Mining Taxonomies from Web Menus: Rule-Based Concepts and Algorithms

Matthias Keller and Hannes Hartenstein

Steinbuch Centre for Computing, Karlsruhe Institute of Technology, D-76128 Karlsruhe, Germany
{matthias.keller,hannes.hartenstein}@kit.edu

Abstract. The logical hierarchies of Web sites (i.e. Web site taxonomies) are obvious to humans, because humans can distinguish different menu levels and their relationships. But such accurate information about the logical structure is not yet available to machines. Many applications would benefit if Web site taxonomies could be mined from menus, but it was an almost unsolvable problem in the past. While a tag newly introduced in HTML5 and novel mining methods allow to distinguish menus from other contents today, it has not yet been researched, how the underlying taxonomies can be extracted, given the menus. In this paper we present the first detailed analysis of the problem and introduce rule-based concepts for addressing each identified sub problem. We report on a large-scale study on mining hierarchical menus of 350 randomly selected domains. Our methods allow extracting Web site taxonomy information that was not available before with high precision and high recall.

Keywords: Web site taxonomies, Web mining, Content hierarchies.

1 Introduction

What would a user do first to gain an overview over the information she can find on the ICWE2013 conference Web site? Most likely she will scan the prominently placed main menu at the top of the page. Maybe she is interested in the call for papers, so she would move the mouse pointer over the corresponding menu item to expand the child items. On the top level the contents of the site are organized hierarchically and the tree structure can be parsed unambiguously by users. This applies to most other Web sites as well.

Web site taxonomies, understood as logical hierarchies, are obvious to users, but not yet available to machines. Although nested lists can be modeled, HTML does not include language features that allow marking the different menu levels, e.g. the root menu of a Web site. There is a lot of previous work that focuses on extracting and generating different kinds of hierarchies based on Web content, e.g. from the hyperlink structure, URL structure or from text features. These hierarchies are very useful for many applications – but only because they approximate the real logical content organization. Humans, in contrast, are able to decode the logical organization from the menu structure and information architects, which are responsible for organizing

F. Daniel, P. Dolog, and Q. Li (Eds.): ICWE 2013, LNCS 7977, pp. 265–282, 2013.

Fig. 1. Integrating taxonomy information in the presentation of search results (mockup based on the search result presentation of google.com)[1]

and labeling information, express it in that way. Information architects emphasize the importance of well-designed taxonomies for usability (e.g. [1],[2]). If, based on menus, Web site taxonomies could be mined more accurately regarding human perception, all methods that rely on approximations of the logical hierarchy of Web sites would benefit, e.g. methods for automated sitemap generation [3], related entity finding [4], keyword enrichment [5] or Web site classification [6]. In addition, precise taxonomy information would allow whole new applications, in particular the integration of taxonomies in the presentation of search results (Fig. 1). The mockup illustrates that the first level of the Web site taxonomy provides useful information about the complete range of site content – information users cannot find in current search result summaries.

However, the seemingly simple problem of extracting taxonomies from menus appears to be a hard one at a closer look. Humans are able to decode visual features such as the position, size and layout of menus as well as color information for distinguishing different menu levels or menu types. These features carry semantics that are lost in the underlying markup code. But recent developments change the situation. The nav-tag introduced in HTML5 brings new possibilities for analyzing menus. Menus can be labeled as such and in turn machines are able to distinguish menus from other content. In addition, the MenuMiner-algorithm [7] presented recently allows identifying fixed menus independent from the underlying markup semantic.

In this paper we present the first thorough analysis of the problem of extracting logical taxonomies from hierarchical menus. We decompose the problem and present solutions for the different sub problems. We demonstrate that general design rules exist that allow solving each sub problem without analyzing visual features. In contrast to previous approaches, we focus on the real logical structure as perceived by humans. This requires an extensive evaluation, because no benchmark exists and the evaluation can only be conducted against the human perception.

The structure and the contributions of this paper are:

In Section 2 we define the central terms and *specify the problem statement* based on these definitions. In Section 3 we present the *first in-depth analysis of the problem* of extracting taxonomies based on menus. We identify three *sub problems that have not been described before* and propose *novel rule-based concepts* for solving each sub problem. We explain how the different levels within a menu can be distinguished, how active menu items can be determined and how Web site taxonomies can be assembled without analyzing visual features. Our concepts are generic because they rely

[1] Google provides shortcut links for some sites at a similar position that are based on ranking algorithms and that do not summarize the site content as the first level of a taxonomy does.

Fig. 2. Examples of different types of menus

on analyze the structure of the underlying HTML code only, but do not consider tag semantics. The methods consume little resources and do not require parsing Javascript or downloading presentational resources as images or CSS-files. In Section 4 we describe in more detail how the partial solutions are implemented and integrated into a single application. We introduce the novel *ListWalker-algorithm* that allows extracting taxonomies based on the order of the menu items only. In Section 5 we evaluate in a *large-scale study* how well the implementation performs. Correctness is evaluated against the human perception of the logical site structure. To our knowledge *no other taxonomy mining approach has been proven to extract the logical structure as perceived by humans with similar accuracy* regarding the identification of the first taxonomy level and, thus, the main content sections of sites. Finally, related work is presented in Section 6.

2 Problem Statement

In this paper the term *menu* denotes a user interface element of a Web site and its markup code. Menus have the single purpose of providing access to other resources (cf. Fig. 2). In other words, menus are implementations of navigation design patterns (e.g. [8]). A single menu can be part of multiple pages of a site. From the Web mining perspective the menu is defined by all its code snippets from all the pages it is presented on. The menu can occur in page-dependent variations with different *menu items* being expanded or collapsed (Fig. 2(B)). Thus, the code snippets of a single menu can differ. Different menu levels that are nested are considered as one menu if the underlying HTML code is continuous, e.g. in case of the menu shown in Fig. 2(B). In contrast, *a4* and *a5* in Fig. 2(A) are two separate menus.

In this paper the term *Web site taxonomy* describes a logical tree structure in which Web documents are arranged by information architects to facilitate access. Each node of the logical tree represents a document and has a label. Each node also represents a *site section* given by the subtree rooted at that node. Web site taxonomies are logical structures, not link structures. Taxonomies can also be distinguished from the design models representing them, e.g. whiteboard drawings, bullet lists or elaborated models part of Web engineering methods as WebML [9], OOHDM [10] or UWE [11]. To serve the purpose of facilitating human access, Web site taxonomies must be obvious to users. In particular, users must be able to decode the original tree structure. Since

menus are the user interface elements that provide access to other resources, taxonomies are usually implemented as hierarchical menus. Humans are able to decode the tree structure based on visual and functional features unambiguously. For example, in Fig. 2(B) a visual feature is that the child nodes are indented. Functional features encoding hierarchical structures are, e.g., that a submenu is expanded when the mouse pointer is moved over the parent item (cf. Fig. 2(A)) or different sub trees are permanently expanded depending on the active page (cf. Fig. 2(B)). Multiple menus can have the same underlying taxonomy, e.g. when there is a main menu at the top and a second level menu at the left side of a page.

This paper addresses the problem of automatically retrieving the underlying taxonomies from Web site menus. Taxonomies are understood as the logical organization as it is perceived by humans. Because of the nav-tag newly introduced in HTML5 and novel mining methods [7] the menus itself can be separated from other content. It is also assumed that if a single menu appears on multiple pages, all of its code snippets can be identified as belonging together.

The mining method should be correct in such a way that the delivered taxonomies match human perception. It should be universal and not be limited to specific menu implementations.

In addition, a viable solution should fulfill the following requirements:

- To enable efficient execution, parsing or interpreting Javascript code should not be necessary. For the same reason HTML rendering or downloading additional resources such as CSS files or images should not be required as well.
- The method should not rely on the way the menus are implemented in HTML, e.g. whether lists or span-tags are used to model menu items.
- The method should be fault tolerant. Input snippets that do not represent hierarchical menus but other page elements, e.g. breadcrumbs, should not lead to incorrect results.

3 Decomposing the Problem

In this section we describe how the problem extracting Web site taxonomies from menus can be broken down into sub problems that can be solved without analyzing visual features, rendering HTML code or executing Javascript code.

Multiple levels of a Web site taxonomy can be implemented by a single menu. Thus, extracting intra-menu hierarchies is the first sub-problem. Pages can be logically arranged under a menu item, even if they are not linked from within a sub menu of that item. Assigning pages to menu items is the second sub problem. Because different levels of a single taxonomy can be implemented by separate menus, extracting inter-menu hierarchies is the third sub problem.

3.1 Intra-menu Hierarchies

Sub Problem: *Extracting taxonomy information from individual menus*. Single menus often represent multiple levels of the Web site taxonomy. Humans can decode the

different levels by visual or functional features, but machines cannot interpret these features. The underlying HTML code does not reliably reflect the taxonomy. For example, Fig. 3 shows the logical structure of the menu snippet *a1* from Fig. 4. In the Web site taxonomy the node *p1* is a parent node of *p2*, but in the structure of the underlying HTML code *p1* and *p2* are siblings.

Solution: Menus that represent more than one level can be divided into two classes: (1) The first class are client menus for which the server always returns the same markup code on all pages. These menus only have a single snippet variant. Sub items are collapsed or expanded on the client-side via Javascript and CSS. (2) The second class are server menus with varying menu snippets depending on the active page (e.g. Fig. 2(B), Fig. 4). The menu state is generated on the server-side and not dynamically changed on the client side.

Client menus: In practice client menus are easier to deal with, since the original hierarchy can be derived by parsing the HTML tree using a simple page segmentation algorithm described in [7][2]. One reason is that usually each menu level has its own container element to switch it on or off. Additionally, since only client-side manipulation of the menu is involved, the logical structure of the menu must be available on client-side. Moreover, instead of proprietary scripts usually Javascript frameworks are used which render menus as nested lists as a kind of standard.

Server menus: Server menus display only fragments of the Web site taxonomy on each page. Fig. 3, for example, shows the logical structure of the menu snippets *a1* and *b1* from Fig. 4. Both menus use different patterns to generate the displayed taxonomy fragment: For the first menu all ancestors are rendered but not their siblings. In the second menu, the siblings of the ancestor p1 are also visible. In case of server menus the logical structure is often not preserved in the underlying HTML code. We found that instead of analyzing the HTML structure, other information can be utilized: There are general design rules almost all menus adhere to. They result from the three basic questions Web navigation has to answer: *"Where am I?"*, *"What's here?"* and *"Where can I go next?"* [2]. To indicate the current location and to allow users to navigate back to previous levels, the ancestors of the current node are always visible,

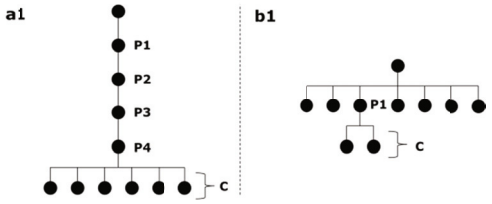


Fig. 3. Logical structure of the Web site taxonomy fragments represented by snippet *a1* and snippet *b1*of Fig. 4.

[2] A minor refinement of the algorithm described in [7] was made: ul-elements were always kept as containers and li-elements were always stripped.

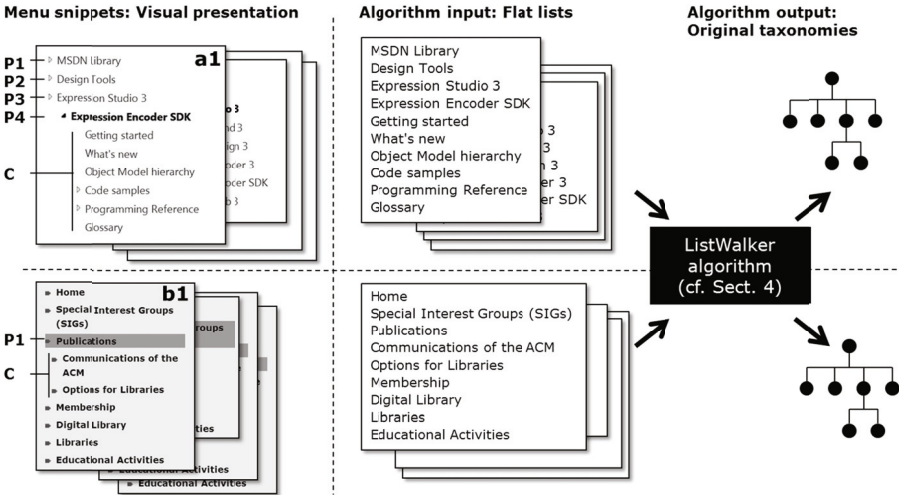
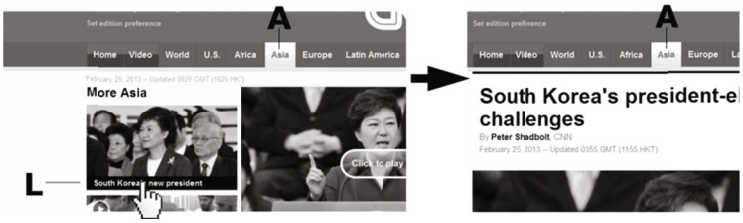


Fig. 4. The ListWalker algorithm processes the menu snippets as flat lists

even if the siblings of the ancestors are collapsed (cf. P1-P4 in snippet a1, Fig. 4). The ancestors appear in their logical order. Another design rule is that child nodes of the active node, if any, are always expanded to answer the question "Where can I go next?". All ancestors are usually presented prior to the child nodes. We found that given these general design rules the taxonomy can be extracted by processing all snippets of a menu as flat lists (Fig. 4), regardless depth or whether parent levels are expanded or not. The ListWalker-algorithm, presented in Section 4.2, extracts the taxonomy solely from the menu items that the snippets contain and their order. The underlying HTML structure and the semantics of the tags are ignored. The solution is very generic and can be applied to menus, regardless of whether tables, lists or other HTML elements are used.

3.2 Page Assignment

Sub Problem: *Assigning pages to site sections.* Each menu item corresponds to a site section, given by all the pages that are logically arranged under that item (at least the page linked by the menu item). There are often pages that are logically arranged under a certain menu item, even if they are not linked from within a sub menu of that item. For example, in Fig. 5 the content link *L* in the section "Asia" links a page that is clearly part of this section from the information architecture point of view. In such cases the parent-child relationship cannot be derived from intra-menu hierarchy information (cf. previous section) because there is no link to the child in the menu. In contrast to machines, humans are able to interpret complex visual features (e.g. that the item "Asia" is highlighted). Even if the item "Asia" would not be highlighted on the right-hand page, humans would attribute the page to this menu item, because of the link context and semantic knowledge.



Source: http://edition.cnn.com

Fig. 5. The page on the right is not linked by a menu item but by a link in the content area (L) only. It is still a child page of the active menu item "Asia" (A).

Solution: If the intra-menu structure is known, features can be extracted that allow to distinguish whether a page is arranged under a certain menu item or not. For example, in Fig. 5 the menu item "Asia" defines a site section and the linked page is part of this section. In order to extract distinguishing features for identifying other pages of this section, it is necessary to find examples of pages that are not part of this section. In general, pages linked by random other menu items cannot be used as negative examples for a section, because the menu items could as well be child nodes of the item defining the section. But if the intra-menu hierarchy is known, child nodes can be identified and excluded. For instance, the menu in Fig. 5 represents only a single level of the taxonomy and, based on this knowledge, all pages linked by other menu items can be used a negative examples for the section "Asia". The features we consider are (1) CSS classes that are assigned to the menu items and (2) URL directories of the linked pages.

(1) By analyzing the CSS classes, in many cases the active menu items can be detected without considering the actual visual presentation. Usually there are certain CSS classes that are used to highlight menu items and our method aims at determining these. Fig. 6 illustrates this approach. For example, the method assumes that the menu item "Europe" is highlighted on the page "Europe" but not on any other page linked in the same menu. If there are one or more CSS classes that are assigned to the menu item "Europe" if that page is active and, at the same time, these classes are not assigned to that menu item on any other page linked in the menu, it can be derived that these classes mark the item as active. Thus, if there are other pages of the site on which the same classes are assigned to the menu item "Europe", it can be concluded that these are child pages of the item as well.

(2) The second feature that can be used for page assignment is the hierarchical URL structure. Often the pages belonging to a site section, i.e. the child pages of a certain menu item, reside under the same directory. While the directory structure of a Web site may or may not reflect the logical structure, aligning it with menu items allows determining whether this is the case. Similar to the CSS feature it can be analyzed for all menu items whether they point to a directory that differs from the directory the other menu items point to. If menu items have child nodes, they can be considered additionally. Fig. 7 illustrates this approach. The URLs of the child nodes of the menu item "Audi Sport" have a common directory prefix that is exclusive in a way that no other pages linked by other menu items reside under this directory. All other pages of

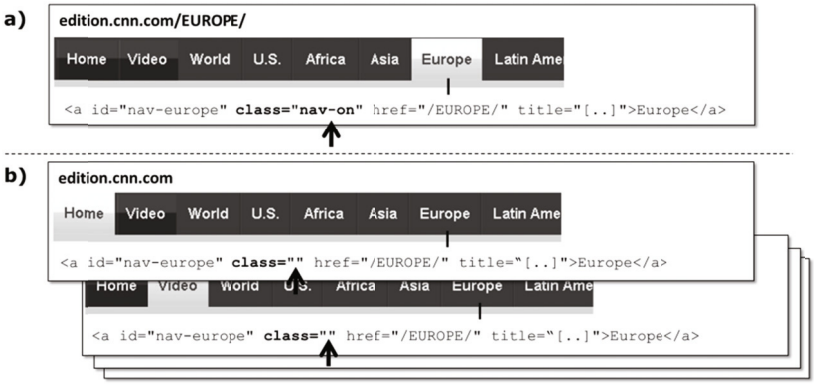


Fig. 6. When the linked page is active, the menu item "Europe" has a CSS class that is missing when another menu item is active, indicating that the class "nav-on" is used to mark the active menu item[3]

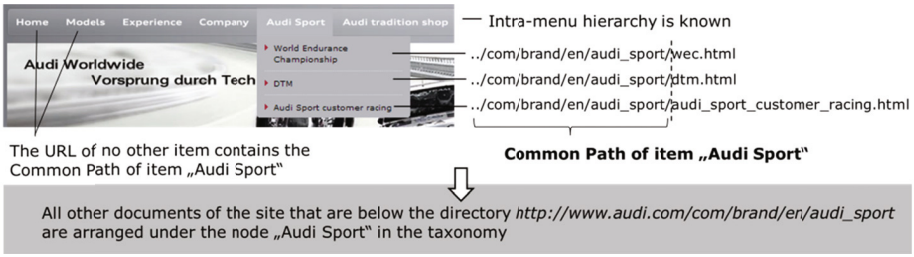


Fig. 7. The child nodes have a common directory and in turn all pages residing under this directory can be interpreted as child nodes of the menu item (source: http://www.audi.com)

the site that are not linked in the menu and are located below this directory can now be assigned as child nodes to the menu item "Audi Sport" as well.

3.3 Inter-menu Hierarchies

Sub Problem: Different levels of a Web site taxonomy are often implemented in separate menus. For example, a horizontal menu bar at the top represents the first level and a separate vertical menu at the left side represents the second level. These relations must be extracted in order to recover the original taxonomy. The menus representing taxonomy levels must also be distinguished from other menus, e.g. menus providing contextual links or navigation aids.

Solution: Menu-submenu relationships can be discovered based on the presented menu items. Fig. 8 illustrates the idea. If the items of a parent menu are traversed, the items of the child menu change with each page transition. The child menu will never have the same items for different active parents if a taxonomy is the underlying

[3] The CSS classes used to mark active items are often not assigned to the links, i.e. a-elements, but their parents (e.g. li-elements). This has to be considered in the implementation.

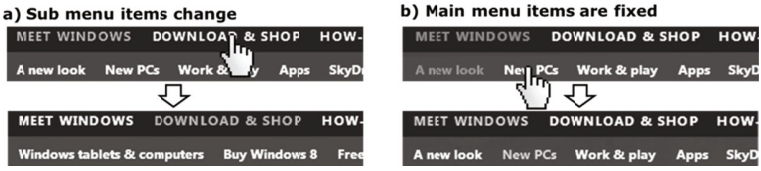


Fig. 8. The menu items of the lower menu change when the upper menu is traversed, while the upper menu is fixed if the lower menu is traversed. Thus, a menu-submenu relationship can be derived without analyzing presentational features (source: http://windows.microsoft.com).

structure. If, in contrast, the items of a sub menu are traversed, the menu items of the parent menu will not change. Despite promising preliminary results, the solution that is evaluated in Section 4 does not include this kind of analysis. In our evaluation runs we found that the task of discovering menu-submenu relationships is more challenging than we expected due to noise and irregularities. However, training a classifier on a large sample would be a solution but this is beyond the scope of this paper.

For distinguishing menus that represent levels of the taxonomy we first extract the intra-menu hierarchies and then try to arrange additional pages in this tree by applying the concepts presented in the previous section. This is done for all menus without testing in advance whether they really represent taxonomies or not. In the end, it is examined whether tree structures have been extracted, and if so, the most extensive is returned as Web site taxonomy (cf. Sect. 4).

4 Implementation

In this section we describe the implementation of the presented concepts in detail. We first describe how the concepts are integrated into a single solution based on the MenuMiner-algorithm [7]. Then, we introduce the ListWalker-algorithm that allows extracting intra-menu hierarchies based on flat lists.

4.1 Solution Overview

We implemented the rule-based methods described in Section 3 on top of the MenuMiner-algorithm [7] that delivers the boundaries of menus that are repeated on multiple pages (Fig. 9). In the example, two menus are found (Fig. 9(2)). Two variations

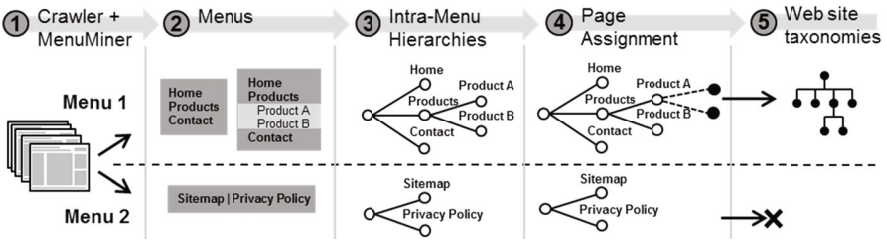


Fig. 9. (1) The MenuMiner-algorithm is used to extract menus. (2) By parsing the HTML structure and applying the ListWalker-algorithm intra-menu hierarchies are extracted. (3) The page assignment methods are applied to extend the hierarchies. (4) Heuristics are used to determine Web site taxonomies.

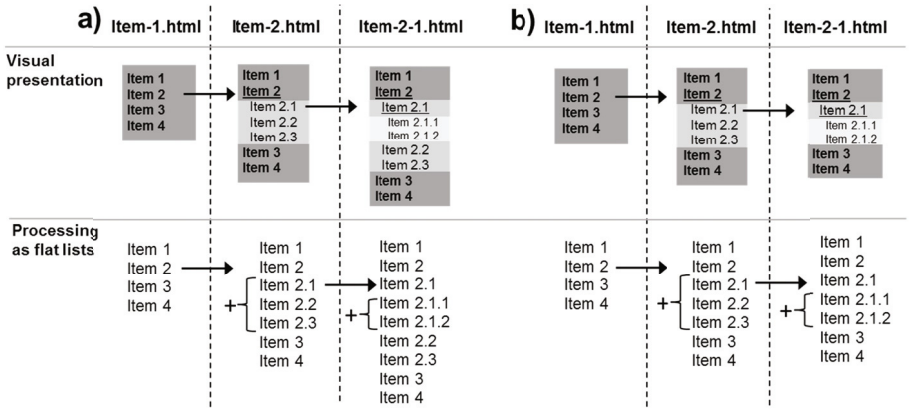


Fig. 10. On navigation paths descending the Web site taxonomy the child nodes are successively expanded. Thus, if the root state is known, the taxonomy can be extracted based on flat lists.

("menu states") of menu 1 are found on the site, one variant in which the child nodes of "Products" are collapsed and another one in which the child nodes are expanded. In the example, Menu 2 has only a single state. Instead of the MenuMiner-algorithm, the nav-tag introduced in HTML5 can be used for retrieving the menus of a site if it is widely applied in the future, because the methods can deal with noise, e.g. other navigation elements, such as breadcrumbs or paginations that not represent levels of the Web site taxonomy. Those elements will not deliver hierarchies and are ignored.

In the next step, HTML tree parsing for analyzing client menus and the ListWalker-algorithm (Section 4.2) for parsing server menus are applied (Fig. 9(3)). Both methods are used for each menu and if they both deliver hierarchy information, the most complete tree structure is kept for further processing. The overlap coefficient (cf. [12]) is used for handling noise, e.g. an additional hyperlink on one page that is missing on others, and deciding whether two slightly different snippets are considered as one and the same.

Then, the page assignment methods are applied (Fig. 9 (4)) and heuristics are used to determine the global taxonomy (Fig. 9 (5)). The implementation presented in this paper relies on first trying to discover hierarchical structures for all menus found individually and then judging which menu represents most likely the first level of the global hierarchy using a metric and other criteria listed below. If there are other pages that do not contain the menu, the menu selection process is repeated in order to find all taxonomies of the domain. In our evaluation runs we experimented with different ways of computing the selection metric K_i and found that a simple heuristic works best: $K_i = A_i / P_i$. The metric A_i is the average depth of the pages in the hierarchy of menu i, including pages that are arranged in the hierarchy by page assignment. If the page is neither part of the hierarchy nor an active menu item can be found, the depth is considered to be 0. The other factor influencing the detection of the main menu is the average position of the menu in the source code. If, for example, menu i is always the first menu in the source code of all pages it is contained, P_i has the value 1. In addition, we discard menus if one of the following boundary conditions is not met: (A) The menu has less than 15 items, (B) not more than 30% of the text content

appears in average before the menu (to exclude footer menus), and (C) no other menu with a lower average position P_i appears on more pages.

4.2 ListWalker-Algorithm

As argued in Section 3.1 the underlying HTML code does usually not reflect the logical structure of the taxonomy fragments rendered on different pages in case of server menus. However, there are two general design rules: (1) All ancestors of the active page are expanded and displayed above the active page (to allow ascending to parent levels). The ancestors appear in their logical order. Sibling of the ancestors may or may not be expanded. (2) If the active page has child nodes theses are always expanded (to allow further descending the hierarchy). The child nodes appear below the active page.

Algorithm 1. ListWalker

Input: $W = \{1,2, \ldots, m\}$ – the m pages of the site
$N_i = \{n_{i,1}, n_{i,2} \ldots n_{i,l_i}\} \subseteq W, i \in W$ – the l_i Menu Items of Page i

Output: G – the edges of the hierarchy

```
1.  // init
2.  G := { }; R := { }
3.  for each (N_i with n_{i,1} = i)
4.      for each (n_{i,j} ∈ N_i with j > 1)
5.          R := R ∪ (N_i, N_j)
6.  // main loop
7.  while (|R| > 0)
8.      Randomly select (N_i, N_j) ∈ R;
9.      R := R \ (N_i, N_j);
10.     k := 0;
11.     while (k < |N_i|)
12.         If ((k=0 OR n_{j,k-1}=j ) AND (notContains(N_i,n_{j,k})))
13.             break;
14.         k := k + 1
15.     while (k < |N_j| AND notContains(N_i, n_{j,k}))
16.         G := G ∪ (j, n_{j,k})
17.         R := R ∪ (N_j, n_{n_{j,k}})
18.         k := k + 1
```

The ListWalker algorithm presented in this section allows extracting the hierarchy by processing the menu snippets ("menu states") as flat lists based on these design rules. Fig. 10 illustrates the fundamental approach. The figure shows the menu behavior when a user descends the taxonomy starting from the root. Example *a* shows an implementation in which the parent levels, the current level and the children of the active item are expanded. In example *b* the intermediate level is collapsed. The child nodes can be easily derived based on the flat list representation because they are successively expanded. For example, the child nodes of Item-2 in example *a* can be

retrieved taking the menu items of page *Item-2.html* and subtracting the items of the previous state (state of *Item-1.html*), leaving Item 2.1, Item 2.2 and Item 2.3. This approach works for example *b* as well. Child nodes cannot be retrieved by subtracting the items of random states. Instead a valid reference state is necessary that is either a parent or sibling. If, e.g., in example *b* the menu state of *Item-2-1.html* would be used as reference to compute the children of *Item-2.html*, Item 2.2 and Item 2.3 would be wrongly assigned as children to that page. The reference state problem can be narrowed down to finding a root page that has no parent. The root page is a reference state for all its menu items. Thus, for each menu item the child nodes can be determined – and each menu item is again a reference state for its children and so on.

Algorithm 1 is the skeleton of the ListWalker algorithm. The pages W of the site are numbered from 1 to m. For page $w \in W$ the menu state is modeled as $N_w = \{n_{w,1}, n_{w,2}, \ldots\} \subseteq W$, the ordered list of pages linked in the menu. The algorithm computes G, the edges of the taxonomy as illustrated in Fig. 9. Because of the two design rules, there is usually a state with $n_{i,1} = i$, which is a state of a page that contains a link to itself at the first position (for example the states of *Item-1.html* in Fig. 10). Such a state will be referred to as first item state (FIS) in the following. The FIS can be determined easily and in case of a top level menu, the FIS is usually the homepage. Since a FIS belongs to the first level of the hierarchy and has no parent, it is always a root state and can be used as initial reference state for its siblings. There might be multiple FIS, because the extracted states of a menu can encompass multiple separated trees, e.g. sub sites in different languages.

In Algorithm 1, R holds tuples of reference states and unprocessed states. R is initially filled by iterating the menu items of the FISs, which are either children or siblings (lines 03-05). In the main loop a random tuple is taken from R (line 08) until R is empty. The loop starting at line 11 traverses the items of the menu state under examination until the first child page, which can be identified by being absent in the reference state. As additional condition the position is considered. Child pages are usually placed directly after their parent, but we also found implementations where a sub menu is positioned above or below the parent menu (line 12). The loop starting at line 15 iterates the child pages until a menu item is reached that is contained in the reference state, denoting the end of the sub menu. The edges from the parent to the child pages are added to G (line 16) and the active state is added as reference state for each child page to R for further processing.

In addition to the basic algorithm, a few extensions must be included to make it applicable for real world Web sites:

- Algorithm 1 delivers valid reference states recursively for all states except the FISs. As listed, the algorithm does not discover their child pages. A reference state for a FIS can be found by searching for extracted menu states whose items are a subset of the FIS.
- Real world menus often contain items that are redirections to pages that are placed somewhere else in the tree. If not considering these crosslinks, wrong reference states will result in faulty edges. Thus a tuple (N_j, N_d) with $d = n_{j,k}$ that represent a crosslink must not be added to R in line 17. Crosslinks can be

identified by testing if $\{n_{d,1}, n_{d,2} \ldots d\} \subseteq \{n_{j,1}, n_{j,2} \ldots d\}$ is not fulfilled. That is, the predecessors of the active item must also be predecessors of that item in the reference state.

5 Evaluation

The MenuMiner-algorithm scales very well [7] and the methods presented in this paper are little resource consuming. The operations including analyzing the crawled pages have linear-time complexity. In relation to the number of pages the number of menus grows sublinear and thus processing the menus is uncritical. Since runtime performance is not an issue and due to space limitations we focus on evaluating the correctness.

5.1 Methodology

For evaluating the method a data set was constructed by crawling 350 domains. The domain list was the result of a first crawl seeded with yahoo.com. The crawler was configured to discover new domains in a depth-first manner. In order to spread the samples, the next 25 discovered domains were skipped each time a domain was added to the list. Finally the 350 domains were crawled separately. A crawl was stopped if all pages were retrieved or 100 pages were processed by the MenuMiner algorithm, which means that all linked pages were downloaded, too. All in all 259,525 pages were crawled. One page from each domain was randomly selected and the main menu, the active menu item, the second menu level and its active item were labeled if existing.[4] In this paper the term main menu refers to a menu that implements the first level of the Web site taxonomy. We labeled menus as main menus if

- The menu is indispensable for site navigation
- The menu items represent the main content sections
- At least three menu items are links to pages of the same site
- No other menu fulfills the previous conditions

By using these conditions for most of the samples either the main menu could be identified clearly or the absence of a main menu could be determined. However, some samples could not be labeled with reasonably certainty and were excluded from the evaluation. These were pages in languages with non-latin alphabets and pages that seem to contain multiple main menus according to the above definitions. The active menu item was labeled as well and if existing, the second level and the active second level menu item were labeled as well. The active menu items were tagged not only if highlighted, but also if they could be determined otherwise, e.g. by an additional breadcrumb navigation. Similar to the main menu, the second level menus that could not be labeled with certainty were excluded (2.9% of the samples).

[4] The labeled data set is available from:
`http://dsn.tm.kit.edu/download/icwe2013/data.zip`

The method was evaluated as a binary classification task. A positive classification means that the method delivers a menu or a menu item respectively. For evaluating the correctness the URLs were compared. If all menu items of a mined menu are contained in the labeled menu, it is considered as true positive (TP), otherwise as false positive (FP). On the other hand, additional menu items in the labeled menu are allowed, because the mined menu items represent the global menu while additional items may appear on certain pages. However, for 76.4% of the samples, the number of labeled items equals the number of mined items, and for the other samples, on average 74.6% of the items were mined. If no menu or menu item respectively was mined and if none was labeled it counts as true negative (TN), otherwise as false negative (FN).We were evaluating Precision as TP/(TP+FP) and Recall as TP/(TP+FN) for the task of detecting the main menu, the active main menu item, the secondary menu and the active secondary menu item (Table 1). In the evaluation of correctness of the active menu items only pages were included for which the menu itself was detected correctly. Four different configurations were evaluated: Configuration A using CSS selectors for page assignment (cf. Section 3.1), configuration B using URL prefixes, configuration AB using both methods and A_{res}, a more restrictive version of A, delivering only menus if they contain a submenu. Similarly, only the TPs from the main menu detection were included in the evaluation of the secondary menu.

5.2 Results

Fig. 11 and Table 1 show that the configurations A, B and AB detect the first Taxonomy level with a Precision around 0.9 and Recall around 0.75. Method A is a very accurate solution for detecting active menu items, with a Precision close to 0.97. Because the active menu items indicate different site sections, the method delivers precise topical segmentations of sites that were not available previously. Method B has a higher Recall but reduced Precision. The combined method performs well with a Precision of 0.89 and Recall of 0.8. For detecting the secondary menu, the configurations succeed with good Precision values above 0.85. Menu items that are no hyperlinks seem to be the main reason for errors here. However, Recall is low, because only secondary menus that are nested within the main menu can be found yet.

We believe that Precision is fundamental for the applicability of a hierarchy mining method in most scenarios. Since no comparable methods exist, even a low Recall is an improvement. The results show that there is room for increasing Precision at the cost of Recall. Thus we implemented method A_{res} which only delivers global menus that contain a second level, based on the idea that if a nested secondary menu was found, the main menu is identified correctly with high probability. As expected, Recall is significantly reduced, but Precision is almost perfect.

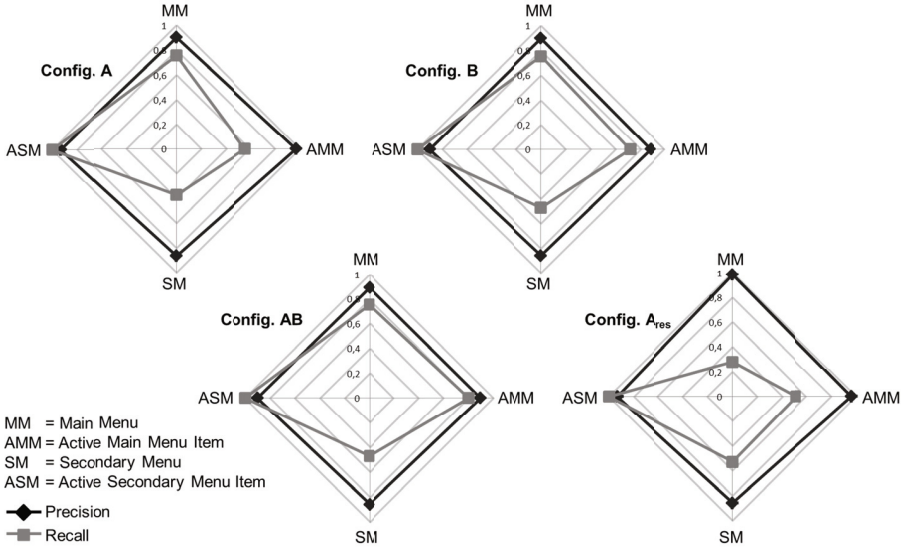


Fig. 11. Precision and Recall of the evaluated configurations

Table 1.

	Config. A	Config. B	Config. AB	Con. A_{res}
Main Menu				
Precision	0.903	0.898	0.893	0.986
Recall	0.756	0.755	0.754	0.278
TP/FP/TN/FN	177/19/44/57	176/20/44/57	175/21/44/57	70/1/44/182
Active Main Menu Item				
Precision	0.968	0.882	0.894	0.96
Recall	0.555	0.714	0.8	0.511
TP/FP/TN/FN	61/2/65/49	75/10/61/30	84/10/60/21	24/1/22/23
Secondary Menu				
Precision	0.864	0.857	0.857	0.857
Recall	0.373	0.471	0.471	0.529
TP/FP/TN/FN	19/3/123/32	24/4/121/27	24/4/120/27	18/3/33/16
Active Secondary Menu Item				
Precision	0.933	0.9	0.9	0.933
Recall	1	1	1	1
TP/FP/TN/FN	14/1/4/0	18/2/4/0	18/2/4/0	14/1/3/0

6 Related Work

In this paper, we use the term taxonomy mining to denote the process of extracting hierarchies that are pre-designed by information architects and that are obvious to humans because of the visual presentation. Thus, works that generate new hierarchies based on textual or structural information (e.g. [13]) are not considered in this section as well as works on extracting non-hierarchical structures (e.g. [14]) or recovering application models (e.g. [15]). Yang et al. [16] state that the extraction of Web site

hierarchies is a very new research topic. Although it is known, that the underlying content hierarchy can only be approximated by analyzing the structure of URLs [17][4], it is the common method (e.g. used in [17],[16],[18],[5]) up until now due to a lack of alternatives. An advanced method of learning hierarchies from URLs and query strings is described in [5]. Despite the limitation of current approaches for Web site hierarchy mining, many fields of application have been proposed, e.g. related entity finding [4], Web site classification [6], topic segmentation [17] or improving recommendation models [19]. A recent paper shows that contextual advertising can benefit from a keyword enrichment method based on Web site taxonomies [5]. Bose et al. [19] obtain the hierarchy information directly from the content designer or the content management system, the other methods include URL-based heuristics. Only Yang et al. [16] describe an isolated evaluation of the quality of the hierarchy information. They extract hierarchies from the Web graph but utilize the hierarchical URL structure to generate edge weights. Their methods perform well in the conducted evaluation that includes five sites. However, the evaluated sites seem to be examples where the URL structure is a good model for the underlying content hierarchy.

The nav-tag introduced in HTML5 allows semantically annotating menus and our approach presented in [7] delivers menus regardless of the used tags. Previously, discerning menus from other content was an unsolved problem. Few other works included the task of mining menus or link lists, but consider it as side issued not evaluated separately. An exception is the method described in [20] that discovers ranked lists of menus and navigation aids (“key information”). The Precision seems to be not very high, since in an evaluation of five sites only for two sites the best ranked block really contains key information. Liu et al. [21] report an average Precision of 0.92 for detection of “navigation link sets” on five news sites but it is not clear if these are equivalent to the main navigation. However that is a very good result compared to our experience with using this kind of features [22].

7 Conclusion

In this paper we demonstrated that Web site taxonomies can be accurately mined based on menus. Our rule-based approach succeeds without resource-consuming HTML rendering and is not bound to the semantics of specific tags. We evaluated the method on 350 randomly selected real-world Web sites. The method was able to detect the first level of the Web site taxonomy correctly with Precision above 0.9 at high Recall and the secondary menu with Precision above 0.86. Page assignment is possible with Precision around 0.97. Thus, the method delivers a very accurate topical segmentation. Such precise information about the logical organization of Web sites was not available before and cannot be extracted from the Web graph or the URL structure. One tested configuration delivers the first menu level with almost perfect precision, but, currently, at the expense of reduced Recall.

However, the methods can be further adjusted based on the labeled data set generated for evaluation and we expect that perfect Precision with a much higher Recall is

possible in the future. The combination of separated menus into a single taxonomy was beyond the scope of this paper and needs more research.

Web site taxonomy information has been used in many fields and thus, many applications can benefit from the more accurate mining methods presented in this paper. Also, the extracted taxonomy information can be used for whole new applications, e.g. to enhance the presentation of search results.

References

1. Morville, P., Rosenfeld, L.: Information architecture for the World Wide Web. O'Reilly, Sebastopol (2006)
2. Kalbach, J.: Designing Web navigation. O'Reilly, Sebastopol (2007)
3. Lin, S.-H., Chu, K.-P., Chiu, C.-M.: Automatic sitemaps generation: Exploring website structures using block extraction and hyperlink analysis. Expert Systems with Applications 38, 3944–3958 (2011)
4. Yang, Q., Jiang, P., Zhang, C., Niu, Z.: Reconstruct Logical Hierarchical Sitemap for Related Entity Finding. In: Voorhees, E.M., Buckland, L.P. (eds.) The Nineteenth Text Retrieval Conf (TREC 2010). National Institute of Standards and Technology, NIST (2010)
5. Pavan Kumar, G.M., Leela, K.P., Parsana, M., Garg, S.: Learning website hierarchies for keyword enrichment in contextual advertising. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 425–434. ACM, Hong Kong (2011)
6. Amitay, E., Carmel, D., Darlow, A., Lempel, R., Soffer, A.: The connectivity sonar: detecting site functionality by structural patterns. In: Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia, pp. 38–47. ACM, Nottingham (2003)
7. Keller, M., Nussbaumer, M.: MenuMiner: revealing the information architecture of large web sites by analyzing maximal cliques. In: Proceedings of the 21st Int'l. Conf. Companion on World Wide Web, pp. 1025–1034. ACM, Lyon (2012)
8. Rossi, G., Schwabe, D., Lyardet, O., Puc-rio, D.D.I., MarquêS, R., Vicente, S.: Improving Web information systems with navigational patterns. Computer Networks 31 (1999)
9. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): a modeling language for designing Web sites. Computer Networks 33, 137–157 (2000)
10. Schwabe, D., Rossi, G., Barbosa, S.D.J.: Systematic hypermedia application design with OOHDM. In: Proc. of the the Seventh ACM Conf. on Hypertext, pp. 116–128. ACM, Bethesda (1996)
11. Koch, N., Knapp, A., Zhang, G., Baumeister, H.: Uml-Based Web Engineering. In: Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.) Web Engineering: Modelling and Implementing Web Applications, pp. 157–191. Springer London, London (2008)
12. Jones, W.P., Furnas, G.W.: Pictures of relevance: a geometric analysis of similarity measures. J. Am. Soc. Inf. Sci. 38, 420–442 (1987)
13. Ho, Q., Eisenstein, J., Xing, E.P.: Document hierarchies from text and links. In: Proceedings of the 21st International Conference on World Wide Web, pp. 739–748. ACM, Lyon (2012)
14. Zheng, X., Gu, Y., Li, Y.: Data extraction from web pages based on structural-semantic entropy. In: Proc. of the 21st Int'l. Conf. Companion on World Wide Web, pp. 93–102. ACM, Lyon (2012)

15. Bernardi, M., Di Lucca, G., Distante, D.: The RE-UWA approach to recover user centered conceptual models from Web applications. International Journal on Software Tools for Technology Transfer 11, 485–501 (2009)
16. Yang, C.C., Liu, N.: Web site topic-hierarchy generation based on link structure. J. Am. Soc. Inf. Sci. Technol. 60, 495–508 (2009)
17. Kumar, R., Punera, K., Tomkins, A.: Hierarchical topic segmentation of websites. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 257–266. ACM, Philadelphia (2006)
18. Cheung, W.K., Sun, Y.: Identifying a hierarchy of bipartite subgraphs for web site abstraction. Web Intelli. and Agent Sys. 5, 343–355 (2007)
19. Bose, A., Beemanapalli, K., Srivastava, J., Sahar, S.: Incorporating concept hierarchies into usage mining based recommendations. In: Nasraoui, O., Spiliopoulou, M., Srivastava, J., Mobasher, B., Masand, B. (eds.) WebKDD 2006. LNCS (LNAI), vol. 4811, pp. 110–126. Springer, Heidelberg (2007)
20. Wang, C., Lu, J., Zhang, G.: Mining key information of web pages: A method and its application. Expert Syst. Appl. 33, 425–433 (2007)
21. Liu, Z., Ng, W.K., Lim, E.-P.: An Automated Algorithm for Extracting Website Skeleton. In: Lee, Y., Li, J., Whang, K.-Y., Lee, D. (eds.) DASFAA 2004. LNCS, vol. 2973, pp. 799–811. Springer, Heidelberg (2004)
22. Keller, M., Nussbaumer, M.: Beyond the Web Graph: Mining the Information Architecture of the WWW with Navigation Structure Graphs. In: Proc. of the 2011 Int'l. Conf. on Emerging Intelligent Data and Web Technologies, pp. 99–106. IEEE Computer Society, Tirana (2011)