MenuMiner: Revealing the Information Architecture of Large Web Sites by Analyzing Maximal Cliques

Matthias Keller
Steinbuch Centre for Computing (SCC)
Karlsruhe Institute of Technology (KIT)
D-76128 Karlsruhe, Germany
matthias.keller@kit.edu

Martin Nussbaumer
Steinbuch Centre for Computing (SCC)
Karlsruhe Institute of Technology (KIT)
D-76128 Karlsruhe, Germany
martin.nussbaumer@kit.edu

ABSTRACT
The foundation of almost all web sites' information architecture is a hierarchical content organization. Thus information architects put much effort in designing taxonomies that structure the content in a comprehensible and sound way. The taxonomies are obvious to human users from the site's system of main and sub menus. But current methods of web structure mining are not able to extract these central aspects of the information architecture. This is because they cannot interpret the visual encoding to recognize menus and their rank as humans do. In this paper we show that a web site's main navigation system can not only be distinguished by visual features but also by certain structural characteristics of the HTML tree and the web graph. We have developed a reliable and scalable solution that solves the problem of extracting menus for mining the information architecture. The novel MenuMiner-algorithm allows retrieving the original content organization of large-scale web sites. These data are very valuable for many applications, e.g. the presentation of search results. In an experiment we applied the method for finding site boundaries within a large domain. The evaluation showed that the method reliably delivers menus and site boundaries where other current approaches fail.

Categories and Subject Descriptors
H.5.4 [**Information Systems**]: Information Interfaces and Presentation – *Hypertext/Hypermedia;* H.3.1 [**Information Systems**]: Information Storage and Retrieval – *Content Analysis and Indexing*

General Terms
Algorithms, Experimentation, Languages

Keywords
Web structure mining, Site boundaries, Site hierarchies, Search result presentation

1. INTRODUCTION
Information architecture is crucial for the success of a web site. Engineering the information architecture means labeling hundreds or thousands of resources, organizing them with coherent schemas and developing understandable systems for accessing them. The technical model of content organization are database schemas while the human model is the information architecture. Information architecture is a way of communication. This is because users must be able to decode the information architecture in order to interact with the information system in the intended way.

Decoding the information architecture means that humans are able to understand content hierarchies, to distinguish navigation elements from page content and to learn the purpose of navigation systems. We can recognize entry pages of web sites and we notice if we cross site boundaries. Because of the visual encoding these structures are obvious to humans but not to machines. If a site contains a reasonable number of pages it is usually not very complicated for an average human to model the site map but it is so for machines, even for Google[1]. For a site map to be included in the search result presentation the site map has to be provided to Google as XML-file. Displaying the position of a page in the content hierarchy is supported by Google (Figure 1), but it works currently only for a small amount of destination pages that contain a similar presentation ("breadcrumbs"), which can be extracted[2].

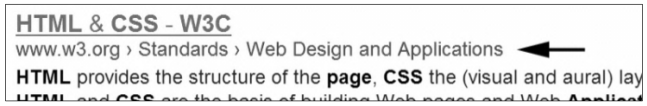

Figure 1. Google is able to display the position in the content hierarchy only for a few pages.

Though research has focused on extracting hierarchies from web sites (e.g. [1],[2],[3]), no solution has been found yet that could compete with human judgment. The same applies for the problem of mining web site boundaries or subsites (e.g. [4], [5]), site entry pages [5] or compound documents [6]. There still is a semantic gap between information architecture as perceived by humans and structure information that can be retrieved by current approaches. The seemingly simple problem of extracting the main navigation elements of a site for reverse engineering the information architecture is hard to solve in practice. The straightforward approach of using heuristics for identifying the menus on all pages of a site and then combining frequent structures is difficult and error-prone [7]. Thus up until now very valuable semantic information is not available for machine processing. The hierarchical content organization of a web site is a human-made, well-designed classification scheme for each page. Mining these taxonomies would not only allow a search result representation as shown in Figure 1 for all sites. For example, it would also allow visualizing all results from a certain site as tree structure. The

WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

[1]Some tools exist to automatically generate sitemaps (e.g. http://www.powermapper.com), but the quality of the results strongly depends on the examined site

[2] The Official Google Blog: New site hierarchies display in search results, http://googleblog.blogspot.com/2009/11/new-site-hierarchies-display-in-search.html

original site hierarchy can also be used to improve ranking, e.g. if a search returns many hits from a certain branch of a site hierarchy, the item at the root of the branch should be ranked first. A method for mining the original content hierarchy of web sites will also open new doors in research topics as web site abstraction and classification, information architecture reverse-engineering and automated usability testing. As a first application we demonstrate how the problem of web site boundary detection can be solved by using the MenuMiner-approach.

2. CONTRIBUTION

In this paper we show how the global and local menus that represent the central concept of content access and the main hierarchical structure of web sites can be mined. To our knowledge until now no other method exists that solves this problem well enough that the data can be used, e.g. for the presentation of search results. The presented approach of analyzing maximal cliques for mining menu systems is a novel and applicable solution that is highly accurate and not prone to spam. It scales and the time complexity is linear in the number of processed pages.

As first application we used the algorithm to solve a known problem of web site structure mining: the detection of site boundaries ([4],[5]). We used the algorithm to find site boundaries in a collection of 10,000 pages retrieved from a large domain. A method for manually validating and comparing site partitions is described. The evaluation showed that the site boundaries delivered by the MenuMiner-algorithm are much more precise than the hierarchy contained in the URLs (subdomains and folders) and those retrieved with clustering methods that performed best in a previous experiment [4]. But the central finding is that the MenuMiner-method extracts the main menus that distinguish sites exactly as perceived by humans. In the evaluation section we also point out by examples how the method described in this paper can solve other problems of web site structure mining such as hierarchy extraction, compound document mining and entry page detection.

3. PROBLEM DEFINITION AND APPROACH

The original content organization of web sites as perceived by humans cannot be extracted today because a reliable method for mining menu systems is lacking. This involves a method to distinguish main or sub menus from other page content and a method to identify recurring menus that are shared by a site or sub section.

3.1 Mining Shared Menus

The straightforward approach is to first split all pages into smaller content segments and to identify segments that represent navigation elements. Then an inter-page analysis can be conducted to find recurring navigation elements that are shared by a set of pages. That is basically the method used in current works that involve the mining of navigation elements (cf. Sect. 7). Our previous work on this topic [7] was based on the same approach. We used an extended set of heuristics to identify navigation elements and conducted an experimental evaluation that was lacking before. Although this method was able to detect most of the shared menus correctly we found that it has inherent limitations that prevent a really satisfying precision. The problem is that the intra-page analysis is prone to errors if the page segmentation does not precisely reflect the bounds of individual menus and sub menus. Consider the example in Figure 9. The same menu has additional items (“Samples”, “Forums”) on the page “Home” that are missing on the page “Library”. By comparing simply the hyperlinks in navigation bars both elements could not be matched. A case like this is not a rare exception but a rather common situation.

Thus a better approach is to compute the percentage of shared hyperlinks and applying a threshold. Rodrigues et al. [8] used a threshold of 0.6 while we were using a threshold of 0.5 in combination with two other metrics. But such a threshold leads to the problem that often elements are matched that do not belong to the same navigation system. Another problem is that sometimes a menu can be distinguished from a submenu only by its visual properties. Then even the most sophisticated algorithm will have difficulties in providing the correct segmentation. With the additional links of a submenu on one page that is not displayed on another page the percentage of shared links can fall below any threshold. Originally we were planning to apply machine learning methods for adjusting the heuristics and improve the precision of the method. Instead we found a different approach that completely avoids the described problems.

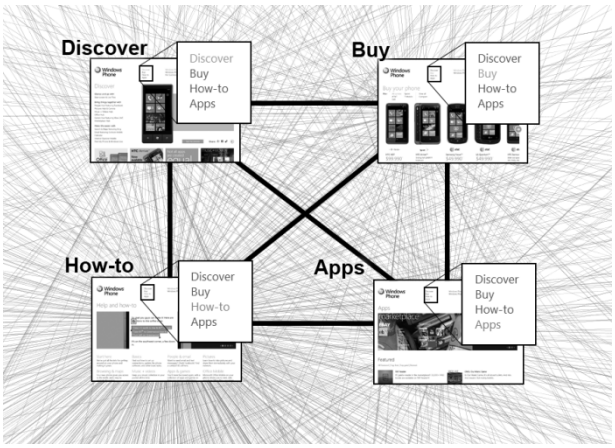

Figure 2. S-menus define cliques in the web Graph. The bold edges represent bidirectional links (example from www.microsoft.com/windowsphone).

3.2 S-Menus

To avoid transitional volatility (cf. Sect. 6.2) the navigation elements that play the key role for the content access are usually invariable elements in page transitions. These global and local menus allow not only navigating *from* one page to another but navigating *over* a group of pages. By this we mean, that we can use a single menu to traverse a group of pages. When we click on an item in the menu and move to the next page the menu is still present. According to [8] such menus will be referred to as structural-menus or s-menus in this paper. Of course not all navigation elements are s-menus. For example, a list of links related to a certain resource or a group of external links do not have the described characteristics. S-menus are the skeleton of the information architecture of sites that are based on menus[3]. Because of their invariability the function of s-menus is not only to provide paths through the information space but also to communicate the organization of the content. Thus s-menus are most suitable for mining the organization of web sites.

[3] Other sites are based on search. The MenuMiner-approach allows distinguishing both types of sites as shown in Sect. 6.3

The most common model of web structure mining is the web graph whose vertices represent pages and whose edges are given by the hyperlinks. As Rodrigues et al. [8] have observed before, s-menus define cliques in the web graph – complete sub graphs in which every two vertices are connected by an edge. This is because the menu itself is also present on the linked pages. Figure 2 shows an example where all pages that are linked in a menu contain the menu itself and thus have links to the other three pages. Instead of using a set of heuristics to identify navigation elements we propose to mine page segments that form cliques in the Web graph.

While the figure only shows the four target pages of the menu, the menu is present on many more pages. In general on almost all sites the main menu is displayed on all pages. Conversely, s-menus define the boundaries of the sites. Other s-menus that are present only on a subset of the pages consequently represent sub sites. Thus the inclusion relation on the sets of pages that share an s-menu reproduces the hierarchical structure of a site.

3.3 Mining S-Menus by Analyzing Cliques

In this section we describe the problem that has to be solved to recognize s-menus by the clique feature. To illustrate that this is not a trivial task we have placed the screenshots in Figure 2 on top of the real web graph. The web graph can be turned into an undirected representation with fewer edges by only keeping bidirectional edges. The graph would have a lower maximum degree Δ and all maximal cliques – cliques that are not part of larger cliques – could be enumerated in $O(\Delta^4)$ time [9]. Still this would not solve the problem of mining s-menus, because these cliques can result from all links on the pages. Instead we need to find page segments that form cliques. We will refer to this kind of cliques as segment cliques, to distinguish them from cliques in the web graph.

The page segment k of page p_i can be considered as a set of $M_{i,k}$ target pages of the hyperlinks it contains (we assume that all hyperlinks that are unidirectional in the web graph are removed in advance):

$$S_{i,k} = \{p_{k_1}, p_{k_2}, \dots, p_{k_{M_{i,k}}}\}$$

Let SE be the set of all $S_{i,k}$ that are all segments of all pages of a domain. SE defines a graph G_{SE} whose nodes are the segments. For a target page $p_x \in S_{i,k}$ this graph contains edges from $S_{i,k}$ to all segments $S_{x,l}$ of p_x. We can define the set of candidate cliques as follows:

$$SC = \left\{ \begin{array}{c} C \in \mathcal{P}(SE): \forall S_{i,k}, S_{j,h} \in C \\ \left[(p_i \neq p_j \Leftrightarrow i \neq j) \wedge (p_i \in S_{j,h}) \wedge (|C| > 2) \right] \end{array} \right\}$$

SC contains sets of segments, each from a different page (first condition). The sets in SC are cliques in G_{SE} because if a segment of a certain page is part of a set, this page also has to be a target page of all other segments in the set (second condition). And finally in the context of s-menus we are only interested in cliques with at least three nodes. Two segments on two pages that contain a link to the other page define a clique of two and this is certainly not enough to consider both segments as s-menus.

Of course not all cliques of page segments in SC represent s-menus. Additional considerations are necessary to find a subset SC^* of SC that is a good representation of s-menus. One consideration is that one certain link can surely be part of only a single menu. On the other hand it should be allowed for a page segment to be part of more than one clique in SC^*. Figure 3 shows an example of page segments that represent a simple hierarchical menu. The pages p_1-p_4 are the top level pages and the pages p_5-p_7 are part of a submenu under page p_2. In this example the page segmentation algorithm has failed to separate the menu levels, which is what often happens as described above. The light gray edges are the edges of the web graph that have been removed because they are not bidirectional. The menu defines two cliques for each level and by the clique method we are able to separate the s-menu of the first level from the s-menu of the second level. While the Segment S is part of both cliques, the cliques do not share an edge (hyperlink).

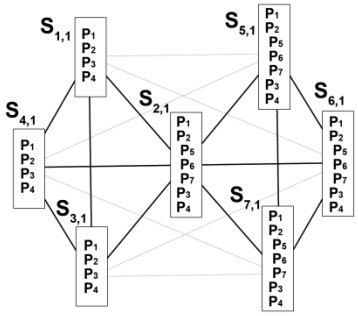

Figure 3. A menu can define multiple cliques

Another consideration is that larger cliques are more likely to be s-menus than smaller cliques. For this reason larger cliques should be preferred over smaller ones. Regarding cliques with the same size we should prefer the clique whose segments are more uniform, because it is more likely that the segments of this clique really belong to the same navigation system.

Let r be a scoring function that rates the uniformity of a clique based on the segments that define it. Resulting from the considerations an iterative procedure to find a subset SC* can be derived:

1. Find $C_i \in SC$:

$$\forall C_j \in SC \left(|C_i| \geq |C_j| \wedge \left(|C_i| = |C_j| \Rightarrow r(C_i) \geq r(C_j) \right) \right)$$

2. Add C_i to SC^*
3. Update SC by removing C_i and all subsets of C_i:

$$SC := SC \setminus \mathcal{P}(C_i)$$

4. Let P_{Ci} be the set of pages, of which a segment is contained in C_i. Those are the pages that form the clique. Update SC by removing all cliques from SC that share an edge with C_i:

$$SC := SC \setminus \left\{ C_j \in SC : \left| P_{C_j} \cap P_{C_i} \right| > 1 \right\}$$

5. If $|SC| > 0$, go back to step 1

In step 1 the largest clique with the highest score is selected. C_i represents a set of page segments that belong to the same menu. Of course all subsets of C_i belong to this menu too and can be removed from SC in step 3. This is necessary because SC is formulated to contain not only maximal cliques but all cliques. The reason for this is that subsets are relevant because in step 4 all cliques are removed that share at least two nodes, and thus a

hyperlink, with C_i – since a hyperlink can only belong to one menu. Subsets of the removed cliques can now become maximal cliques and candidates for *SC**.

The procedure above is an illustration and formalization of the method we used to find sets of page segments that most likely represent s-menus. In this form it is not an algorithm that can be used to compute the sets. The problem is the generation of *SC* that cannot be achieved by solely using the known algorithms for solving the clique problem (e.g. [9]).

First the size of the graph whose nodes are given by all segments of all pages exceeds the size of the web graph by far and the computational costs would be high. Second, only cliques consisting of segments from different pages are allowed in *SC*. Thus the retrieved maximal cliques would still have to be split to fulfill this condition somehow. In Sect. 5 we present an efficient algorithm to generate *SC** without computing *SC*.

4. PAGE SEGMENTATION

The proposed approach mines menus by finding segments on different pages that define cliques in the web graph. Thus, the method of page segmentation is crucial. It is important that two different navigation elements are not merged into one segment. It does not matter on the other hand if a menu and a submenu belonging to the same navigation element are merged (cf. Figure 3). Other methods of page segmentation as described in [10] or [11] use heuristics in combination with visual attributes as background color or element size.

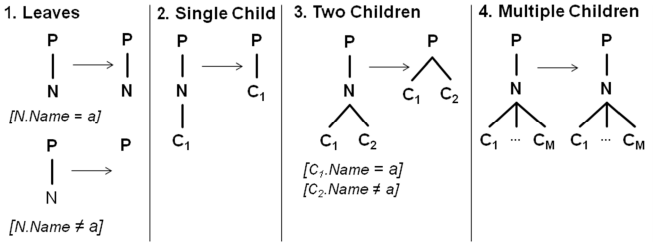

Figure 4. DOM transformation rules

The method we implemented in contrast does not depend on heuristics and does not require any layout related information. Since the DOM tree [12] reflects very well the visual and logical organization of a page, our method transforms the DOM in a way that only the hierarchical intra-page-structure of the hyperlinks remains. For this all nodes of the DOM tree are processed bottom up and four rules are applied to each node (Figure 4).

Because we are only interested in hyperlinks, leaves that are not hyperlinks are deleted (rule 1). If a node has a single child, the node is deleted and the child is appended to the parent node (rule 2), because these nodes do not provide structural information. If a node has two children and the first one is a hyperlink while the other is not, the node is deleted and both hyperlinks are appended to the parent node (rule 3). Nodes with more than two children or nodes with two children who do not fulfill the condition of rule 3 are left unchanged (rule 4). These nodes represent page segments.

Rule 3 covers special cases where rule 2 and rule 4 interfere and destroy the logical structure. Figure 5 shows a typical DOM tree of a menu with a submenu beneath the third hyperlink. Without rule 2 the third hyperlink is assigned to an additional segment even though it logically belongs to the same segment as the other three links of the first level. The inner nodes of the remaining tree constitute the segments. Though the segments retrieved this way are organized hierarchically, this information is not necessary for the proposed approach. Instead, a flat list of page segments is created with each segment containing only its direct hyperlink-children.

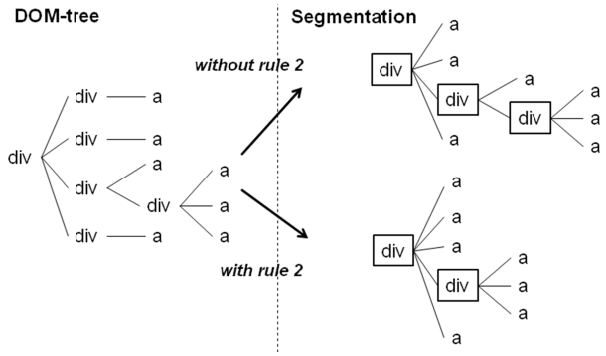

Figure 5. Necessity of rule 2

5. SEGMENT CLIQUE DETECTION

5.1 Overview

By applying the page segmentation algorithm we obtain a larger graph G_{SE} (cf. Sect. 3.3) whose nodes represent segments instead of pages. A single hyperlink in a segment defines multiple edges to all segments of the target page. From the set *SC* that contains all segment cliques the subset *SC** shall be extracted according to the described concept.

One observation that can be utilized for computing *SC** efficiently is that segment cliques in *SC* correspond to cliques in the web graph. Given a set of segments from different pages, this set cannot belong to *SC* if the pages do not form a clique in the web graph. Thus the maximal cliques in the web graph can be computed and afterwards decomposed in order to compute *SC**.

The second observation is that *SC** can be computed locally by processing page after page. To find all the s-menus a certain page belongs to, only its neighbors in the web graph have to be considered because only they can be part of the same clique. We preferred the local approach to reduce the memory requirements. The pages indexed by the crawler are analyzed one after another. Loaded pages are cached to avoid multiple requests for the same resource.

For computing *SC** locally our implementation proceeds as follows:

1. Build local web graph from current page and all its neighbors.
2. Reduce local web graph. Hyperlinks that belong to segments of the neighboring pages that do not contain a hyperlink the current page are removed. Those segments cannot form a clique with segments from the current page.
3. Remove unidirectional edges to obtain an undirected graph.
4. Compute maximal cliques. An implementation of the Bron-Kerbosch algorithm [13] is used to enumerate all maximal cliques.
5. Use the SegmentCliqueFinder algorithm (see next section) to decompose the web graph cliques and compute the local *SC**.

5.2 Decomposing Web Graph Cliques

The SegmentCliqueFinder algorithm returns the local *SC** which is the set containing all cliques representing s-menus to which a segment of the processed page belongs. Given is the list of maximal cliques in the partial web graph defined by the current

page and all its neighbors. According to Sect. 3.3 the algorithm successively computes the largest clique of segments from different pages and removes all links that are part of that s-menu. It terminates if no segment clique of a minimal size of 3 is found. If multiple segment cliques with maximal size are possible the algorithm returns the one whose segments are most uniform concerning their placement in the DOM tree.

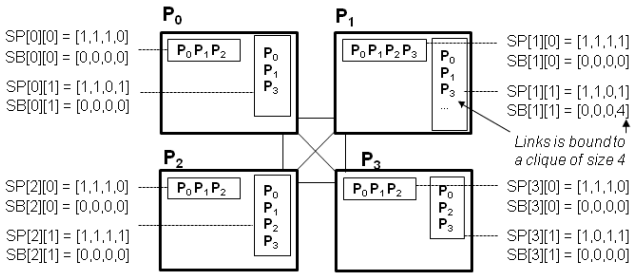

Figure 6: Four sample pages that form a clique in the web graph

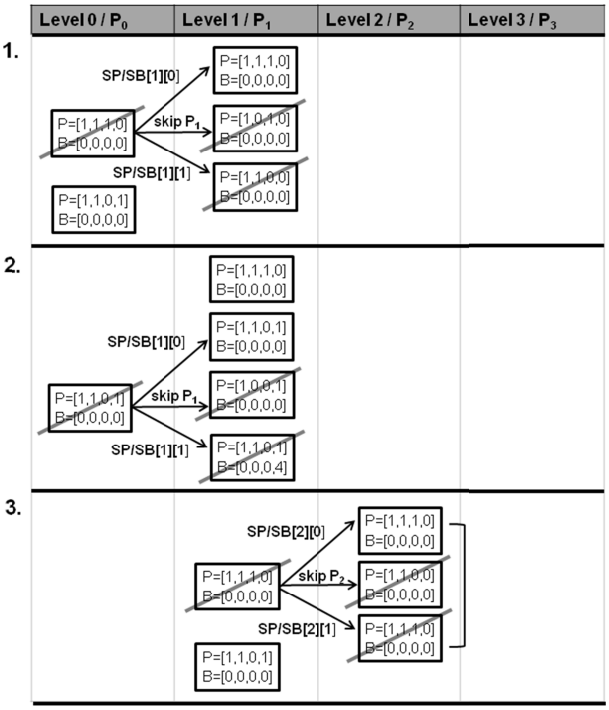

Figure 7. Illustration of the SegmentCliqueFinder algorithm

One, multiple or no segment clique can be embedded in a web graph clique. For each page of a web graph clique *WC* that contains a segment clique *SC*, a segment on that page is either part of *SC* or not, because *SC* can be smaller than *WC*. To avoid testing all possible combinations a greedy approach can be used. Figure 6 shows an example of four pages p_0-p_4 that form a clique in the web graph. The menu in the upper left corner of each page is a typical main menu with links to p_0-p_3 and an additional link in p_1. The first page p_0 is the page that is currently processed, so only segments containing links to p_0 have to be considered on the other pages. The segments are encoded as a three dimensional array *SP[i][j][k]* that has the value 1 if the segment *j* on page *i* contains a hyperlink to page *k* and the value 0 otherwise. The segments on pages p_1-p_3 can contain links that are already assigned to a segment cliques when previous pages were processed. Since according to 3.3(3) larger segment cliques are preferred, these links have to be removed from the original clique and assigned to a new segment clique if the new clique is larger. For this the array *SB[i][j][k]* contains the minimum sizes that a new segment clique must have to include a hyperlink or target page respectively. In the example shown in Figure 6 only the link to p_3 in segment 1 of p_1 is bound to another clique, which has the size of 4. The other links can be assigned to segment cliques of any size so SB has the value 0.

Each web graph clique is compared to all segments of p_0. If the web graph clique and the segment *j* share at least three pages, the segment arrays *SP[0][j]* and *SB[0][j]* become initial states of the algorithm by removing from *SP[0][j]* the pages that are not shared. The initial states, *SP* and *SB* are the input of the *SegmentCliqueFinder* algorithm. The example includes only a single web graph clique and both segments of p_0 are added to the initial state list. The ordered list of pages represents levels that the states have to pass to become end states. The initial states are associated with level 0 or page p_0 respectively. In each iteration of the algorithm one state is removed from the list and processed (line 04, function *GetStateWithMaxScore*). From the states with the maximal number of pages the state associated with the smallest level is selected.

In Figure 7 both initial states contain three pages and both states are associated with level 0, so a random state is picked. A new state is created that represents a segment clique that does not contain the page of the next level (Algorithm 1, lines 10-12). New states are also created for each segment of the page representing the next level (line 13). If such a segment does not contain a target page it is removed from the target pages of the state (line 16) and the binding of all links is updated (line17). Before a new state is added to the state list it is tested to see if links are already bound to larger segment cliques. If that is the case these links are removed (lines 21-23). If the segment clique represented by the state is still larger than 2 and the state has not reached the final level, it is added to the list *States* (line 25). The function *AddToStateList* consolidates the state list by applying a scoring function that measures the uniformity of the page segments that are included in a state (to fulfill 3.3(3)). If an equal state (regarding *SP* and *SB*) associated with the same level already exists only the state with the higher segment uniformity is kept. To compute the uniformity we align the DOM paths of all segments. Node names, class- or id-attributes that differ are replaced by wildcards. A lower number of wildcards indicated that the segments are placed at similar positions in the page templates and it is more likely that their visual representation is similar too. In the example shown in Figure 7(3) the state that is joined with *SP[2][0]* is kept because its segments are more uniform.

New states that have reached the maximal level are added to the list of end states only if the list does not contain an end state that represents a larger clique. If there are end states that represent smaller cliques these end states are removed (lines 26-30). The algorithm terminates if no other end states of at least the same clique size than the current end states can be reached. The end state with the highest uniformity is then returned (line 06) if one or more valid end states were generated. For the finding of all segment cliques of *SC** in which the page is part of, *SP* has to be updated by removing all links that are bound by the returned end state and the algorithm has to be executed again until no more end states can be found.

Algorithm 1: SegmentCliqueFinder

Input: *States* (initial states), *SP* (target pages of segments), *SB* (clique binding of segments)

Output: Maximal clique of page segments

```
01: EndStates ← new List; MaxEndScore ← 0;
02: WHILE (States.count > 0)
03:     NewStates ← new List;
04:     S ← GetStateWithMaxScore(States);
05:     IF (∑j S.pages[j] < MaxEndScore)
06:         RETURN BestState(EndStates);
07:     NextLevel = S.level+1;
08:     States.remove(S);
09:     IF (S.level < M - 1)
10:         SN ← S.copy();
11:         SN.pages[NextLevel] ← 0;
12:         NewStates.add(SN)
13:     FOR(all segments k of page P_NextLevel)
14:         SN ← new State;
15:         FOR(all Pj)
16:             SN.pages[j] ← min(S.pages[j], SP[NextLevel][k][j]);
17:             SN.binding[j] ←
                    max(S.binding [j], SB[NextLevel][k][j]);
18:         SN.level ← NextLevel
19:         NewStates.add(SN);
20:     FOR(all States SN in NewStates)
21:         FOR(i = 0…M)
22:             IF(SN.binding[i] > ∑j SN.pages[j])
23:             SN.pages[i] ← 0; i ← 0;
24:         IF (∑j SN.pages[j] > 2)
25:             IF (SN.level < M-1) AddToStateList(States, SN);
26:             ELSE IF (∑j SN.pages[j] = MaxEndScore)
27:                 EndStates.add(SN);
28:             ELSE IF (∑j SN.pages[j] > MaxEndScore)
29:                 MaxEndScore ← ∑j SN.pages[j];
30:                 EndStates ← new List; EndStates.add(SN);
```

6. EVALUATION

To evaluate the proposed method we analyzed 10,000 pages downloaded from microsoft.com. This domain was chosen because of its size and the diversity of the content and sub sites. Only pages targeting the US audience were indexed by testing for the substring "en-us" in the URL. The web crawler retrieved the pages in breadth-first order. Since we did not perform a full crawl of the domain and all neighboring pages are necessary for analyzing a page, a total number of 74,198 pages were downloaded. This overhead can be avoided if either a complete crawl of a domain is performed or the boundaries of the crawled space are defined in another way in advance.

6.1 Runtime Performance

The algorithms proposed in this paper are low resource consuming. We were able to conduct the experiment with a single Pentium D, 3 GHZ machine equipped with 3 GB RAM. Running the Bron-Kerbosch algorithm to enumerate the maximal local web graph cliques only required a mean execution time of 0.15ms. For the SegmentCliqueFinder algorithm, we measured a mean execution time of 2.33ms. Interestingly there were few pages that required a much longer processing time, up to a maximum of 371ms while for more than 87% of the pages the execution took no longer than 2ms. We measured the number of input pages, the number of input segments, the number of input cliques and the number of output segment cliques. We found that the number of input cliques correlates most strongly with the execution time (Figure 8). The mean execution time seems to increase almost linearly with the number of input cliques. In our experiment there were very few pages with more than 100 cliques but this might be different in the general case. However, the algorithm is able to process a higher number of cliques in a reasonable time as the plot shows. If we assume a maximal complexity of the local web graph, the algorithm has a linear-time complexity in the number of processed pages because all pages are analyzed independently.

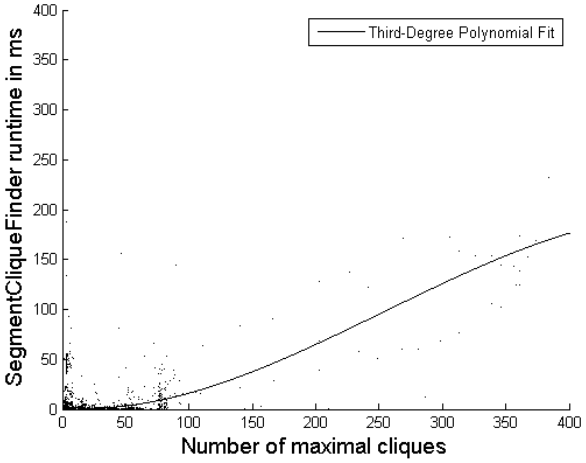

Figure 8. Number of input cliques vs. SegmentCliqueFinder runtime in ms

6.2 Web Site Boundaries

To evaluate the reliability of the method and its potential for solving problems in the field of web structure mining we applied it to detect site boundaries. The result shows that the approach is more accurate than existing methods and that the shared menus found are exactly the site-wide navigation systems as perceived by humans.

6.2.1 Definition and Approach

Often a number of different web (sub) sites are hosted under the same domain. In the case of the domain microsoft.com there are for example the Windows site, the Office site, the MSDN site and many more. Identifying site boundaries is useful for processing crawled domains in many ways ([4], [5]). But it is also one of the tasks that are easy for humans but difficult for machines. According to Nielson [14] a sub site is defined by three characteristics:

a. A common style
b. A shared navigation mechanism
c. An entry page

This definition was adopted by Rodrigues et al. [5] and the same criteria for defining "site" was used by Alshukri et al. [4]. Sites and sub sites refer essentially to the same concepts, except that a sub site is part of another larger site / sub site. The criterion *b* is obviously the strongest. It is hard to picture a site that does not have some kind of global menu that is shared by all pages. From the usability perspective we can argue that global menus are necessary to avoid navigational volatility leading to disorientation or at least forcing users to reorient [15]. From a technical

perspective we can argue that today global page templates are common and this includes global navigation templates. Thus we can say that two pages that belong to the same site share at least one menu.

Figure 9. Different styles within the same site.

On the other hand, we can argue that two pages that share a menu always belong to the same site. The common style criterion is not decisive as Figure 9 shows. The visual design of the two pages from the MSDN website[4] is clearly different while they share the same global menu. Also an entry page can always be found. It often does not have any special characteristics, except being the first item in the shared menu.

Based on these considerations, the MenuMiner-approach can be used to detect site boundaries by identifying clusters of pages that share an s-menu.

6.2.2 Experiment and Results

After executing the SegmentCliqueFinder algorithm the s-menus represented by the extracted segment cliques are associated only with the pages that are linked by this menu. Obviously a menu is likely to be present on many more pages, at least, if it is some kind of main menu. For the finding of all pages that contain a certain s-menu, only those pages that link all pages of the s-menu clique have to be considered because the links are defined by the menu. To accurately identify an s-menu on a candidate page the method for measuring segment uniformity described in Sect. 5.2 can be used. This ensures that the segment is located in a similar position in the DOM tree and has similar style attributes. The aligned DOM tree paths of the s-menu segments are like signatures for menu templates which are very robust because of the wildcards. The obtained sets of pages that share an s-menu overlap because of the different scopes of, e.g., a global menu and a local menu. We retrieved a disjunctive set of 58 clusters by joining all overlapping sets. The joined sets are a precise presentation of the Site boundaries as our evaluation proves. We were not able to detect s-menus on 591 pages but we assume that this due to the incomplete crawl. We placed these pages in a single generic cluster.

Since we have observed that the URLs reflect the content organization of this domain very well, we were planning to compare our site segmentation to the folder structure. But it showed that even in this case the site boundaries cannot be derived from the URLs. So, we decided to conduct an additional manual evaluation.

However, we were faced with the problem of having to manually group 10,000 pages according to their site membership in a reasonable time. Even with a subset that is large enough to be representative, the task would be difficult for an assessor. It is much easier and less error-prone to present two pages to an assessor and ask him to decide if both belong to the same site or not. By this method we were able to have 845 randomly selected pairs compared by one person in approximately 10 hours. The person was given no further information about the purpose of the evaluation. We gave the hint that in case of uncertainty it should be checked if the linked homepages are identical. The assessor reported that he was able to decide in all cases without ambiguity.

The metric we used is based on the Rand index that measures the agreement of two partitions [16]. Let S be a set of elements and X_1 and X_2 two partitions of S. Let $C = \{(x_1, x_2) \in S \times S\}$ be the set of all tuples of elements in S^2. A is the set of agreements that contains the tuples of two elements that are either in the same cluster in both partitions or are in different clusters in both partitions. D is the set of disagreements. The Rand index measures the ratio of agreements:

$$R = \frac{A}{A+D}$$

Instead of considering all tuples we were using the random subset U of C from our evaluation for which the agreement set A_U and the disagreement set D_U are known as sample. For our partition of 58 clusters we computed a sample rand index R_U of 0.996. It is likely that the remaining error rate results from generic cluster and would be even lower for a complete crawl.

Three other clustering methods were used as benchmark. First we were analyzing the hierarchical structure of the URLs. The first level of the hierarchy was given by separating subdomains. The subdomains were split up again by the first folder and the resulting clusters again by the second folder. An R_U of 0.97 was computed for the segmentation by subdomains but only 9 clusters were extracted. This proves that subdomains contain multiple sites. By including the folder structure, the number of clusters increases but R_U drops to 0.95 and 0.87 (Figure 10).

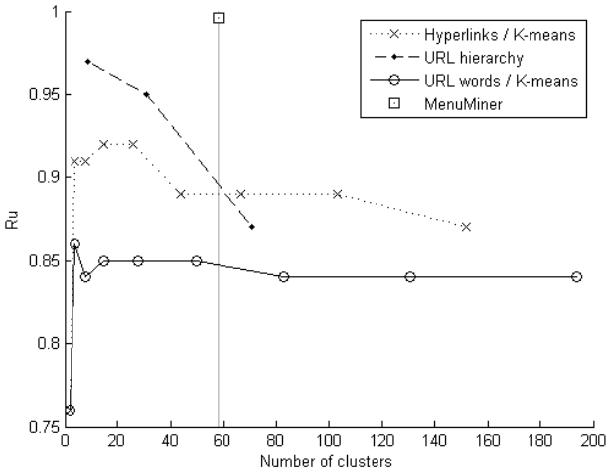

Figure 10: Rand index R_U based on manually evaluated samples for different segmentation methods

We were also implementing the methods for web site boundary detection that performed best in a recent experiment [4][5]. A bisecting k-means algorithm was applied iteratively to split clusters if they are large enough. Even though this method is not really an alternative to the MenuMiner algorithm because the number of clusters has to be known in advance, we wanted to see if a similar

[4] http://msdn.microsoft.com/en-us

[5] We did not include the combination of eight features that was evaluated in [4] too, because it did not perform significantly better than the best individual features in the reported experiments.

high value for R_U can be reproduced. As features the internal hyperlinks were used and the bag of words obtained by separating URLs with the delimiters "." and "/". The hyperlink feature performed better with a maximal R_U of 0.92 for 26 clusters compared to a maximal R_U of 0.86 for four clusters achieved by using the URL word feature. With these methods we were not able to reproduce an R_U as high as the one computed for the MenuMiner partition. The results showed the superiority of the proposed approach when applied to detect site boundaries.

But even more important, the almost perfect agreement with the human assessor relies on the very good performance of the MenuMiner-algorithm in detecting the main navigation systems of a site. This shows the reliability of the method in general.

6.3 Site Analysis

Without further analyzing the s-menus a basic hierarchical structure is already revealed by the relations of the clusters defined by the s-menus. If A and B are sets of pages that share an s-menu and $A \subset B$ while $A \neq B$, then A defines a subsection of a site. Figure 11 shows the discovered relations together with the sizes of the clusters. The connected components represent sites. One interesting observation is that sites that consist of few, smaller subsections are design-oriented sites focused on product presentation, e.g., the sites identified by the root clusters J (Microsoft Windows site), K (Windows Phone site) and M (Visual Studio site). Large clusters contain resources that cannot be accessed by a single hierarchical structure but can be so by facetted search such as the Pinpoint marketplace (A – cf. Table 1) and the template library of the Microsoft Office site (B). The small uniform nodes under the same root as A are compound documents [6] that consist of three sections (Figure 12).

These first observations are meant to illustrate the structural information gained with the proposed method based on clustering with s-menus. A further analysis of the data including the s-menu items promises a more fine-grained model of the information architecture.

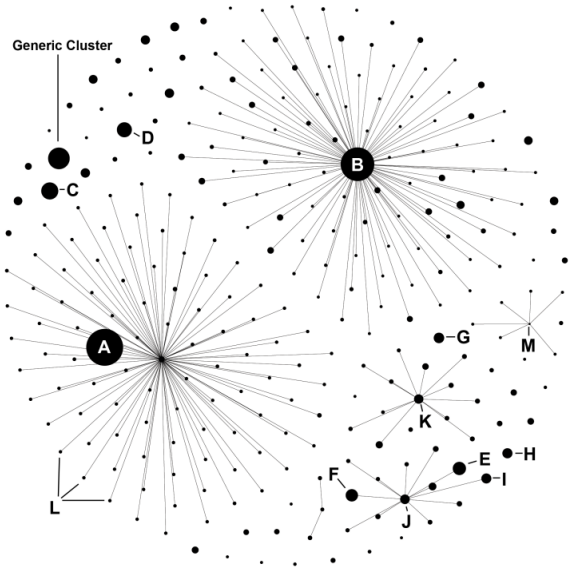

Figure 11. Clusters of sites and subsites

Figure 12. Three section of a compound document (L-nodes in Figure 11)

6.4 Main Menu Detection

Web site boundary detection is only a first application of our method that we have chosen as a demonstration and to evaluate the method. The s-menus delivered by the algorithm provide important information about the site's content organization and its menu structure. As a demonstration we have listed all extracted items of the main menu for the 10 largest clusters found in Table 1. Since the site clusters are defined by overlapping s-menus, the s-menu that is shared by the largest number of pages in each site can be considered as main navigation. If there were multiple s-menus that are shared by the same pages we were selecting one page and picked the s-menu that was first found when traversing the DOM tree in depth-first order. As Table I shows all main menus, and thus the main categories, were extracted correctly for the listed sites.

Table 1. Menu items of the then largest clusters

	#P.	Entry Page	Menu Items
A	4060	pinpoint.microsoft.com/en-US/applications/search?q=	Applications, Professional Services, Companies
B	2313	office.microsoft.com/en-us	home, products, support, images, templates, downloads, more
C	316	msdn.microsoft.com/en-us/default	Home, Library, Learn, Samples, Downloads, Support, Community, Forums
D	227	technet.microsoft.com/en-us/default	Home, Library, Wiki, Learn, Downloads, Support, Forum, Blogs
E	164	windows.microsoft.com/en-US/windows-vista/help	Windows Vista Help home, Top solutions, Using Windows Vista, Getting started, Community & forums, Contact support
F	137	windows.microsoft.com/en-us/windows7/help	Windows 7 Help home, Getting started, Top solutions, How-to videos, Community & forums, Contact support
G	89	msdn.microsoft.com/en-us/windows/aa904944.aspx	Home, Library, Learn, Downloads, Gallery, Support, Community, Forums
H	81	technet.microsoft.com/en-us/windowsserver/bb250589.aspx	Home, 2008, 2003, 2000, Library, Forums
I	79	windows.microsoft.com/en-US/windows7/products/home	Windows 7 home, What is Windows 7?, Compare, Features, Videos
J	70	http://windows.microsoft.com/en-US/windows/home	Home, Explore Windows, Products, Shop, Downloads, Help & How-to

7. RELATED WORK

Depending on the objectives two different research directions can be distinguished in the field of web structure mining. The first direction aims at generating *new* structures as rankings or topic hierarchies based on web documents and their structure. Algorithms such as PageRank or HITS and their variations belong to this direction as well as approaches that cluster web documents based on their content. The research presented in this paper belongs to a second research direction that aims at mining *existing*

structures that are difficult to retrieve as navigational hierarchies or boundaries of web sites.

Mining Navigation Elements

Some work has been done on mining navigation elements. Li and Kit describe an approach in [17] that is based solely on the web graph. Frequent item set mining algorithms are applied on the sets of outgoing hyperlinks of the pages to detect repeated menus. However a more comprehensive evaluation of the approach would be interesting. The work of Rodrigues et al. [5][8] on mining link blocks for representing sites and finding site boundaries has been described above. However the authors do not evaluate how well the page segmentation into link blocks really reproduces the navigation elements as perceived by humans. The ratio of linked text to all text is a common method for recognizing navigation elements or link lists [1][18] while in [19] other metrics as the text length and the hyperlink targets are used. The performance of these metrics is not reported. The work presented in [20] uses several other metrics to find navigation elements on page level. An inter-page analysis to find repeated navigation structures is not included. An evaluation was conducted showing a high recall and moderate precision. We used the link text ratio criterion in combination with other metrics in our previous work which included an inter-page analysis [7]. The evaluation showed that achieving a high accuracy with this approach is difficult.

There are some approaches that do not mine navigation elements explicitly but do take into account the structural information they provide. For instance, the clustering method described in [3] considers "parallel links" – links that are siblings in the DOM tree of a page. Such links are likely to belong to the same navigation element.

Clustering and Hierarchy Detection

Several approaches in this direction work solely on the web graph model, whose vertices represent pages and whose arcs represent hyperlinks. One of these is based on hyperlink co-citation for web clustering as introduced by Pitkow and Pirolli [21]. It is based on the idea that hyperlinks that frequently occur on a page together point to semantically related resources. Other approaches computed additional edge weights for the web graph in order to improve the clustering results. Extracting a hierarchical structure with standard graph algorithms based on the web graph is described in [1]. The edge weights are computed by machine learning methods that distinguish two link types based on eight link features. In [22] the edge weights for the web graph are computed based on text similarity and co-citation of hyperlinks. Three algorithms (k-means, multilevel METIS and Normalized Cut) are evaluated to partition up to 3500 documents. Normalized Cut performs best in the evaluation, but the objectives of the experiments are not the detection of Site boundaries but the clustering of topically related documents. Other web graph clustering methods consider hyperlink transitivity to compare pages that are not connected by a direct hyperlink (e.g. [23]).

An evaluation of the performance of four clustering algorithms in conjunction with several different features aside from the web graph is described in [4]. The features include word co-occurrences of the complete text as well as of the titles, hyperlinks, script-links and the URLs that are split into components using delimiters as "." and "/". A bisecting k-means algorithm on the URL components performs best. Instead of using a clustering algorithm, the site segmentation can be retrieved directly from the hierarchical structure of URLs. Using the hierarchical structure of URLs seems to be a very common approach (used e.g. in [1],[2],[24]) but it was not evaluated in [4]. However, it is well known that the hierarchical structure of URLs does not reflect the Site organization accurately [2].

An interesting approach of hierarchical web site segmentation is presented in [2]. The algorithm requires an existing tree structure on the resource, e.g., retrieved from the URL hierarchy and knowing the class (topic) of each resource. The tree is segmented into topically-cohesive regions, representing subsites. Also in the end a similar problem is addressed, the approach is very different from the work presented in this paper, which does not require a given classification and hierarchy.

Evaluation

The existing work on web pages shows that the evaluation of clustering methods for finding site boundaries and intra-site structures is a challenge. The main problem is that a reasonably large data set is necessary for meaningful results but the results have to be evaluated manually.

In [4] four sets of pages from different university departments are used, each representing a site and consisting of 500 pages. Thus the number of Sites is low and they are selected in advance what might bias the results. In [5] Rodrigues et al. describe an evaluation method that does not measure the aggregation of pages to sites but the precision and recall of detected entry pages. This allows considering sites as well as sub sites. They compare five methods, two of which are based on their own approach. Although the results are mixed and no method achieves a high F measure, the authors show that their approach is able to detect entry pages that are not found by other methods. In the experiments in [3] and [24] a large number of pages are clustered, but no metrics are used for evaluating the clustering quality. Instead the resulting clusters themselves are listed in tables and figures.

8. CONCLUSION

We believe that the MenuMiner-method proposed in this paper is a contribution that opens new doors for analyzing the structure of domains and sites. The algorithm is fast and its time-complexity is linear in the number of pages. It is solely based on analyzing the HTML structure and no additional resources such as CSS style sheets are required. A visual model is not necessary for identifying the s-menus of a page. The evaluation shows that the approach allows identifying with high precision the main menu systems that are a common characteristic of all pages of a site and that represent its central organization scheme. Applied to the problem of site boundary detection the presented approach provides almost perfect results in contrast to other current methods. The data obtained in the experiment also gave interesting information about the concepts of content access a site implements, based on which the site can be classified. In our experiment it also allowed the identification of compound documents.

The focus of the experiment and evaluation described was to show the reliability of the MenuMiner method. We found that the method is a very solid foundation that is ready to be applied in practice. Thus further research can be done on the interpretation and processing of the obtained data. S-menus can be considered as the structural skeleton of web sites. We believe that it is possible to retrieve the complete content hierarchy of web sites based on this skeleton with high precision. This would close the gap between the human perception of a site's content structure and the model generated by current structure mining methods. It would bring

improvements in many areas as e.g. the representation of search results, ranking, automated usability testing or web site reverse engineering.

9. REFERENCES

[1] C. C. Yang and N. Liu, "Web site topic-hierarchy generation based on link structure," *Journal of the American Society for Information Science and Technology*, vol. 60, no. 3, pp. 495-508, 2009.

[2] R. Kumar, K. Punera, and A. Tomkins, "Hierarchical topic segmentation of websites," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, Philadelphia, PA, USA, 2006, pp. 257-266.

[3] C. Lin, Y. Yu, J. Han, and B. Liu, "Hierarchical Web-Page Clustering via In-Page and Cross-Page Link Structures," in *Advances in Knowledge Discovery and Data Mining*, vol. 6119, Springer Berlin / Heidelberg, 2010, pp. 222-229.

[4] A. Alshukri, F. Coenen, and M. Zito, "Web-site boundary detection," in *Proceedings of the 10th industrial conference on Advances in data mining: applications and theoretical aspects*, Berlin, Germany, 2010, pp. 529-543.

[5] E. Mendes Rodrigues, N. Milic-Frayling, and B. Fortuna, "Detection of Web Subsites: Concepts, Algorithms, and Evaluation Issues," in *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Web Intelligence*, 2007, pp. 66-73.

[6] N. Eiron and K. S. McCurley, "Untangling compound documents on the web," in *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, Nottingham, UK, 2003, pp. 85-94.

[7] M. Keller and M. Nussbaumer, "Beyond the Web Graph: Mining the Information Architecture of the WWW with Navigation Structure Graphs," in *Proceedings of 2nd International Conference on Emerging Intelligent Data and Web Technologies (EIDWT 2011)*, Tirana, Albania.

[8] Eduarda Mendes Rodrigues, Natasa Milic-Frayling, Martin Hicks, and Gavin Smyth, "Link Structure Graphs for Representing and Analyzing Web Sites," Microsoft Research, Technical Report MSR-TR-2006-94, Jun. 2006.

[9] K. Makino and T. Uno, "New Algorithms for Enumerating All Maximal Cliques," in *Algorithm Theory - SWAT 2004*, vol. 3111, T. Hagerup and J. Katajainen, Eds. Springer Berlin / Heidelberg, 2004, pp. 260-272.

[10] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma, "Extracting content structure for web pages based on visual representation," in *Proceedings of the 5th Asia-Pacific web conference on Web technologies and applications*, Xian, China, 2003, pp. 406-417.

[11] G. Hattori, K. Hoashi, K. Matsumoto, and F. Sugaya, "Robust web page segmentation for mobile terminal using content-distances and page layout information," in *Proceedings of the 16th international conference on World Wide Web*, Banff, Alberta, Canada, 2007, pp. 361-370.

[12] "Document Object Model (DOM) Level 2 Core Specification, Version 1.0. W3C Recommendation 13 November, 2000." [Online]. Available: http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/.

[13] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Commun. ACM*, vol. 16, no. 9, pp. 575-577, 1973.

[14] J. Nielsen, "Subsite Structure (Alertbox)." [Online]. Available: http://www.useit.com/alertbox/9609.html. [Accessed: 24-May-2011].

[15] D. R. Danielson, "Transitional volatility in web navigation," *Information Technology and Society, 1(3),Special Issue on Web Navigation*, pp. 131–158, 2003.

[16] W. Rand, "Objective Criteria for the Evaluation of Clustering Methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846-850, 1971.

[17] C. Chui and C. Li, "Navigational Structure Mining for Usability Analysis," in *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05)*, 2005, pp. 126-131.

[18] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm, "DOM-based content extraction of HTML documents," in *Proceedings of the 12th international conference on World Wide Web*, Budapest, Hungary, 2003, pp. 207-214.

[19] J. Chen, B. Zhou, J. Shi, H. Zhang, and Q. Fengwu, "Function-based object model towards website adaptation," in *Proceedings of the 10th international conference on World Wide Web*, Hong Kong, Hong Kong, 2001, pp. 587-596.

[20] Z. Liu, W. K. Ng, and E.-P. Lim, "An Automated Algorithm for Extracting Website Skeleton," in *In Proceedings of the 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004), Jeju Island, Korea*, 2004, pp. 799–811.

[21] J. Pitkow and P. Pirolli, "Life, death, and lawfulness on the electronic frontier," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, Atlanta, Georgia, United States, 1997, pp. 383-390.

[22] X. He, H. Zha, C. H.Q. Ding, and H. D. Simon, "Web document clustering using hyperlink structures," *Computational Statistics & Data Analysis*, vol. 41, no. 1, pp. 19-45, Nov. 2002.

[23] J. Hou and Y. Zhang, "Utilizing hyperlink transitivity to improve web page clustering," in *Proceedings of the 14th Australasian database conference - Volume 17*, Adelaide, Australia, 2003, pp. 49-57.

[24] W. K. Cheung and Y. Sun, "Identifying a hierarchy of bipartite subgraphs for web site abstraction," *Web Intelli. and Agent Sys.*, vol. 5, no. 3, pp. 343-355, 2007.