or recall rates, and no matter simple (Mozilla) or complicated (Yahoo! News) website structures.

5.2. Evaluation results of Block Identifier (BI)

Based on the results of PP, the evaluation of BI is relatively simple since the precision rate of CB (P_{CB}) is corresponding to the $P_{S\to A}$ of "predicted" CBs in par with those CBs in the answer set. The recall rate of CB (R_{CB}) is the $P_{A\to S}$ of the match ratio of "answer" CBs. Precision and recall of LB are evaluated in the same way. F1 measure is also calculated as shown in Table 5. The result illustrates that the performance of BI is quite stable without biases to CB or LB. The average F1 for CB and LB is quite good (85%). Therefore, the results of PP and BI can be applied to extract CB and LB for SMG pages.

5.3. Evaluation results of SiteMap Generator (SMG)

Finally, we evaluate the correctness of sitemaps generated by SMG. Precision and recall rates are also employed to evaluate the overall performance of SMG. By setting the similarity threshold of Block Cluster to ($\theta = 0.9$), results of precision, recall and F1 are shown in Table 6. Considering the effectiveness of extracting the whole sitemap, the best F1 = 0.73 is obtained from Mozilla due to the higher regularity of page layout. The structure of Yahoo! News

Table 6
The precision, recall and F1 of SMG for three websites. ($\theta = 0.9$, All-level).

Website	SMP(A)	SMP(S)	SMP(AS)	Precision	Recall	F1
Mozilla	25	33	21	0.64	0.84	0.73
CNN	30	31	19	0.61	0.63	0.62
Y!News	132	161	81	0.50	0.61	0.55
Average	62.3	75	40.3	0.58	0.69	0.63

SMP(A), SMP(S) and SMP(AS) are numbers of sitemap pages corresponding to the answer set, SMG and "common pages of both", respectively.

Table 7
The precision, recall and F1 of SMG for three websites. ($\theta = 0.9$, Level-1).

Website	L1(A)	L1 (S)	L1(AS)	Recall	Precision	F1
Mozilla	4	4	4	1.0	1.0	1.0
CNN	13	22	13	1.0	0.59	0.74
Y!News	18	21	18	1.0	0.86	0.93
Average	62.3	75	40.3	0.58	0.69	0.63

L1(A), L1(S) and L1(AS) are numbers of level-1 sitemap pages corresponding to the answer set, SMG and "common pages of both", respectively.

Table 8
The precision, recall and F1 of SMG for three websites. (θ = 0.9, Level-2).

Website	L2(A)	L2(S)	L2(AS)	Recall	Precision	F1
Mozilla	21	29	17	0.81	0.59	0.68
CNN	17	9	6	0.35	0.67	0.46
Y!News	114	140	63	0.55	0.45	0.50
Average	50.7	59.3	28.7	0.57	0.57	0.55

is more complicated than other websites, the result (F1 = 0.55) is ordinary, but a half of the sitemap can be automatically built.

Since SMG hierarchically analyzes DOM trees of pages and blocks starting from the homepage, results of top two levels are individually evaluated as shown in Tables 7 and 8, respectively. Considering the results of level-1 sitemaps (Table 7), recall rates for three websites are perfect since all major categories (navigation menus) can be correctly extracted for their high occurrence frequencies and hub values. However, variations occur in precision rates. The reason results from that the identified SM blocks containing several links not belonging to the sitemap. By carefully verifying these "false-positive" links, we found these links are adequate to be added into the sitemap, although webmasters did not include them.

Results of level-2 sitemaps (Table 8) are more divergent. One reason results from the ambiguities between SB and RB. The average F1 on SB is 0.84 (Table 5), i.e., some RBs are recognized as SBs so that false-positive SM links are identified in the level-2 expansion. This case results in the lower precision rate. Another reason is that level-2 SM links are less regular than level-1 links. These links are even located in different kinds of blocks (CB instead of SB), especially to CNN; the recall of CNN is therefore low. According to this observation, level-2 SM links should not be restricted in SBs. In the future, the threshold θ, occurrence frequency and hub value should be analyzed to find an optimal combination.

6. Conclusion and future work

In this paper, we propose the SiteMap Generator (SMG) system to automatically extract sitemaps from systematic websites based on four novel methods: Page Partition (PP), Block Identification (BI), Block Clustering (BC) and block-to-block Hyperlink Analysis (HA). Each method can be an independent research topic and should be refined in the future. By translating an HTML DOM tree into a long sequence, PP applies SEG to analyze the sequence complexity by refining the granularity of sequence into segments based on the hierarchical DOM structure. The sequence complexity also contributes to the identification of block types in BI. Furthermore, BC employs BLAST to group similar blocks (with high sequence similarities) to identify important blocks for the sitemap. We also propose block-to-block hyperlink analysis algorithm from the traditional page-to-page HITS. Consequently, the idea of translating an HTML DOM tree into a sequence and using open sourced sequence-analysis tools facilitates SMG to seamlessly integrate these methods to be an automatic web agent system for extracting the website's topology.

We also perform experiments on three websites, Mozilla, CNN and Yahoo! News, to evaluate the effectiveness of these methods based on precision, recall and F1 measures. Experiments show that both PP and BI work well to extract blocks from pages (F1 = 86%) and identify block types (F1 = 86% and 84% for CB and SB), respectively. The result of the level-1 sitemap extraction is good (F1 = 89%). However, finding level-2 sitemap pages still has room to be improved (F1 = 55%). Nevertheless, the overall performance of SMG (F1 = 63%) is considered to be acceptable. These proposed methods are novel and still have room for improvement. For example, ambiguities between SB and CB should be clearly extracted to improve BI and enhance the performance SMG. In addition to generating a machine readable sitemap from a systematic website, SMG can also be used to find the most important blocks and links of the website for various web applications, such as "reading a website like an electronic book through smart phone or PDA".

In the future, we would continuously improve the performance of SMG by extracting more useful features from blocks and links. Extending the research of sitemap generation to extract the hierarchical domain knowledge of a website is quite interesting and important for the pre-subsystem while developing intelligent web agents. Given a website, the sitemap can be regarded as the initial domain knowledge for advanced data mining tasks; e.g., focused crawler, information extraction, website ontology, classification for web documents, catalog fusion from several domain-related websites, etc.

References

Agris, P. F. (2004). Decoding the genome: A modified view. *Nucleic Acids Research, 32*(1), 223–238.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology, 215*(3), 403–410.

Ashish, N., & Knoblock, C. (1997). Wrapper generation for semi-structured internet sources. *ACM SIGMOD Record, 26*(4), 8–15.

Bharat, K., & Henzinger, M. R. (1998). Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st ACM SIGIR conference on Research and Development in Information Retrieval*, New York, NY, USA (pp. 104–111).

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th international World Wide Web conference*, Brisbane, Australia (pp. 107–117).

Cai, D., He, X., Wen, J. R., & Ma, W. Y. (2004). Block-level link analysis. In *Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval*, Sheffield, UK (pp. 440–447).

Chakrabarti, S., Dom, B., Raghavan, P., Rajagopalan, S., Gibson, D., & Kleinberg, J. M. (1998). Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proceedings of the 7th international World Wide Web conference*, Brisbane, Australia (pp. 65–74).

Chakrabarti, S., Dom, B., Kumar, S., Raghavan, P., Rajagopalan, S., Tomkins, A., et al. (1999a). Mining the Web's link structure. *IEEE Computer, 32*(8), 60–67.

Chakrabarti, S., Van Den Berg, M., & Dom, B. (1999b). Focused crawling: A new approach for topic-specific resource discovery. *Computer Networks, 31*, 1623–1640.

Chakrabarti, S., Joshi, M., & Tawde, V. (2001). Enhanced topic distillation using text, markup tags, and hyperlinks. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*, New Orleans, LA, USA (pp. 208–216).

Debnath, S., Mitra, P., Pal, N., & Giles, C. L. (2005). Automatic identification of informative sections of web pages. *IEEE Transactions on Knowledge and Data Engineering, 17*(9), 1233–1246.

Diligenti, M., Coetzee, F. M., Lawrence, S., Giles, C. L., & Gori, M. (2000). Focused crawling using context graphs. In *Proceedings of the 26th international conference on Very Large Data Bases*, Cairo, Egypt (pp. 527–534).

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition, 5*(2), 199–220.

Henikoff, S., & Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences, 89*(22), 10915–10919.

Hsu, C. W., & Lin, C. L. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks, 13*(2), 415–425.

Kao, H. Y., Lin, S. H., Ho, J. M., & Chen, M.S. (2002). Entropy-based link analysis for mining web informative structures. In *Proceedings of the 11th ACM international conference on Information and Knowledge Management*, McLean, VA, USA (pp. 574–581).

Kao, H. Y., Lin, S. H., Ho, J. M., & Chen, M. S. (2004). Mining web information structures and contents based on entropy analysis. *IEEE Transactions on Knowledge and Data Engineering, 16*(1), 41–55.

Kleinberg, J. M. (1998). Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th ACM SIAM Symposium on Discrete Algorithms*, Baltimore, MD, USA (pp. 668–677).

Kushmerick, N., Weld, D. S., & Doorenbos, R. B. (1997). Wrapper induction for information extraction. In *Proceedings of the fifteenth International Joint Conference on Artificial Intelligence*, Nagoya, Japan (pp. 729–737).

Lee, K. C., & Lee, S. (2003). A cognitive map simulation approach to adjusting the design factors of the electronic commerce web sites. *Expert Systems with Applications, 24*, 1–11.

Lin, S. H., Ho, J. M. (2002). Discovering informative content blocks from web documents. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, Edmonton, Canada (pp. 588–593).

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). The pagerank citation ranking: Bringing order to the web. *Technical report, Computer Systems Laboratory, Stanford University*.

Sethi, A., O'donoghue, P., & Luthey-Schulten, Z. (2005). Evolutionary profiles from the QR factorization of multiple sequence alignments. *Proceedings of the National Academy of Sciences of the United States of America, 102*(11), 4045–4050.

Setubal, J., & Meidanis, J. (1997). *Introduction to Computational Molecular Biology*. Boston, MA: PWS Publishing Company.

Sigrist, C. J. A., Cerutti, L., Hulo, N., Gattiker, A., Falquet, L., Pagni, M., et al. (2002). PROSITE: A documented database using patterns and profiles as motif descriptors. *Briefings in Bioinformatics, 3*(3), 265–274.

Smith, T. F., & Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology, 147*(1), 195–197.

Song, R., Liu, H., Wen, J. R., & Ma, W. Y. (2004). Learning block importance models for web pages. In *Proceedings of the 13th international conference on World Wide Web*, New York, NY, USA (pp. 203–211).

Thompson, J. D., Higgins, D. G., & Gibson, T. J. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Research, 22*(22), 4673–4680.

Wootton, J. C., & Federhen, S. (1993). Statistics of local complexity in amino acid sequences and sequence databases. *Computers and Chemistry, 17*(2), 149–163.

Wootton, J. C., & Federhen, S. (1996). Analysis of compositionally biased regions in sequence databases. *Methods in Enzymology, 266*, 554–571.

Yi, L., Liu, B. (2003). Eliminating noisy information in web pages for data mining. In *Proceedings of the 9th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, Washington, D.C., USA (pp. 296–305).

Yu, S., Cai, D., Wen, J. R., & Ma, W. Y. (2003). Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *Proceeding of the 12th international conference on World Wide Web*, Budapest, Hungary (pp. 11–18).