# Understanding Site Structure by Reverse Engineering Web Navigation Elements

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

**genehmigte**

**Dissertation**

von

**Matthias Jens Keller**

aus Heilbronn

| | |
|---|---|
| Tag der mündlichen Prüfung: | 25. Juni 2014 |
| Erster Gutachter: | Prof. Dr. rer.nat. Hannes Hartenstein |
| | Karlsruher Institut für Technology (KIT) |
| Zweiter Gutachter: | Prof. Dr.-Ing. Martin Gaedke |
| | Technische Universität Chemnitz |

# Zusammenfassung

Die Informationsarchitektur der meisten Websites – also die logische Organisation und Darstellung der Inhalte – stützt sich auf eine hierarchische Gliederung. Diese Gliederung wird den Benutzern durch mehrstufige Menüstrukturen kommuniziert, d.h. Haupt- und Untermenüs unterteilen Websites aus Perspektive der Benutzer in eine logische Hierarchie. Menschen können diese Struktur aufgrund der grafischen Darstellung identifizieren, die es ihnen ermöglicht, Menüs von anderen Inhalten und Ober- von Untermenüs zu unterscheiden. HTML erlaubt zwar, mit semantischen Annotationen die interne Struktur einzelner Dokumente in maschinenlesbarer Form abzubilden, allerdings kann die logische Struktur ganzer Sites nicht mit Hilfe von HTML spezifiziert werden. Dabei ist die logische Struktur in der Regel klar definiert, nur eben nicht explizit, sondern visuell kodiert. In der Dissertation wird aufgezeigt, dass das Konzept der logischen Struktur, wie es im Bereich des User Interface Designs verwendet wird, im Bereich des Web Mining kaum aufgegriffen wurde und Methoden zur automatischen Extraktion logischer Hierarchien fehlen. Damit bleiben Metadaten ungenutzt, die etwa für die Bewertung und Darstellung von Suchergebnissen wertvoll sein können – und der Nutzen des WWW als Informationsquelle hängt wesentlich von der Qualität der Suchergebnisse und deren Darstellung ab. Eine präzisere Abbildung der Informationsarchitektur, wie sie von Menschen wahrgenommen wird, kann neue Crawling-Strategien ermöglichen, Ranking-Verfahren verbessern und eine übersichtlichere Darstellung der Suchergebnisse ermöglichen. Daher ist das Ziel dieser Arbeit, Lösungen für die automatische Extraktion logischer Hierarchien zu entwickeln und zu evaluieren.

Zunächst werden die grundlegenden Konzepte geklärt und Arbeitsdefinitionen wichtiger Begriffe entwickelt. Die besondere Herausforderung liegt im Umgang mit der Subjektivität und Unschärfe des Konzepts der von Benutzern wahrgenommenen logischen Struktur, die Gegenstand der vorliegenden Forschungsarbeit ist. Dazu wird das $O^3$-Modell eingeführt, das drei Aspekte der Organisation von Websites voneinander abgrenzt: Die inhaltliche Organisation („logische Struktur"), die funktionale Organisation (Hyperlinks) und die Seitenorganisation („Layout"). Die Analyse des Zusammenspiels der inhaltlichen und funktionalen Organisation bildet die Grundlage für die in der Arbeit vorgestellten Lösungsansätze. Als

Bindeglied zwischen beiden Aspekten werden Navigations-Entwurfsmuster identifiziert, die nun als Regeln für die Transformation der inhaltlichen Organisation in die funktionale Organisation aufgefasst werden. Das zugrundeliegende Navigations-Entwurfsmuster fassen wir als den Typ eines Navigationselements, d.h. Menüs, auf. Auf diesen Überlegungen aufbauend, werden vier Teilprobleme der automatischen Hierarchieextraktion identifiziert: (1) die Unterteilung von Seiten in inhaltliche Blöcke, (2) die Erkennung von Blöcken, die Navigationselemente darstellen, (3) das Erkennen der Instanzen gleicher Navigationselemente auf unterschiedlichen Seiten und (4) die Rekonstruktion der Hierarchie auf Basis der Navigationselemente.

Währende die Teilprobleme (3) und (4) in existierenden Arbeiten bisher kaum behandelt wurden, wurden Lösungen für die Teilprobleme (1) und (2) als Nebenaspekte einiger Arbeiten mit ganz unterschiedlichen Schwerpunkten bereits diskutiert. Allerdings zeigt die Evaluierung eines ersten Prototyps, dass konventionelle Methoden zu fehleranfällig sind und sich als Grundlage für die Hierarchie-Extraktion nur bedingt eignen. Dies führt zur Entwicklung des GRABEX-Ansatzes („**Gr**aph-based **b**lock **ex**traction“) für die Extraktion bestimmter Navigationselement-Typen. Der GRABEX-Ansatz unterscheidet sich durch zwei wesentliche Merkmale von konventionellen Methoden: (a) Alle Links, die zu den Instanzen eines bestimmten Navigationselements gehören, werden als Graph aufgefasst. Bestimmte Graph-Muster lassen auf den Typ des Navigationselements schließen und werden für die Klassifizierung verwendet. (b) Die Navigationselemente und ihre Instanzen werden nicht zuerst extrahiert und dann klassifiziert. Stattdessen wird angenommen, dass alle Seitenunterteilungen und alle Kombinationen von Blöcken potentielle Navigationselemente darstellen und dieser Raum wird durchsucht, bzw. die Klassifizierung wird auf dieser Menge durchgeführt. Wird ein charakteristisches Muster gefunden, kann auch auf eine korrekte Seitenunterteilung und Gruppierung von Navigationselement-Instanzen geschlossen werden.

Wir stellen die MenuMiner-Methode vor, eine Umsetzung des GRABEX-Ansatzes zur Extraktion hierarchischer Menüs. Charakteristische Graph-Muster sind vollständigen Teilgraphen (Cliquen). Zur Evaluierung werden die erkannten Menüs verwendet, um die Grenzen unterschiedlicher Websites zu identifizieren, die unter derselben Domain abgelegt sind. Im Gegensatz zu Lösungen, die in vorherigen Arbeiten beschrieben wurden, konnte die neue Methode die Sitebegrenzungen nahezu fehlerfrei bestimmen. Das MenuMiner-Verfahren ist die Grundlage für weitere Algorithmen, die darauf aufbauend die logische Hierarchie extrahieren. Zur Evaluierung wurde das Verfahren auf 350 Domains angewandt und die automatisch extrahierte hierarchische Struktur mit der zuvor manuell annotierten Hierarchie verglichen. Die Resultate zeigen, dass mit dieser Methode nun ein Verfahren verfügbar ist, das es erlaubt, z.B. die erste Menü-Ebene, welche die Hauptthemen einer Website repräsentiert, mit großer Genauigkeit zu bestimmen. Weiterhin wird eine zweite Umsetzung des GRABEX-Ansatzes beschrieben, BreadcrumbMiner, das erste veröffentlichte und praxistaugliche Verfahren zur Extraktion von Breadcrumbs. Breadcrumbs zeigen die Position einer Seite in der Hierarchie an (z.B. Home > Produkte > Elektronik) und stellen deshalb eine weitere Möglichkeit zur Rekonstruktion der logischen Struktur einer Site dar. BreadcrumbMiner kombiniert das GRABEX-Verfahren mit klassischen Maschine-Learning-Methoden.

Ein weiterer Beitrag der vorliegenden Arbeit ist ein empirisches Experiment, dass Aussagen darüber ermöglicht, wie stark die logische Struktur das Navigationsverhalten von Benutzern beeinflusst. Es konnte gezeigt werden, dass Benutzer, die von einer Suchmaschine zu einer Seite einer hierarchisch organisierten Website navigieren, im nächsten Schritt dazu tendieren, die Hierarchie weiter hinabzusteigen. Daraus folgern wir, dass die Integration von Hierarchie-Informationen in die Suchergebnisdarstellung es Benutzern ermöglichen würde, die nächsten möglichen Navigations-Schritte im Voraus zu bewerten.

# Abstract

The information architecture of most Web sites, i.e. the logical organization of the contents, is based on a hierarchical structure. This structure is communicated to the users through multi-level menu systems, i.e. main and submenus define a logical hierarchy from the perspective of the users. Humans are able to identify this structure based on the visual presentation, which allows them to distinguish menus from other contents as well as main menus from submenus. While HTML allows encoding the logical structure of individual pages, it does not include language features that allow specifying the logical structure of entire sites in a machine-readable way. At the same time, the logical organization of a site *is* clearly defined, although not explicitly but visually encoded. In this thesis, we show that the concept of logical organization as used in the field of user interface design did not yet play a major role in the area of Web mining. Hence, there is a lack of methods for automatically extracting the logical structure. As a results, valuable metadata are left unexploited that could be used, e.g. for improving Web search – which is a highly relevant research area because the benefit that we can gain from the information source WWW strongly depends on the functionality provided by search engines. Being able to extract the actual information architecture automatically with more accuracy than before enables more efficient crawling strategies, improved ranking methods and an enhanced arrangement of search results. Hence, this thesis aims at the development and evaluation of solutions for automatically extracting logical hierarchies.

First, the basic concepts are clarified and working definitions of important terms are developed. In this context, it is a particular challenge to find a way for dealing with the fact that the logical organization as understood in this thesis is based on human perception and hence a fuzzy concept. For that purpose, the O$^3$-model is introduced in which three aspects of the organization of a Web site are distinguished: The content organization (logical structure), the functional organization (hyperlink structure) and the page organization (layout). The solutions presented in this thesis are based on analyzing the relationship between the content organization and the functional organization. We identify navigation design patterns as link between both aspects, which we interpret as rules transforming the content organization into the functional organization. We will refer to the underlying navigation design pattern of a navigation element as its type. Based on these observations, four partial problems of automated hierarchy extraction are identified: (1) Segmenting pages into visual blocks, (2) recognizing blocks that represent navigation elements, (3) identifying all instances of the same navigation element on different pages, and (4) reverse engineering navigation elements in order to recover the content hierarchy.

There is not much related work that addresses the partial problems (3) and (4), but the partial problems (1) and (2) have been considered as side issues of some previous research on different topics. However, the evaluation of a first prototype indicates that conventional methods are prone to error and do not represent a suitable foundation for hierarchy extraction solutions. This leads to the development of the GRABEX-approach (**gr**aph-based **b**lock **ex**traction approach) for mining specific types of navigation elements. The GRABEX-approach has two basic characteristics that differ from previous approaches: (a) The links that belong to the instances of a particular navigation element are represented as graph. Specific graph patterns reveal the type of a navigation element and can be used for classification. (b) The navigation elements and their instances are *not* first extracted and afterwards classified. In contrast, it is assumed that all possible segmentations and all possible combinations of blocks represent potential navigation elements and this space is searched for characteristic patterns, i.e. classification is conducted on this set. If a characteristic pattern is found, it can be concluded that it results from correctly segmented and joined navigation element instances.

We present MenuMiner, an implementation of the GRABEX-approach for extracting hierarchical menus. Characteristic patterns that are mined are complete subgraphs (cliques). To evaluate the solution, the retrieved menus are used to detect the boundaries of sites that are hosted under the same domain. In contrast to previous solutions, the MenuMiner-based method solves this task almost errorless. For further processing the menus delivered by MenuMiner in order to recover the hierarchy, rule-based methods are implemented and evaluated on a manually labeled dataset of 350 domains. The results of the experiment demonstrate that the method allows extracting, e.g., the first hierarchy level which represents the main topics of a site with high accuracy – previous approaches were not able to extract such information. Furthermore, we describe a second implementation of the GRABEX approach, BreadcrumbMiner, the first published and applicable method for mining breadcrumbs. Breadcrumbs show the location of a page in the hierarchy (e.g., home > products > electronics) and, hence, are another way of recovering the content hierarchy of a site. BreadcrumbMiner combines the GRABEX-approach with common machine-learning methods.

Another contribution of this thesis is an empirical experiment that allows insights into the influence of hierarchies on human browsing behavior. We show that users that navigate from a search engine to a page of a hierarchically organized Web site tend to further descend the hierarchy in the next browsing step. This allows the conclusion that integrating hierarchy information into the presentation of search results would help users to anticipate the next browsing steps.

# Danksagung

Ich bedanke mich bei all jenen, die mich in den letzten Jahren unterstützt und begleitet haben. Ich habe von vielen fachlichen Ratschlägen, Anregungen und Diskussionen profitiert. Kollegen, Freunde und Familie gaben mir stets den Rückhalt und Motivation. Ihr habt dafür gesorgt, dass die Promotionszeit im Rückblick nicht nur eine arbeitsreiche sondern auch eine schöne und erfüllte Zeit war!

Besonders möchte ich mich bei den Personen bedanken, die meine wissenschaftliche Arbeit in den letzten Jahren betreut haben: bei meinem Doktorvater Prof. Dr. Hannes Hartenstein, der fachlich und im persönlichen Umgang stets ein großes Vorbild war; bei Dr. Martin Nussbaumer, der mir in gemeinsamen Diskussionen geholfen hat mein Promotionsthema zu finden; und bei Prof. Dr. Martin Gaedke, der mich während seiner Zeit in Karlsruhe für die Forschungsarbeit begeisterte.

Die wunderbaren Kollegen der Forschungsgruppe waren eine Inspiration in fachlichen und fachfremden Dingen – danke dafür!

Meiner Frau und meinem Sohn verdanke ich am meisten: Winnie, ich hätte mir keine bessere Unterstützung wünschen können! Kim, die Begeisterung mit der Du die Welt entdeckst ist wunderbar und motivierend!

# Contents

# List of Figures

# List of Tables

# 1 Introduction

*"The Web is the minimal concession to hypertext that a sequence-and-hierarchy chauvinist could possibly make."* [TEDN99]

Ted Nelson, hypertext visionary and creator of the term hypertext

To structure ideas and information can be hard work. For instance, before the first words of this thesis were written, the author faced the challenge of specifying the order in which the contents would be presented. In addition to this sequential structure, a hierarchical structure had to be defined on top: the text was divided into chapters, sections and subsections. Such structures that describe the overall organization of a document or document collection will be referred to as content organization (CO).[1] In this thesis, we understand the CO as the conceptual arrangement of the contents as it is perceived by humans. The CO may or may not correspond to the syntactical organization of the document. The CO can consist of multiple, overlapping conceptual structures, e.g. a hierarchy and a sequence. Each CO-structure can be modeled as a graph: The vertices represent pieces of content and the edges their relation. Considering the structure of this thesis as example again, the content pieces are given by headlines and text paragraphs and the edges by their hierarchical relationships. While this example illustrates that the principle of organizing contents into hierarchies is universal for all kind of media, the focus of this thesis will be on COs for organizing collections of HTML documents. The collections that we consider are Web sites. The CO of a Web site is the logical structure in which the pages of a Web site are organized in the eyes of humans. In this thesis, we research how COs of Web sites can be automatically extracted and how they can be exploited to enhance applications such as search engine user interfaces.

[1] A definition of the term “content organization” is provided in Chapter 2.

**Benefits and Relevance**

Many applications would benefit from methods that allow mining the CO because the CO represents the organization of a Web site as it is perceived and interpreted by humans. The CO guides the human interaction with a site. The CO describes how the pages of a site relate in detail. A hierarchical CO corresponds to a precise, well-engineered topical segmentation. It allows, for instance, to identify pages that contain more specific information on the same topic of a given page (those pages are the child pages). Machine-processing of Web-resources becomes increasingly important at the moment and if the CO could be made available to machines at a large scale, this information could be used in different fields. On the one hand, for instance, applications that involve human search could be improved, e.g. methods of indexing, ranking and presenting search results, because they could build on top of a more fine-grained model of the site structure. On the other hand, methods that involve the analysis of human behavior such as Web analytics or usability engineering could be enhanced because the site structure as a major influence factor could be reverse engineered and included in the analysis as well.

**Why Hierarchies?**

In principle, the usage of sequences and hierarchies for structuring documents has not changed in centuries. But inspired by the rise of new possibilities for storing and representing content electronically, a fundamentally different way of writing and representing knowledge has fascinated visionaries and researchers since the mid-20th-century: Hypertext – the idea of content chunks that are not formed into sequential or hierarchical structures a priori but are loosely coupled by associative linking. Ted Nelson who coined the term "hypertext" considers it as a concept that is opposed to traditional COs: "Hypertext is non-sequential writing" [NELS75]. In hypertexts, users can freely choose those paths through the information chunks that best suit their information needs. The contents are presented without predefined structures such as hierarchies and sequences.

Today, outside the research community, the term hypertext does not denote a concept of representing information anymore but a specific implementation of a hypertext system that outshines all others: the World Wide Web (WWW). The above quote by Ted Nelson illustrates that not all hypertext pioneers believe that the WWW is a hypertext system deserving this name. In the eyes of Nelson, sequential and hierarchical COs dominate the WWW. We argue that this is true. Almost every Web site today has some kind of hierarchical menu. Usability experts agree that hierarchical structures are "far and away the most common" [GARR11]. *But*, the negative connotation Nelson attributes to sequences and hierarchies was neither shared by other hypertext pioneers nor is it shared by today's experts on user experience. In a study on early hypertext systems that preceded the WWW, Conklin reports that 16 out of 18 considered hypertext systems support additional hierarchies [CONK87]. One can conclude that other early adopters considered hypertext as a complement to traditional CO structures, not as a replacement. Up-to-date books on Web design also recommend the use of hierarchies because well-designed hierarchies are the "foundation of almost all good information architectures" [MORO06]. Sequential structures are common as well, and it is hard to imagine a WWW without them. For instance, search results are returned as sequential lists and it seems there is no reasonable alternative. Nevertheless, in the following, the focus will be on hierarchies because (1) sequences can be considered as single-level hierarchies, and (2) hierarchies provide more valuable information (as we will see in Section 3.1).

**Current Web Standards Do Not Feature Machine-Readable COs**

The ability of current Web standards to encode COs in a machine readable way is limited, even though the idea of hierarchically organized contents is an integral part of the WWW since its beginnings, which is reflected in the design of Uniform Resource Locators (URLs), the addressing scheme used to identify resources. The path part of URLs assumes that the resources residing on a host are arranged hierarchically, similar to a hierarchical file system [TBLM94]. However, URL hierarchies and the CO as it is perceived by humans diverge and the CO cannot be extracted from URLs in general. One reason is that today's Web pages are generated dynamically. Thus, one resource, i.e., one URL path on a host, often represents a multitude of different pages, which are distinguished by non-hierarchical query strings. Another reason is that multiple hierarchy levels would lead to long URLs which are difficult to memorize and to type. Hence, in practice, COs are not accurately represented by URL paths. Aside from URL paths, current Web standards do not allow Web developers to explicitly define hierarchical COs in a machine-readable way. In HTML-code, there is only a single type of hyperlink, which can represent parent-child relationships as well as non-hierarchical, associative relationships. Hence, these semantics cannot be distinguished on code level and CO structures cannot be easily extracted automatically.

**Challenges and Limitations of Current Methods for Site Structure Mining**

Current Web standards provide great freedom of visual design. Hence, although the COs of current Web sites are not encoded in a machine-readable way, hierarchical structures can indeed be visualized for humans. This is achieved by grouping hyperlinks visually into *navigation elements*, i.e. menus. Navigation elements reflect the semantics of the hyperlinks, e.g., the fact that the main menu contains links to the pages of the first hierarchy level and a local menu contains links to the siblings of the active page and its child pages and so on. Humans can easily decode these semantics based on the page layout. Users are able to understand which navigation elements constitute the hierarchical CO and which fulfill other purposes. In contrast, current methods of *Web structure mining*, i.e. approaches for automatically analyzing the structure of the WWW, do not take into account that links can have different semantics in dependence of the navigation element to which they belong. The fundamental model of current Web structure mining approaches is the Web graph. The vertices of the Web graph represent documents and the edges represent the hyperlinks between the documents. The Web graph can be derived directly from the untyped hyperlinks in the HTML markup of the analyzed documents. The Web graph is the most straightforward way of representing the structure of the Web or of parts of the Web. Many successful solutions are based on the Web graph, e.g. well-known ranking methods such as PageRank [PBMW99]. But parent-child relationships cannot be distinguished from other

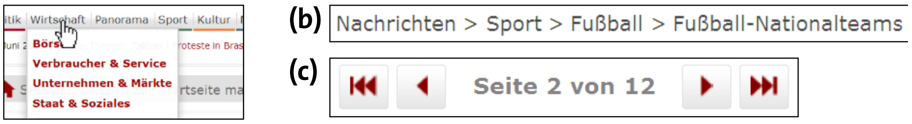

itik | Wirtschaft | Panorama | Sport | Kultur
Börse
Verbraucher & Service
Unternehmen & Märkte
Staat & Soziales
(b)
Nachrichten > Sport > Fußball > Fußball-Nationalteams
(c)
Seite 2 von 12

**Figure 1: Examples of navigation design patterns.** (a) Hierarchical pop-up menu, (b) breadcrumb trail, (c) pagination

relationship types in the Web graph and, thus, this model is not suitable for extracting the original CO.

If neither the URL structure nor the link structure provide machine-readable hierarchy information, the visual presentation could be automatically analyzed instead. But imitating the human perception and conducting a visual page analysis in order to mine navigation elements, i.e. link semantics, is a task that seems to be almost unsolvable in the near future. Automated recognition and distinction of objects based on vision is up until now considered as an extremely difficult computational problem [PICD08] and general solutions are not in sight. Even the automated segmentation of Web pages into visually distinct regions as they are perceived by humans is challenging today (cf. Section 4.3). And even more difficult is the task of determining the purpose of a navigation element and deriving the link semantics based on the graphical presentation. All kind of visual characteristics such as font types, font sizes, positions, colors, et cetera must be taken into account as well as the visual characteristics of all other blocks on the page.

### An Alternative Approach to CO Mining: Reverse Engineering Web Navigation Elements

In this thesis, we present an alternative way, which is based on the idea of *navigation design patterns*. With the growth of the WWW, certain types of navigation elements such as hierarchical pop-up menus, paginations or breadcrumb trails (Figure 1) have evolved and are now used as de facto standards, even if they have not been formally defined. Users as well as designers are familiar with these concepts. From the designer's perspective, they can be considered as design patterns because they represent well-known solutions to common navigation problems. Academia and designers widely agree on the existence and importance of navigation design patterns. Inspired by the concept of design patterns in architecture and software engineering, academia has focused on collecting common navigation design patterns (e.g. [GECO00]). Furthermore, many practical books on Web information architecture or navigation design contain catalogues of navigation design patterns (e.g. [KALB07]).

The idea of navigation design patterns is the foundation of all mining methods presented in this thesis, because we can base our mining methods not only on explicitly defined standards but also on de-facto standards for implementing Web navigation. This reduces the problem from mining for all kind of navigation elements, i.e. visually separate blocks of links, to the problem of mining specific patterns with specific properties. We can focus on those navigation design patterns that allow determining the hierarchical CO, e.g. bread-

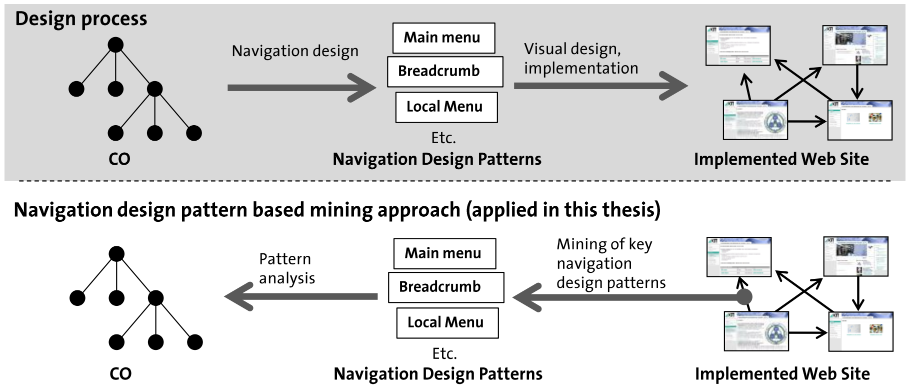

Design process
Navigation design
Main menu
Breadcrumb
Local Menu
Etc.
Visual design, implementation
CO
Navigation Design Patterns
Implemented Web Site
Navigation design pattern based mining approach (applied in this thesis)
Pattern analysis
Mining of key navigation design patterns
Main menu
Breadcrumb
Local Menu
Etc.
CO
Navigation Design Patterns
Implemented Web Site

**Figure 2. Approach overview. We exploit the fact that designers use navigation design patterns as de-facto standards for implementing Web navigation. The properties of common navigation design patterns are well-known and we can derive dedicated extraction-rules for selected patterns.**

crumb trails. Hence, instead of ignoring link semantics as done in Web-graph based mining approaches or conducting a holistic visual analysis, we focus on well-understood patterns that designers use to implement Web navigation (Figure 2). We exploit the knowledge about these patterns for finding features that allow mining them. Those features are not necessarily visual features because the characteristics of design patterns are also reflected in the way they are implemented in HTML-code. Since we rely on a-priori knowledge about specific design patterns, most mining methods presented in this thesis are rule-based. If rules alone are not sufficient because of noise and irregularities, we apply machine-learning methods in addition (cf. Section 6.4).

## Research Questions and Contributions

In this thesis, we research how the navigation elements belonging to specific navigation design patterns can be mined and how they have to be processed in order to regain the original CO. We furthermore present selected applications based on CO-mining.

In more detail, the research problems addressed in this thesis are as follows:

– *Which navigation design patterns can be used for CO-extraction?*

  Most books on Web information architecture discuss navigation design patterns that can be applied to transform hierarchical COs into a hyperlink structure. For instance, breadcrumbs or pop-up menus are typical patterns for implementing navigation on hierarchical COs. But the opposite direction – how the CO can be regained from the HTML-code of specific navigation design patterns – has not been addressed yet.

– *How can those navigation design patterns that allow CO-extraction be mined? Are the methods accurate enough to produce valuable hierarchy information?*

Methods for segmenting Web pages and determining the types of the resulting blocks have been described before. However, most of these methods do not have a particular focus on mining navigation elements and their types. Some methods extract navigation elements in general, without discerning different types. In addition, the existing methods have only been evaluated as part of more complex solutions. Thus, the actual performance of existing methods for classifying navigation elements and their suitability for CO-mining is widely unknown.

– *How can we evaluate the correctness of CO-mining solutions?*

  The CO as understood in this thesis is the content structure as perceived by humans. Designers usually specify the CO formally, e.g. by using tables or graphs. But in the resulting implementation, the CO is only visually encoded without a machine-readable specification. Thus, to evaluate CO-mining methods at a large scale, the results have to be compared to the human interpretations of the visual design.

– *Which new possibilities arise from the availability of CO-information?*

  Although Web structure mining is a lively field of research and all information that can be used for indexing, ranking and searching Web resources is considered to be very valuable by the research community, few existing works have focused on CO-mining and consequently applications based on CO-data have rarely been discussed.

This thesis addresses the above research questions. The structure of this thesis and the main contributions are as follows:

**Chapter 2: Conceptual Approach**

- In Chapter 2, we define concepts and terms related to the addressed research questions and **introduce the O$^3$-model, a reference model for the structural aspects of Web sites.** Most practical literature on Web usability distinguishes between the underlying content organization on the one hand and the hyperlink structure, i.e. navigation, on the other hand (e.g. [GARR11], [MORO06]). We formalize this separation of concerns and present a model for Web navigation that consists of three distinct aspects, Content Organization (CO), Functional Organization (FO) and Page Organization (PO). In the O$^3$-model, these aspects are closely connected by transformation rules. We show that these rules actually correspond to the well-researched navigation design patterns and that the O$^3$-model can be applied to describe and classify patterns. The O$^3$-model is the foundation of the rest of this thesis, including the analysis of related work.

**Chapter 3: Problem Statement**

- In Chapter 3, we analyze and refine the problem statement based on the conceptual framework developed in Chapter 2. We describe applications that would benefit from CO-mining solutions, outline the basic CO-mining process and technical challenges.

**Chapter 4: Related Work**

- The tasks of the basic CO mining process (cf. Section 3.3.1) define the scope and structure of an extensive related work review, conducted in Chapter 4.

**Chapter 5: Towards a Solution**

- In Chapter 5, we show that graph representations of navigation elements reveal the used navigation design patterns. We discuss **a catalogue of sample patterns**. Furthermore, we describe the implementation and evaluation **of a basic method for mining navigation elements.** Mining navigation elements has been a side-issue of previous research on Web mining. The common approach is to extract textual, structural and visual features of page segments to identify navigation elements. However, these methods have not been evaluated previously since they were only used as part of larger solutions. We implemented and extended conventional methods and demonstrate their limits.

**Chapter 6: The GRABEX-Approach**

- In Chapter 6, we introduce the **Graph-based Block Extraction (GRABEX) approach for mining navigation elements,** an accurate and efficient approach that overcomes the limitations of conventional methods. GRABEX is based on link analysis but in contrast to previous approaches, it does not analyze the links between entire pages but between page blocks that potentially belong to the same navigation element. Whether or not these page blocks represent a valid navigation element is revealed by characteristic graph patterns. The proof of concept is delivered by the **presentation and evaluation of CO mining solutions based on GRABEX.** We introduce two dedicated instances of the GRABEX-approach, which focus on mining navigation design patterns that are particularly suitable for extracting the hierarchies. The first GRABEX-instance is the Menu-Miner-algorithm, which allows the efficient extraction of menus that represent hierarchy levels. We demonstrate that the method can be applied to detect site boundaries more accurately than previous solutions. We propose additional rule-based algorithms to process the extracted menus and reassemble the underlying hierarchies. An extensive evaluation shows that the method performs with high precision and good recall. The second GRABEX-instance allows the extraction of breadcrumbs. In an evaluation, we found that it is able to achieve perfect precision and a good recall. Thus, we are the first ones to describe an applicable solution to the breadcrumb mining task.

**Chapter 7: Augmentation of Search Results with Site Structure Information**

- In Chapter 7, we focus on **applications that benefit from CO-mining**, in particular search engine user interfaces. By analyzing the usage data of three highly-frequented Web sites, aligned with the hierarchical CO-structure, we show that the availability of CO-data provides new insights. We found for all of the three analyzed sites that the CO-structure strongly influences the navigation behavior. Users tend to navigate down, following the edges of the hierarchy. We discuss the implications for the design of search user interfaces and we conclude that hierarchy information can help users save clicks.

The contributions presented in this thesis have been partially published in:

- M. Keller, A. Blanchard, and M. Nussbaumer, "Decomposing the Web Graph: Towards Information Architecture Mining," in *22nd ACM Hypertext Conference, Posters and Demos*, Einhoven, Netherlands, 2011

- M. Keller and M. Nussbaumer, "Beyond the Web Graph: Mining the Information Architecture of the WWW with Navigation Structure Graphs," in *Proceedings of the 2011 International Conference on Emerging Intelligent Data and Web Technologies*, Tirana, Albania, 2011

- M. Keller and M. Nussbaumer, "Graph Access Pattern Diagrams (GAP-D): Towards a Unified Approach for Modeling Navigation over Hierarchical, Linear and Networked Structures," in *Current Trends in Web Engineering*, vol. 7059, A. Harth and N. Koch, Eds. Springer Berlin Heidelberg, 2012, pp. 155–158

- M. Keller and M. Nussbaumer, "MenuMiner: revealing the information architecture of large web sites by analyzing maximal cliques," in *Proceedings of the 21st international conference companion on World Wide Web*, Lyon, France, 2012

- M. Keller, P. Mühlschlegel, and H. Hartenstein, "Search result presentation: supporting post-search navigation by integration of taxonomy data," in *Proceedings of the 22nd international conference on World Wide Web companion*, Rio de Janeiro, Brazil, 2013

- M. Keller and H. Hartenstein, "Mining Taxonomies from Web Menus: Rule-Based Concepts and Algorithms," in *Proceedings of the 13th International Conference on Web Engineering*, Aalborg, Denmark, 2013.

- M. Keller and H. Hartenstein, "GRABEX: A Graph-Based Method for Web Site Block Classification and its Application on Mining Breadcrumb Trails" in *Proceedings of the 2013 IEEE/WIC/ACM International Conference on Web Intelligence*, Atlanta, USA

# 2 Conceptual Approach

In the introduction of this thesis, we argued that Web sites have *predefined* hierarchical COs, which are visually encoded and differ from the hyperlink structure. In contrast to many previous methods in the field of Web mining (e.g. [BORS92], [MLGR98], [DUCH00], [CLZC05], [LYHL10] or [HOEX12]), we claim not to generate new hierarchies but extract existing, though not easily machine-readable, structures. By existing, we mean that the hierarchical COs have been conceptually created by designers. Although the hierarchical CO is embedded in a non-hierarchical data structure (the hyperlink structure) it can be decoded by human observers based on the visual design (Figure 3). Developing a deeper understanding of the process depicted in Figure 3 and the nature of the involved objects is crucial to formulating the problem statement, assessing related work as well as for evaluating and developing solutions: In this thesis, we are developing methods for automatically analyzing a hypertext system in order to retrieve the hierarchical structures that correspond to the COs observed by human users. Hence, we clarify our perspective on the hypertext system WWW in Section 2.1. Since COs do not correspond to physical objects, we analyze the nature of COs and operationalize the fuzzy concept by providing working definitions in Section 2.2. Furthermore, the CO implementation process and in particular the role of navigation design pat-

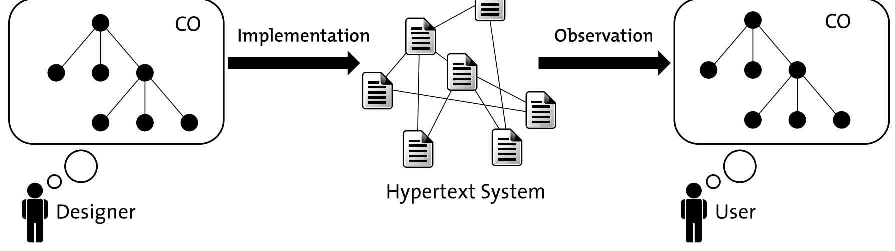


**Figure 3. Creation and observation of hierarchical COs**

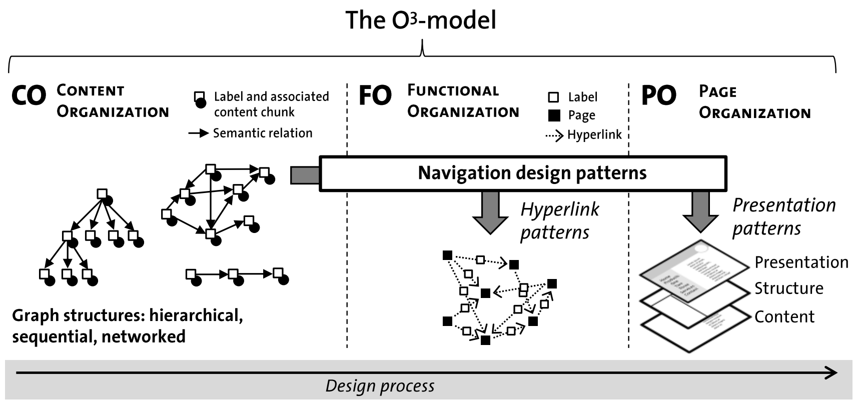


**Figure 4. The O$^3$-model. Navigation design patterns determine the hyperlink structure and parts of the PO.**

terns is examined in Section 2.3 because the resulting observations provide the basis for the solutions presented in this thesis.

The conceptual approach taken in this thesis, including the underlying perspective on the hypertext system, the definition of a CO and the relationship between these aspects is subsumed in the O$^3$-model (Figure 4). In the O$^3$-model, we distinguish three different aspects: The content organization (CO), the functional organization (FO) and the page organization (PO). The CO models the logical organization of the content as graph on labels, each of which is associated with a content chunk. The FO represents the possible page transitions, i.e. the link structure. The PO describes the organization on page level, including the presented content, the page structure and the page presentation. To analyze the implementation process, we assume rational designers, who are trying to maximize usability with minimal effort. Such designers will specify the FO and those parts of the PO that are related to navigation by combining well-engineered and site-specific COs with simple, common menu patterns. We will refer to these menu patterns as navigation design patterns (cf. Section 2.3). Based on the O$^3$-model, we conclude that the key to successful CO mining is reverse engineering navigation design patterns. While the focus of this thesis is on reverse engineering menus, the engineering perspective can also benefit from the concepts developed in this chapter as the GAPD-notation described in Appendix A demonstrates.

An overview over the terms that are introduced in this chapter can be found in Table 1.

**Table 1: Overview over terms introduced in this section**

| Terminology Overview | | |
|---|---|---|
| **Aspect** | **Term** | **Introduced in** |
| Content organization (CO) | Site | Section 2.1.1 |
| | Labels | Section 2.2.2 |
| | Self-explanatory relationships | Section 2.2.3 |
| | Content chunks | Section 2.2.4 |
| | Content organization | Section 2.2.5 |
| | Organizational schema | Section 2.2.5 |
| | – Taxonomy | Section 2.2.5 |
| | – Sequential schema | Section 2.2.5 |
| | – Associative schema (Hypertext) | Section 2.2.5 |
| | – Database-Schema | Section 2.2.5 |
| Functional organization (FO) | Functional Organization | Section 2.1.2 |
| Functional organization (FO) / Page organization (PO) | Navigation design patterns | Section 2.3.2 |
| | – Horizontal / vertical / global / local menu | Section 2.3.2 |
| | – Breadcrumb | Section 2.3.2 |
| | Navigational block | Section 2.3.3 |
| | Navigation element | Section 2.3.3 |
| | Navigational block /navigation element type | Section 2.3.3 |
| Page organization (PO) | Page organization | Section 2.1.3 |
| | Page structure | Section 2.1.3 |
| | Page content | Section 2.1.3 |
| | Page presentation | Section 2.1.3 |
| | Page block | Section 2.1.3 |

## 2.1 The Hypertext System: FO and PO

In Section 2.1.1, we will clarify the hypertext system model that we assume and discuss its generalizability. In the $O^3$-model, the hypertext system is represented by the FO and PO, which will be discussed in Section 2.1.2 and Section 2.1.3 respectively.

### 2.1.1 Choice of Perspective

The focus of the Web mining methods discussed in this thesis is neither on analyzing individual pages nor on analyzing the entire Web. Instead, we are analyzing collections of closely related pages, so-called Web sites. One domain can host multiple Web sites: Figure 5 shows four pages hosted under the same domain. Pages a) and d) are similar with respect to the visual design, the navigation mechanisms and the topics. The same applies to pages b) and c). But comparing both pairs to each other, we can identify differences regarding these aspects. One would say that pages a) and d) originate from the same site and pages b) and c) originate from another site. There are few attempts in literature to define the term site (e.g. in [MEMF07], [ALCZ10]), which usually agree that the pages of a site have at least 1) a common entry point, 2) consistent styles and 3) shared navigation mechanisms. We adopt

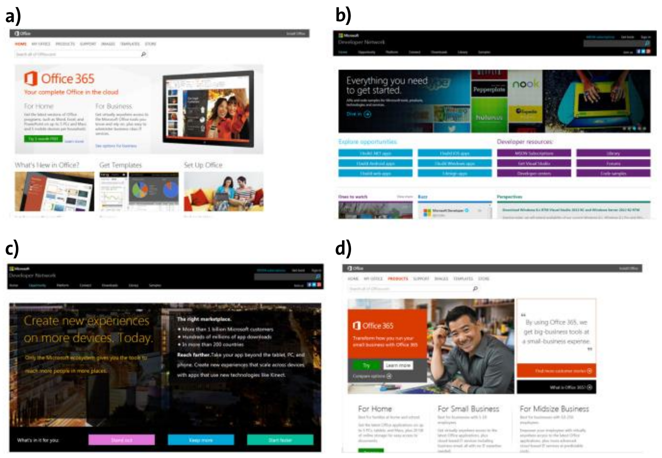

a)
b)
c)
d)
Office 365
Your complete Office in the cloud
Everything you need to get started.
Create new experiences on more devices. Today.
Office 365

**Figure 5. Different sites hosted under the same domain name. Pages a) and d) originate from the same site, pages b) and c) from another.**

this understanding of a site. Since a common entry point and a shared navigation mechanism imply a shared CO and FO, while consistent styles imply a shared PO design, we can say that *a collection of Web pages with shared CO, FO and PO is called a Web site*.

### 2.1.1.1 Hypertext System Model

The focus of this thesis is on the CO as it is perceived by human users and how it can be mined automatically. In the client-server architecture of the WWW, both, human users and Web crawlers represent the client-side. Thus, the client-perspective is adopted in this thesis. We abstract from all server-side data management and business logic because these aspects are hidden from users as well as from Web crawlers. In the original design of the WWW, the client-perspective was simple: User agents retrieved static HTML pages from Web servers. They parsed the HTML code to render the visual representation. Documents contained hyperlinks to other static HTML pages. User interaction was limited to proceeding from one document to another by the use of hyperlinks. We adopt this fundamental perspective and consider a Web site as collection of linked pages. The PO describes the visual organization of the pages, while the FO represents the hyperlink structure. All analyses and methods presented in the remainder of this thesis will assume this fundamental hypertext system model.

### 2.1.1.2 Generalizability of the Hypertext System Model

Applied to today's Web, the fundamental hypertext system model is a simplification because user interactions are not limited to moving from one URL to another by clicking hyperlinks. For instance, Web forms can be used to submit search queries or other data and user actions can trigger client-side scripts that manipulate the document using the DOM[2] API without retrieving an entire new document from a different URL. However, the hypertext-navigation model is appropriate for addressing the research questions of this thesis:

- From an information architecture perspective, the appearance and behavior of the user interface matters but not how the changes in the state of the UI are implemented. From this perspective, it makes no difference, whether the DOM is modified on the client-side or whether a new page from a different URL is loaded when a user clicks on an element. Both can be considered as valid transitions according to the hypertext navigation model. However, from the Web crawling perspective, the difference matters, because crawling client-side modifications of the DOM is challenging and a research topic on its own [CDMM12]. However, Web developers are aware of this fact and usually implement Web sites in a way that all important information can be accessed without Javascript execution to ensure their availability to search engines. The Web crawling experiments conducted in this thesis are also based solely on following links and do not involve the execution of Javascript – with good success as the evaluations show.
- Aside from content that can only be accessed via Javascript execution, Web forms that require user input represent a barrier that Web crawlers cannot easily cross. For example, Web contents that can only be accessed by keyword search are difficult to retrieve by Web crawlers, even though some strategies exist (e.g. [MKKG08]). However, this is a general problem of Web crawling and not specific to the methods presented in this thesis. The so-called hidden Web can be mapped on the hypertext-navigation perspective if each possible form entry is considered as a hyperlink and each follow-up state is considered as an individual Web resource.

### 2.1.2 The Functional Organization

While the CO defines semantic relationships, we consider hyperlinks, which form the FO, solely as page transitions. CO and FO are not merely two stages in the design process but describe different aspects. The distinction of both aspects is not only suitable in case of hierarchies. To illustrate this, we consider an example from the physical world: a book. The book pages are obviously arranged sequentially and, hence, the CO can be thought of as a linear graph. However, a reader does not have to traverse the pages in this order. For example, from the first page he can proceed to any other page of the book (Figure 6(a)). These options for navigating define the FO. In the digital world, there are many different ways of implementing the FO on top of a sequential structure. For example, image galleries can often be traversed only in a strictly sequential way (Figure 6(b)). In other cases, the FO allows

[2] Document Object Model (DOM) is an API specification that allows an object-oriented manipulation of structured documents such as HTML instances (http://www.w3.org/DOM/). In contrast to the definition, the term DOM is commonly used – also in this thesis – to denote not only the API but also the browser's internal representation of the document.

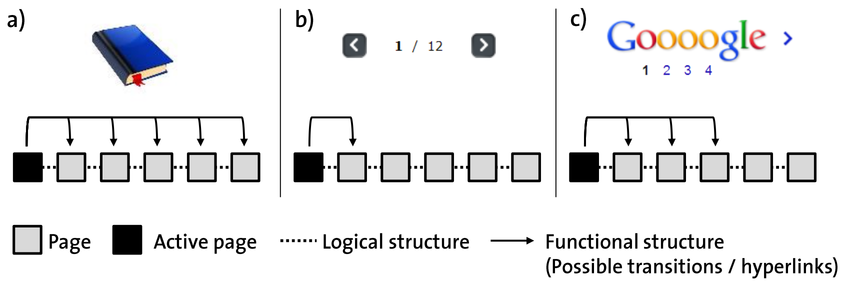


**Figure 6. The FO differs from the CO. (a) The pages in a book are arranged sequentially (CO) but a user may jump from a specific page to any other page. (b, c) The possible transitions between pages presented on a screen depend on the FO. In example b) users are able to proceed to the successor of the active page while they can proceed to one of the next three pages in example c).**

skipping a defined number of pages (Figure 6(c)) or jumping directly to the first or last page. The CO models characteristics of the content. In contrast, the FO models characteristics of the content *and* the medium used to present it.

### 2.1.3 The Page Organization

So far, we only considered pages as atomic nodes in the FO. However, navigation design patterns, which are the foundation of the mining methods present in this thesis, also have implications on the structure and design of individual pages. We need to consider these aspects in order to implement successful CO mining solutions and hence, integrate them into our model. We refer to the page-related aspects as page organization (PO). In our model, Web pages consist of chunks of informative content and outgoing hyperlinks. The PO is the way in which hyperlinks and content chunks are arranged and decorated on a page. We distinguish three PO layers: content, structure and presentation (Figure 7). This corresponds to the document model proposed by the W3C in the Web Content Accessibility Guidelines 1.0 [WCRE99]. According to the W3C recommendation, the content is what the document "says to the user" [WCRE99], structure is given by the logical organization of the document and the presentation is the way the document is rendered. We specify the layers in more detail and adopt them in the O$^3$-model:

– **Page content:** The page content is given by the content chunks and hyperlinks that are presented on the page.

– **Page structure:** The spatial arrangement of the content on the page is called page structure. In contrast to the definition by the W3C in context with the Web Content Accessibility Guidelines [WCRE99], structure is not understood as the logical organization of the page but as the basic layout. This understanding of structure corresponds to a wireframe model (Figure 7), which is commonly used in current Web design practice.

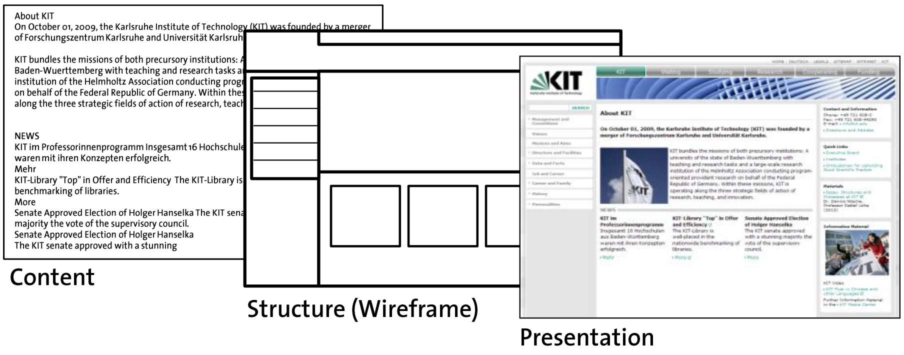


**Figure 7. The PO consists of the content layer, structure layer and presentation layer**

Wireframes are abstracted presentations of the visual design that lack colors and other decorative elements.

– **Page presentation:** The presentation layer includes all further aspects of the visual design such as colors, fonts or decorative elements.

A wireframe model typically divides a page into a number of rectangular areas that serve different purposes (e.g. page header, main content, menus, etc.). The wireframe model illustrates the arrangement of these functional blocks on a page, not the graphic design and the visual borders. However, the functional blocks are also distinguishable in the final graphic design (cf. Figure 7), i.e. the page presentation layer. This is required because users must be able to distinguish these areas and understand their purpose in order to interact with a Web site. We refer to these rectangular areas that are visually separated from their surroundings as **page blocks**. Depending on the context, the term is also used to denote the markup code that corresponds to the area. Page blocks can be nested, i.e. a page block can contain other page blocks. Visual separation does not always refer to explicitly drawn boundaries. Different background colors or delimiting white spaces can also separate page blocks in the eyes of the users. The underlying mental processes are complex – an own branch of research, Gestalt psychology, has focused on the laws of perceiving shapes and their boundaries (cf. Section 3.3.2).

Web sites are visual media and the page organization defines the final appearance. Thus, the CO and the FO are communicated through the PO. This means, that users must be able to understand the logical structure of the contents (CO) and the possible next navigation steps (FO) based on the visual presentation.

## 2.2 The Perceived Organization: CO

While the components of the FO, pages and hyperlinks, correspond to physical entities, this does not apply for the CO. Hence, we provide formal definitions for the components of the CO in order to clarify our model.

If designers want to specify a hierarchical CO, they can simply draw a tree-structured graph on the whiteboard or create a nested bullet list. In these typical representations, the pages are represented by their titles. The mathematical equivalent would be a tree-structured graph, whose nodes are textual labels. While a hierarchical CO can always be thought of as tree on labels, not all trees on labels are potential COs. There are special characteristics that such trees must fulfill in order to be a potential CO that could be used to organize a Web site. We analyze these characteristics in Section 2.2.1 and find that 1) labels must to be meaningful, 2) relations must be self-explanatory and 3) labels must describe the content well. The definitions in Section 2.2.2 (labels), Section 2.2.3 (self-explanatory relations) and Section 2.2.4 (descriptive label assignment) follow these observations.

### 2.2.1 Definition Approach

To motivate the definitions provided in this section, we consider the most basic and intuitive hierarchy browsing scenario: A users arrives at the homepage of a site and descends the hierarchy in order to locate some specific information. Based on this fundamental scenario, we derive properties that COs must have to fulfil their purposes. These properties are used to formulate working definitions afterwards. In the basic scenario, we assume that a user arrives on the homepage of a site seeking specific information. The hierarchy consists of labels which are associated with pages. On each page, only the labels of the child nodes are displayed to identify the corresponding links. We assume that the user has an idea about the probable label of the page of interest and that this page is not located at the first level of the hierarchy and, hence, not linked on the homepage. We can now ask, which requirements the hierarchy must fulfill to allow the user to navigate to the page of interest:

- First, the user must be able to *decode the meaning of each label* in order to make the right decision.
- Second, he must be able to understand that it makes more sense to associate the page of interest with a specific first level node (the correct branch) than with the other first level nodes. This means that he must be able to *decode the meaning of the tree edges* with no other information at hand except the link names. In this sense, the relation must be self-explanatory.
- Third, if the user reaches a destination page directed by the link labels, navigation was only successful if the information he is seeking for is really located on this page. Hence, it is required that the *labels and the content match*.

We define COs as structures that fulfill all three requirements.

## 2.2.2 Labels

***Term definition 2-1 (Labels)*** *A label is a textual or visual symbol that different human interpreters associate with similar mental models.*

One difference between a random graph of labels and a (potential) CO is that the CO is "meaningful" or, in other words, the graph edges represent semantic relationships. Figure 8(a) shows a hierarchical structure that has semantic edges. In contrast to the structure shown in Figure 8(b), the edges intuitively "make sense". If we assume that the structure shown in Figure 8(a) is the CO of a Web site, we can state that it is a reasonable way of organizing the content based on the labels, i.e. page titles –without even knowing the actual content. The site could offer operating instructions for various products, but it could be a shopping portal or a store locator as well. Thus, in the $O^3$-model, COs are modeled as graphs on labels not as graphs on content objects. As labels we consider symbols that carry meanings. To clarify what this means, we use the concept of mental models in our definition. Mental models have been proposed in the field of cognitive sciences [GEST83] to explain human thinking and problem solving. Mental models denote abstractions of real world entities in our minds, or, in other words, the data-objects on which our mind operates. For example, we consider the textual symbol "Kitchen" as a label, because human interpreters would associate it with a similar real world entity, which means that they agree on the meaning. In contrast, the symbol "aaaaa" is not a label, because it has no agreed-upon meaning. The agreed-upon meaning of a symbol is essential for its suitability as a vehicle for communication, which requires that consumer (user) is able to decode the intended message of the sender (designer). In the context of this thesis, labels have the purpose of describing the content of Web pages.

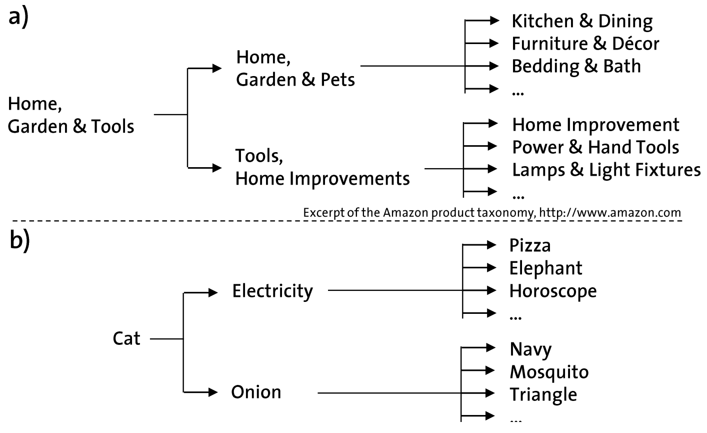


**Figure 8: Two hierarchies of labels. Obviously, example a) can be used to organize the contents of a site in an appropriate manner because the relations "make sense", while example b) is not useful for organizing the contents. We would consider example a) as a valid CO, in contrast to example b).**

### 2.2.3 Self-Explanatory Relationships

***Term definition 2-2 (self-explanatory relationships):*** *Given a relation $R \subseteq L \times L$ on a set of labels L, $R$ is self-explanatory if different human interpreters are able to map the relation on mental associations with similar semantics.*

This means that a relation on a set of labels is self-explanatory if the meaning of the relation is obvious to humans. The difference between the two structures depicted in Figure 8 is that the edges in Figure 8(a) are self-explanatory, while they are not in Figure 8(b). The idea of Term definition 2-2 is illustrated in Figure 9. According to the definition of a label, humans are able to associate labels with mental models. In case of a self-explanatory relation, humans are also able to map each tuple of labels in the relation on a mental association between the two corresponding mental models. Since the purpose of our considerations is a term definition and not gaining insights into the field of cognitive sciences, we think of a mental association simply as an obvious logical connection between two things or, more precisely, between their cognitive representations, i.e. mental models. For example, humans would agree that there is an obvious connection between the concepts "car" and "steering wheel", which is that a steering wheel is part of a car. At the same time, for instance, the association between "soap" and "rhinoceros" is much more difficult to find. Another aspect of the definition is that it requires semantic similarity of the mental associations two individuals have. This means that two individuals must agree on the meaning of the connection associated with a tuple of labels.

### 2.2.4 Descriptiveness of Labels

We require that a label is associated to a piece of content in a way that it describes the content well. To be more precisely, a label creates expectations about what information a user can gain from a piece of content. This is a two-step process (cf. Figure 10): First, a user decodes the label itself (e.g., he understands the meaning of the word "contact") and, second, he hypothesizes which information he can gain from the content denoted by the

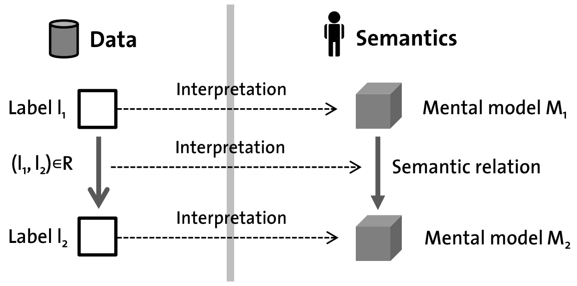


**Figure 9. Self-explanatory relationships.Tuple of labels are considered as self-explanatory relationships if humans are able to associate them with agreed-upon meanings. For instance, a tuple (kitchen, oven) "makes sense" in a way that the connection between the labels (which represent mental models) is obvious.**

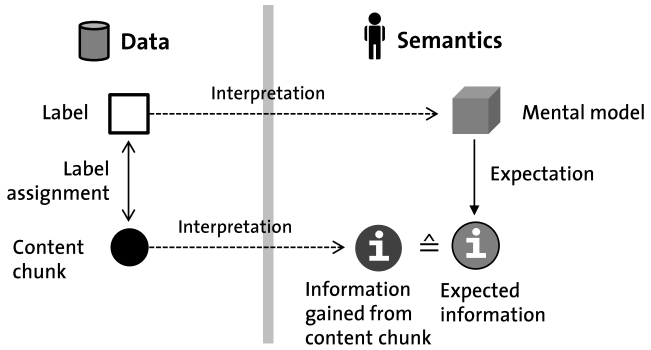


**Figure 10. Descriptiveness of labels. An assignment of labels to content chunks is descriptive if the expectations about the available information created by the labels match the actual information that can be gained from the content.**

label (e.g., he assumes that he will find email and postal addresses on a page). At this point, it becomes clear that it is useful to assume that the pieces of content we are dealing with can be generally exploited by humans to gain information[3]. Hence, we introduce the term content chunk as follows:

***Term definition 2-3 (Content chunk):*** *A data object is called content chunk if it can be exploited by humans to gain information.*

Humans are able to gain information from a data object if they are able to decode the semantics. A data object can be everything, for example a word, a paragraph, a book or a picture. We can apply different levels of abstraction: Single sentences or pictures are regarded as content chunks, but also larger aggregations of content. Although the "page"-concept is not part of the CO but the FO in the $O^3$-model, we can assume that the content chunks we are dealing with correspond to the content of an entire Web page each. While all definitions also apply to content chunks on a sub-page level, these fine-grained structures are beyond the scope of this thesis.

***Term definition 2-4 (Descriptiveness of labels):*** *Given a set of labels L, a set of content chunks C and a relation $A \subseteq L \times C$ that assigns labels to content chunks; A is called a descriptive label assignment if for each tuple $(l, c) \in A$ the information that users expect based on the mental model inferred from l is similar to the information that can actually be gained from c.*

With **Term definition 2-3**, we can say that a label assignment is descriptive if the expected information matches the information that can be gained from a content chunk (Figure 10). For example, the label "shop" triggers a mental model of a facility or service that allows purchasing things. In context with a Web site, it creates the expectation of an entry page to a Web store. If the label is assigned to a content chunk that represents such an entry

[3] We use the term information to denote meaning attributed to a message (data) by humans (cf. Goos and Zimmermann [GOZI06])

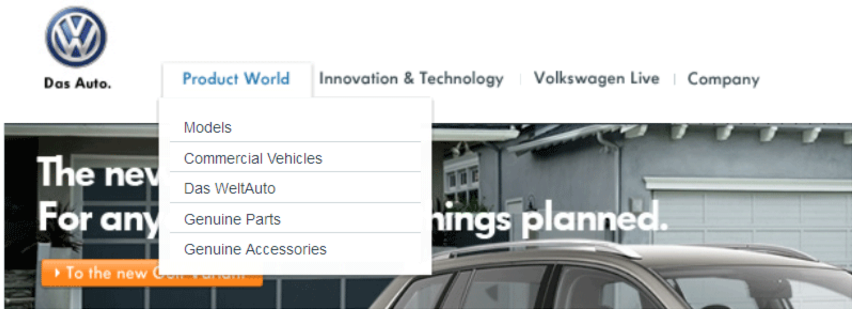


Figure 11. A hierarchical CO implemented as a pop-up menu[4]

page, i.e. if the menu item “shop” really links a store entry page, the user expectations are met and the label assignment is descriptive according to the definition.

### 2.2.5 Content Organization and Organizational Schemas

***Term definition 2-5 (Content organization, CO):*** *A self-explanatory relation on a set of labels L is called content organization of Web site W if a descriptive assignment of the elements of L to the content chunks provided by W exists.*

We define the term CO based on the introduced concepts. To summarize, a CO of a Web site consists of a set of labels on which a relation is defined. Human interpreters agree on the meaning of the labels and are able to infer the meaning of the relation. In addition, the labels are assigned to the content chunks, i.e. pages, of a Web site in a way that the user expectations are met. To illustrate this, Figure 11 shows a screenshot of a Web menu that represents a CO (or a part of a CO). Humans are able to associate meanings with the menu items (“Product World”, “Innovation & Technology”, etc.), thus they are valid labels. Parent-child relationships are visually encoded, e.g. it is obvious that the label “Model” is a child of “Product World”. The relation is self-explanatory because humans can infer the meaning of the relation, e.g. that all the child items of “Product World” refer to products of the brand. Each label is linked with a Web page (content chunk) that contains the expected information. Hence, the labels are assigned to content chunks in a descriptive way and we can conclude that the underlying structure is a CO according to our definitions. It should be pointed out here that the menu shown in the figure only *represents* the CO. The CO is the graph structure that is underlying the menu.

For readability, we will not always strictly follow the terminology introduced in this section: According to the proposed definitions, the CO is a relation on labels and not on the content chunks or the pages themselves. However, each label is associated with a specific content chunk and page. Hence, whenever reasonable, we will not explicitly distinguish between a content chunk, the containing page and its label.

[4] Source: http://en.volkswagen.com/en.html

We can distinguish different types of COs in dependence of their graph structure, e.g. hierarchical COs and sequential COs. We will refer to the type of a CO as its organizational schema:

***Term definition 2-6 (Organizational Schema):*** *An organizational schema denotes a subset of all possible COs whose graph-structures have specific characteristics.*

Although the focus of this thesis is on the hierarchical schema, we will also shortly discuss other schemas in the following because the framework developed in this section is not limited to hierarchies. We propose to distinguish four different organizational schemas (cf. Figure 12):

– **Hierarchical schema (taxonomy):** A CO is organized according to the hierarchical (taxonomical) schema if the graph-representation of the CO is an arborescence, i.e. a directed tree in which all edges are directed away from the root. In a hierarchical CO, the nodes, i.e. labels, can be considered as classes. Each label describes not only the associated content chunk but all content chunks associated with descendant labels. For example, if the structure shown in Figure 8(a) is the CO of a Web site, the label "Home, Garden & Pets" describes not only a single page but also the pages that are associated with the child labels. Hence, hierarchical COs can be considered as hierarchical classification schemas, i.e. taxonomies. In this thesis, we will use the terms hierarchical schema and taxonomy synonymously.

– **Sequential schema:** A CO is organized according to the sequential schema if the graph-representation of the CO is a linear graph. A linear graph is an arborescence without branches. The sequential schema is used to arrange content chunks in a strict linear order. For example, if a long article is split up into multiple pages, these pages are arranged sequentially, because it only makes sense to traverse the pages in the original order. Another example is a photo gallery in which the images are arranged chronologically.

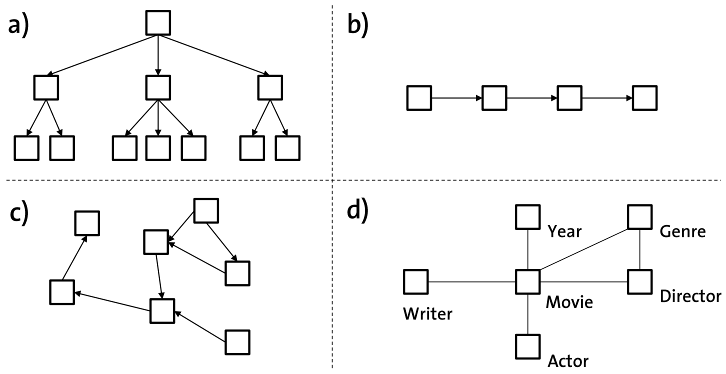


**Figure 12. Organizational schemas: a) Hierarchical, b) sequential, c) associative and d) database schema.**

– **Associative schema:** A CO is organized according to the associative schema if the graph-representation of the CO is not based on a specific pattern. The edges associate related labels without following a fixed schema. In both the hierarchical and the sequential schema, a single node exists from which all other nodes can be reached and that has no incoming edge itself. Such an entry node does not necessarily exist in the associative schema. Hierarchical and sequential organizational schemas can be used to organize traditional media, e.g. books, as well, but the associative schema is strongly connected to the idea of hypertext. The content chunks are not pushed into fixed organizational schemas but are loosely coupled by associative linking. Nelson, who coined the word hypertext, had the associative schema in mind (cf. [NELS75]). Based on the proposed framework, we can describe the concept of hypertext as follows: A CO that is organized according to the associative schema is called **hypertext** if the labels are themselves part of the content chunks. In hypertext, the labels are not separated from the content. Instead, parts of the content (usually words, phrases or images) are linked with other content chunks.

– **Database schema:** The database schema is another way of organizing content chunks, which are, in this case, database records. Although more powerful representations such as entity-relationship models (cf. [CHEN76]) are used to model database schemas, the basic structure of a database schema can be represented as a relation on labels as Figure 12(d) illustrates. The labels in the database schema do not describe individual content chunks but types of content chunks, for example author names. Web sites that allow browsing a large number of similar structured information chunks follow the database content organization if there is a fixed linking schema (e.g. if product summary pages always link a product gallery and a product detail page on a shopping site, or movie pages always link pages about the featured actors on a movie database site).

Often, multiple COs are used to organize the contents of a Web site. For example, the high-level organization of a page might be hierarchical but some sections might follow the database schema. In addition, an associative schema might be used link related pages.

### 2.2.6 Discussion

We aim at clarifying that hierarchies have special characteristics and are useful for a broad range of applications because they carry meaning. But meaning is attributed by humans and, thus, humans come into play. Human interpretation is fuzzy and defining a model that includes aspects of human interpretation and that is simple enough to be applicable in the context of this thesis is challenging. To solve this problem, our definitions are based on the following assumptions:

- The focus is not on a detailed and accurate model of human interpretation but on a detailed and accurate model of the design aspects of interactive media. We model human interpretation in a naïve and straightforward fashion, without considering current research in the fields of cognitive sciences or neurolinguistics. Instead, the focus is on clarifying the nature of the design aspects. For example, *mental models* have been discussed in depth in several fields of psychology [GEST83]. But in this thesis, the term can be understood in a literal sense, without the psychological background – as a mental representation of a real-world entity. Whether human thinking and reasoning is really based on mental models is irrelevant in the context of this thesis. What matters is that the idea of mental models can be utilized to define our model.

- The focus is on clarifying the nature of the three design aspects and not on providing a method for discerning whether a given object is a valid instance of the $O^3$-model in the individual case. For example, we define a *label* as a symbol on whose meaning human interpreters agree. Because a label means the same thing to different individuals, it can be utilized as a vehicle for communication. This characteristic explains the nature of labels but does not allow discerning labels from symbols that are not labels in non-obvious cases. This is because a label will never mean exactly the same thing to two different individuals. It is also unclear how to determine the overlap of meaning for two individuals and a given label. However, an empirical test could be developed and practical thresholds could be identified – but this is beyond the scope of this thesis.

- We assume an idealized perspective and abstract from usability flaws. We illustrate this, again, by the example of the term label. We assume that a label is a symbol that has an agreed-upon meaning (cf. section above) and that all link names are labels. In practice, this is not always the case. Some link names might be misleading and difficult to interpret. But we can assume that rational designers strive to use symbols as link names that fall under the proposed definition of a label in order to maximize usability. Thus, in practice, link names typically are labels, with few exceptions due to usability problems. Therefore, it is reasonable to simplify and assume the absence of usability problems.

- We assume a homogeneous target audience. Some of the proposed term definitions rely on the distinction whether two individuals agree on the interpretation of a thing. But given a group of people, some pairs of individuals might have a shared understanding while others have not. However, we disregard these cases. This simplification does not limit the applicability of the $O^3$-model because it is in accordance with the design objectives. For example, all target users should be able to understand a link name, thus, only symbols should be used on whose meaning all target users agree, i.e. labels. If it is not possible to find labels that fulfill this requirement, the target audience must be divided and separate labels must be specified for each group. This is necessary, e.g., if the target audience includes an English speaking community as well as a Chinese speaking community.

## 2.3 The Implementation Process: Navigation Design Patterns

Although a hierarchical CO must have certain properties for being suitable in the basic browsing scenario (cf. previous section), such a tree structure alone would not represent an appropriate FO in the eyes of a rational designer. If the edges of the CO would be mapped directly onto hyperlinks, users would only be able to navigate downward in this tree structure. Users should at least be able to return to the homepage (root node) from all other pages. Typically, there is not only a link to the homepage but to all pages of the first hierarchy level on each page and to the siblings of a page, too. However, many different variations are possible (cf. examples in Figure 13). In all these examples, the edges extending the tree structure can be described by few simple, schematic rules. We can assume that a rational designer will act in a very similar way by not making design decisions involving individual links but by making design decision involving general transformation patterns. In fact, the benefit of using common design patterns for implementing Web navigation is widely agreed-upon in academic and practical literature.

We can consider navigation design patterns as patterns that, on the one hand, define how the CO is translated into the FO and, on the other hand, determine certain aspects of the PO[5]. For example, a typical pattern specifies that from each page in a hierarchical CO, the ancestors, the siblings and the child pages are accessible. In fact, considering navigation design patterns as transformation rules is a new perspective on a well-known concept. This alternative point of view allows are more systematic pattern analysis and classification.

### 2.3.1 Navigation Design Patterns: An Established Concept

Patterns for designing Web navigation have been widely discussed in academic and practical literature, although they have not yet been described as rules transforming CO structures into FOs. Inspired by the concept of software design patterns for object-oriented programming languages, academia brought up the idea of patterns for hypermedia design in the late 1990s. Design patterns are well-understood and well-documented solutions for common design problems. When adopting design patterns in the area of hypermedia development, not only the idea of providing templates to solve recurrent problems played a role. Design patterns also provide a shared vocabulary within development teams and improve communication [ROSG97]. Rossi et al. [ROSG97] were among the first to introduce hypermedia design patterns in the year 1997 and in the following years a number of design patterns were proposed and described by different authors. Germán and Cowan [GECO00]

[5] In the following, we will focus only on the aspects of the PO that are related to the interactive elements, i.e. spatial arrangement and visual presentation of the hyperlinks not on the design of the entire page.

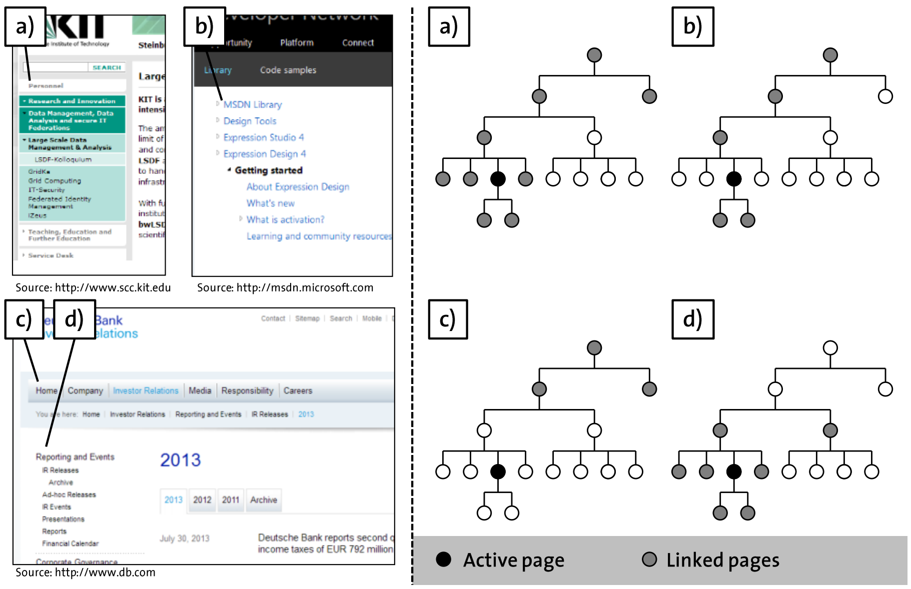

a)
Source: http://www.scc.kit.edu
b)
Source: http://msdn.microsoft.com
c)
d)
Source: http://www.db.com
a)
b)
c)
d)
● Active page
○ Linked pages

**Figure 13. Examples for different patterns of menu behavior. a) The menu contains links to the child pages, the siblings, the parent and all first level pages. The siblings of the second level are collapsed. b) The menu contains links to the child nodes and all ancestors. Siblings of ancestors are collapsed. c) The menu shows the first level of the hierarchy. d) The menu contains links to the child pages, the siblings of the active page, the parent and the siblings of the parent.**

made an attempt to list all proposed patterns in the year 2000 and gathered more than 50 patterns.

Although research interest has moved away from hypermedia design patterns, they are an integral part of Web design today. Many practical books on Web user interface development contain collections of hypermedia design patterns (e.g. [MORO06], [KALB07]). The main reason, why Web design heavily relies on design patterns today was not discussed in the early academic works on this topic: Not only designers benefit from design patterns but also users. Over the years a set of design patterns has evolved as de-facto standards, e.g. breadcrumbs, tag clouds or navigation tabs (see [TOXB13] for a description of these patterns). Those patterns are solutions that are well-known, not only by designers but also by users (although not necessarily by name). Reusing known patterns is essential in Web design because, as Garrett remarks [GARR11], there are no instruction manuals or training seminars for Web sites.

The academic design patterns gathered, e.g., in [GECO00] as well as more recent collections (e.g. [TOXB13]) by practitioners list not only navigation-related patterns but also other kind of patterns, for instance, related to forms (autocomplete, captcha, etc.). In this thesis, we are only interested in hypermedia design patterns related to navigation, hence, we use the more specific term navigation design patterns.

### 2.3.2 Categorizing Navigation Design Patterns

Integrating the concept of navigation design patterns into the O$^3$-model does not only explain the interrelation between the CO, FO and PO, it also provides a systematic way of characterizing and distinguishing navigation design patterns.

Existing catalogues of navigation design patterns do not agree, which concepts should be considered as patterns and which should not. They differ in granularity and often, a pattern that can be found in one catalogue either overlaps with a different pattern from another catalogue or has no corresponding pattern. For example, a horizontal bar representing the first level of the hierarchy (the main menu) most closely matches the general pattern "navigation bar" in [KALB07], which includes any "horizontal chain of plain hypertext links"[6]. In contrast, the corresponding pattern from [GARR11] would be the "global navigation", which contains the "key access points"[7], while the catalogue from [TOXB13] does not list a matching pattern. These inconsistencies demonstrate that there is no systematic methodology for classifying navigation design patterns. This prevents consolidating existing catalogues or conducting sound empirical studies in this field.

The O$^3$-model provides such a methodology. We can use three dimensions to characterize navigation design patterns and to clarify which patterns should be distinguished: the underlying organizational schema of the CO, the implications of the pattern on the FO and its implications on the PO. We propose that a navigation design pattern should be defined by describing its unique characteristics regarding each dimension. Two navigation design patterns should be regarded as distinct if they differ in at least one dimension. For example, the horizontal main menu should be regarded as distinct navigation design pattern, which can be specified as follows:

| *Navigation Design Pattern:* | **Horizontal main menu** |
|---|---|
| *CO-dimension:* | Based on a hierarchical CO |
| *FO-dimension:* | The pattern generates links to the first level of the hierarchy |
| *PO-dimension:* | Links are arranged horizontally at the top of each page |

In a similar way, we can distinguish **vertical main menus**, **horizontal local menus** (representing any other than the first hierarchy level) and **vertical local menus**.

Figure 6(b) and Figure 6(c) on page 14 show examples of another pattern, commonly called "pagination" that can be defined as follows:

[6] Kalbach [KALB07] uses the term navigation mechanisms, which closely resembles the concept of navigation design patterns.

[7] Garrett [GARR11] refers to navigation design patters as navigation systems.

| *Navigation Design Pattern* | **Pagination** |
|---|---|
| *CO-dimension* | Based on a sequential CO |
| *FO-dimension* | The pattern generates links to a fixed number of direct predecessors and direct successors of the active element in the sequence. Optionally, a link to the first and last element of the sequence is provided. |
| *PO-dimension* | Links are arranged horizontally above or below the presented content chunk. |

The focus of this section is not on providing a complete list of navigation design patterns, but on clarifying the central concepts of this thesis. A design pattern that we will consider later in this thesis is commonly called breadcrumb navigation (e.g. in [KALB07] or [TOXB13]). It corresponds to the pattern referred to as active reference in the more academic publications (cf. [GECO00]). Using the proposed methodology we define a breadcrumb navigation as follows:

| *Navigation Design Pattern* | **Breadcrumb navigation** |
|---|---|
| *CO-dimension* | Based on a hierarchical CO |
| *FO-dimension* | The pattern generates links to all ancestor pages in the hierarchical CO, i.e. pages on the path from the root node to the active page |
| *PO-dimension* | Links are arranged horizontally above the presented content chunk. Separator symbols between the links indicate breadcrumb trails. Common separator symbols are > and \| |

Breadcrumb navigations indicate the current position in the CO to the user (cf. Figure 14). If they can be extracted automatically, the hierarchy of the entire site can be recovered.

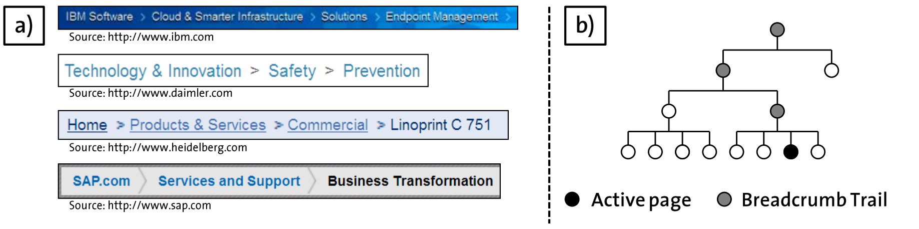

a)
IBM Software > Cloud & Smarter Infrastructure > Solutions > Endpoint Management >
Source: http://www.ibm.com
Technology & Innovation > Safety > Prevention
Source: http://www.daimler.com
Home > Products & Services > Commercial > Linoprint C 751
Source: http://www.heidelberg.com
SAP.com Services and Support Business Transformation
Source: http://www.sap.com
b)
● Active page
● Breadcrumb Trail

**Figure 14. (a) Breadcrumb trails are a common navigation design pattern that (b) represents the path to the active page in a hierarchical CO.**

### 2.3.3 Navigational Blocks and Navigation Elements

If a designer decides to apply a certain navigation design pattern, he implicitly specifies hyperlinks between the pages of the site. The hyperlinks that belong to the same navigation design pattern usually form an adjacent block on each page that can be distinguished visually from the surrounding content. This is due to the fact that navigation design patterns are not only utilized by designers but also by users because they are familiar with these concepts. Page blocks resulting from navigation design pattern will be referred to as navigational blocks:

***Definition 2-7 (Navigational block).*** *A page block that contains no content chunks but only hyperlinks is called navigational block if the hyperlinks result from the same navigation design pattern and the block contains all hyperlinks from this navigation design pattern.*

A navigational block is a code snippet that corresponds to a navigation design pattern. All pages from the same Web site usually have a similar FO and PO (cf. Section 2.1.1) and, usually, only a small number of sample pages are visually modeled by designers, using dedicated graphic-design software. These mockups are used as templates for generating all pages of the site. Hence, a navigation design pattern that is specified once within a template is often applied to a multitude of pages. This is in accordance with usability requirements because users are not forced to reorient themselves after each page transition (cf. transitional volatility [DANI03]). After a page transition, users should find the same navigation design patterns being implemented at similar positions on a page. From the client-perspective, a navigation design pattern is physically manifested as all code snippets, i.e. navigational blocks that result from this pattern. This set of navigational blocks will be referred to as navigation element:

***Term definition 2-8 (Navigation element).*** *The implementation of a navigation design pattern is called navigation element. From the client-perspective, a navigation element consists of the set of navigational blocks from different pages that result from the same navigation design pattern.*

We can classify navigation elements and navigational blocks based on the navigation design patterns from which they result:

***Term definition 2-9 (Type of a navigation element / navigational block).*** *The type of a navigation element or a navigational block is given by the used navigation design pattern.*

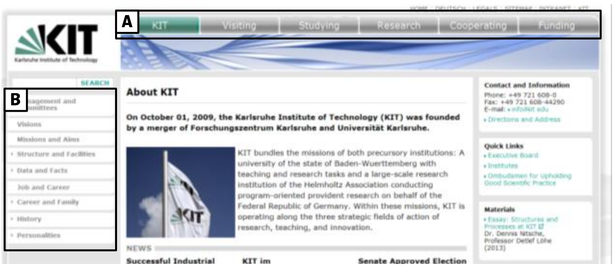
# 3 Problem Statement

We will refer to the problem of automatically extracting the CO based on the FO and PO, which is introduced in this section, as CO-mining. In Section 3.1 we explain, how CO-mining, on the one hand, can improve existing applications and, on the other hand, can enable new applications. In Section 3.2, we present the main methodological challenges, which might have hindered research on this subject previously and explain how we were dealing with these challenges. In Section 3.3, we discuss the technical challenges in more detail by analyzing the sub-tasks that are necessary in order to mine the CO.

## 3.1 CO-Mining Applications

If the CO can be automatically extracted from Web sites, applications in different fields will benefit. These fields include (1) automated sitemap generation and reverse engineering, (2) search result presentation and ranking, (3) Web analytics, (4) focused crawling (5) contextual advertising and (6) Web site transcoding for mobile applications. In this section, we explain how CO-information can be utilized in these fields. The utilization of CO information has been discussed in most of these areas before and we pick up existing ideas. An exception is the field of search result presentation, for which we contribute novel ideas of utilizing CO-information in Chapter 7.

**(1) Automated sitemap generation and reverse engineering:** Most Web sites today are based on a content management system (CMS). Content management is the process of collecting, managing and publishing contents [BOIK05]. Hence, CMSs are systems that support these tasks, e.g. by providing tools for non-technical editing, which allow users to contribute contents without requiring special technical skills. In other words, most Web sites are built for frequently changing contents. Thus, the original information architecture evolves over time. Since Web sites often consist of thousands or ten-thousands of pages, the global information architecture may become more and more unclear and unmanageable in this process, even if the microscopic information architecture, i.e. the appearance of individual pages is well-designed and maintained. In theory, the CMS should provide an overview over the global information architecture. This is, however, not always the case, e.g., if the underlying CMS is not well-adapted to the application or different data-sources are involved. As a result, resource-consuming reverse engineering of the information architecture is necessary prior to redesigning and/or re-engineering the site. With respect to the $O^3$-model, this means, first of all, to generate a model of the CO. Currently, this can only be achieved



About KIT

On October 01, 2009, the Karlsruhe Institute of Technology (KIT) was founded by a merger of Forschungszentrum Karlsruhe and Universität Karlsruhe.

KIT bundles the missions of both precursory institutions: A university of the state of Baden-Wuerttemberg with teaching and research tasks and a large-scale research institution of the Helmholtz Association conducting program-oriented provident research on behalf of the Federal Republic of Germany. Within these missions, KIT is operating along the three strategic fields of action of research, teaching, and innovation.

NEWS

Successful Industrial KIT im Senate Approved Election

Contact and Information
Phone: +49 721 608-0
Fax: +49 721 608-44290
E-mail: info@kit.edu
Directions and Address

Quick Links
Executive Board
Institutes
Ombudsmen for Upholding Good Scientific Practice

Materials
Essay: Structures and Processes at KIT
Dr. Dennis Nitsche, Professor Detlef Löhe (2013)

Figure 15. A sample page from the English KIT Web site (source: http://www.kit.edu/kit/english/index.php). The (A) first level of the hierarchy and (B) the second level can clearly be identified. The first level consists of 6 items.

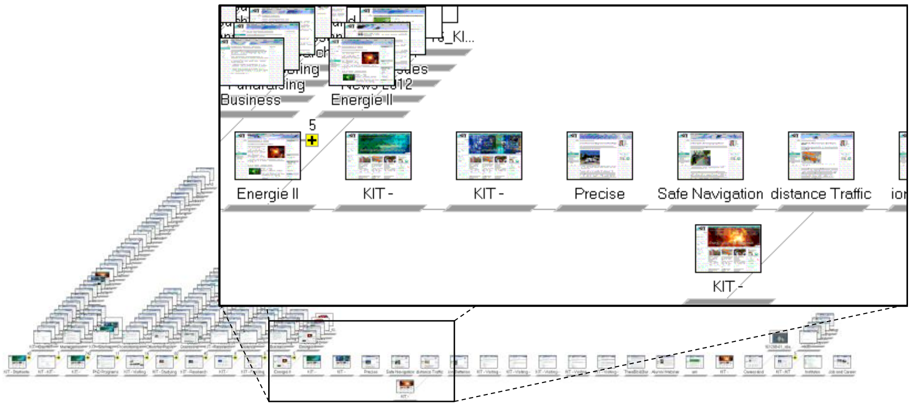
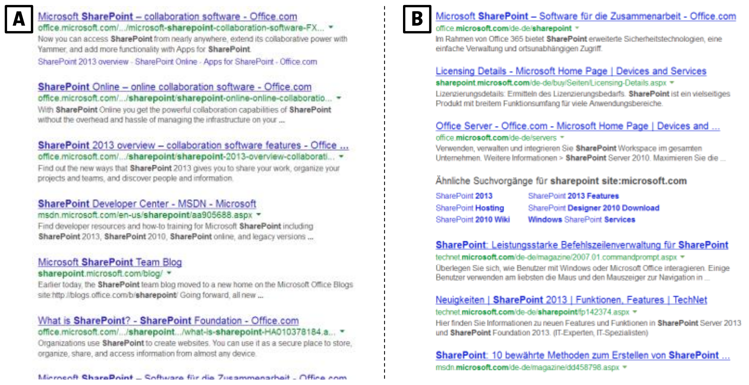


Figure 16. The sitemap automatically generated by the commercial tool Powermapper (Vers. 5.11, www.powermapper.com) does not reproduce the original CO correctly. For example, the first level of the generated sitemap consists of 26 items instead of 6.

through time-consuming manual work, since no automated approaches for mapping the CO exist. If a Web site is dominated by a single hierarchical CO, a model of this CO, e.g., a graph or list, is commonly called *sitemap*. Up until now, it is not possible to automatically mine sitemaps with satisfying precision, although some commercial tools exist.

PowerMapper seems to be the leading tool[8], but it is not able to reproduce the CO as perceived by humans (cf. Figure 15 and Figure 16)[9]. Previous academic works on automatic sitemap extraction achieve only low accuracies as well (e.g. [LICC11], see also Section 4.5). More precise approaches for mining the CO, or at least for mining certain aspects of the CO, will lower the costs for reverse engineering the information architecture and facilitate the redesign of large, grown Web sites.

**(2) Search result presentation and ranking:** Web search engines maintain an index of crawled Web sites. If a user submits a search query, the search engine returns an ordered list of the best-matching results for this query. But the COs of the indexed Web sites are not available to current search engines and, hence, they cannot be evaluated in order to enhance the ranking order and the result presentation. For example, search results from the same site are presented as flat lists (cf. Figure 17). Information about the site section from which the results originate or the relationships of the listed pages (e.g., whether a page is a child page of another) is not presented to the user. Another limitation of this kind of presentation is that it does not consider the fact that a user's information need often cannot be satisfied by a single page alone. Users that click on a search result often continue to navigate on the target site (e.g. [WHHU10]). But users cannot anticipate from the result presentations provided by current search engines, whether a search result is a good starting point for further exploration. The integration of CO-information, especially the utilization of hierarchical COs, would allow several improvements:


A

Microsoft **SharePoint** – collaboration software - Office.com
office.microsoft.com/.../microsoft-**sharepoint**-collaboration-software-FX...
Now you can access **SharePoint** from nearly anywhere, extend its collaborative power with Yammer, and add more functionality with Apps for **SharePoint**.
SharePoint 2013 overview - SharePoint Online - Apps for SharePoint - Office.com

**SharePoint** Online – online collaboration software - Office.com
office.microsoft.com/.../**sharepoint/sharepoint**-online-online-collaboratio...
With **SharePoint** Online you get the powerful collaboration capabilities of **SharePoint** without the overhead and hassle of managing the infrastructure on your ...

**SharePoint** 2013 overview – collaboration software features - Office ...
office.microsoft.com/.../**sharepoint/sharepoint**-2013-overview-collaborati...
Find out the new ways that **SharePoint** 2013 gives you to share your work, organize your projects and teams, and discover people and information.

**SharePoint** Developer Center - MSDN - Microsoft
msdn.microsoft.com/en-us/**sharepoint**/aa905688.aspx
Find developer resources and how-to training for Microsoft **SharePoint** including **SharePoint** 2013, **SharePoint** 2010, **SharePoint** online, and legacy versions ...

Microsoft **SharePoint** Team Blog
**sharepoint**.microsoft.com/blog/
Earlier today, the **SharePoint** team blog moved to a new home on the Microsoft Office Blogs site:http://blogs.office.com/b/**sharepoint**/ Going forward, all new ...

What is **SharePoint**? - **SharePoint** Foundation - Office.com
office.microsoft.com/.../**sharepoint**.../what-is-**sharepoint**-HA010378184.a...
Organizations use **SharePoint** to create websites. You can use it as a secure place to store, organize, share, and access information from almost any device.

Microsoft **SharePoint** – Software für die Zusammenarbeit - Office.com

B

Microsoft **SharePoint** – Software für die Zusammenarbeit - Office.com
office.**microsoft.com**/de-de/**sharepoint**
Im Rahmen von Office 365 bietet **SharePoint** erweiterte Sicherheitstechnologien, eine einfache Verwaltung und ortsunabhängigen Zugriff.

Licensing Details - Microsoft Home Page | Devices and Services
**sharepoint.microsoft.com**/de-de/buy/Seiten/Licensing-Details.aspx
Lizenzierungsdetails: Ermitteln des Lizenzierungsbedarfs. **SharePoint** ist ein vielseitiges Produkt mit breitem Funktionsumfang für viele Anwendungsbereiche.

Office Server - Office.com - Microsoft Home Page | Devices and ...
office.**microsoft.com**/de-de/servers
Verwenden, verwalten und integrieren Sie **SharePoint** Workspace im gesamten Unternehmen. Weitere Informationen > **SharePoint** Server 2010. Maximieren Sie die ...

Ähnliche Suchvorgänge für **sharepoint site:microsoft.com**
SharePoint **2013** | SharePoint **2013 Features**
SharePoint **Hosting** | SharePoint **Designer 2010 Download**
SharePoint **2010 Wiki** | **Windows** SharePoint **Services**

**SharePoint**: Leistungsstarke Befehlszeilenverwaltung für **SharePoint**
technet.**microsoft.com**/de-de/magazine/2007.01.commandprompt.aspx
Überlegen Sie sich, wie Benutzer mit Windows oder Microsoft Office interagieren. Einige Benutzer verwenden am liebsten die Maus und den Mauszeiger zur Navigation in ...

Neuigkeiten | **SharePoint** 2013 | Funktionen, Features | TechNet
technet.**microsoft.com**/de-de/**sharepoint**/fp142374.aspx
Hier finden Sie Informationen zu neuen Features und Funktionen in **SharePoint** Server 2013 und **SharePoint** Foundation 2013. (IT-Experten, IT-Spezialisten)

**SharePoint**: 10 bewährte Methoden zum Erstellen von **SharePoint** ...
msdn.**microsoft.com**/de-de/magazine/dd458798.aspx

Figure 17. Search results from the same site are presented as a flat list by the most popular search engines, e.g. (A) google.com or (B) bing.com. Since the COs of the sites are not considered by the search engines, it is not obvious to the users from which section of the site these results originate or how the pages relate, e.g., which page is a child page of another.

*First*, pages that originate from the same site could be aggregated. If, e.g., several pages from a site section match the query, the shared parent page could be returned as search result instead of showing each individual page. The result list would be condensed and more clearly arranged. *Second*, the original hierarchical structure could be depicted in the search result presentation, i.e. the search results could be arranged not as a list but as a tree. This would allow the users to get an idea of the global organization of the target site. *Third*, hierarchy information could be used to rank search results. For example, a result page that is an ancestor of another result page in a hierarchical CO could be placed more prominently. *Fourth*, the summary of a linked search result page could be extended by the page's child nodes. As we show in Chapter 7, this would allow users to anticipate, whether it is worth visiting a search result or not.

**(3) Web analytics:** User interactions with Web sites can directly be observed and monitored by the vendor. HTTP-requests logged by the Web server already provide enough information to gain valuable insights into the Web site usage. In addition, client-based technologies such as tracking cookies can be applied to gather more detailed and precise information. The process of analyzing the collected usage data in order to exploit this feedback information for improving Web sites is called Web analytics. A key method in the field of Web analytics is the analysis of clickstreams. The term clickstream denotes the trail a user leaves in the log files, or, more precisely, the ordered list of visited pages and the dwell time on each page. Through the log files, thousands or ten-thousands of clickstreams are usually available and in order to be interpreted, they must be aggregated. Interpretation is relatively simple if the ultimate goal is to maximize the number of transactions and if it is possible to identify consecutive steps (each of which is represented by a page) that lead to such transactions. Based on the clickstream data, the percentages of users that proceeded to the next step can be calculated and visualized as illustrated in Figure 18. State of the art tools such as Google Analytics[10] are able to generate similar visualizations for live data. The percentage of users that proceed from one step to another towards the transaction goal is called the conversion rate. Conversion rates can provide important insights. For instance, the conversion rate between the last two steps in Figure 18 is very low, which means that most of the users cancel the checkout process. The shop vendor can now apply different changes to the UI in order to track which changes have a positive influence on the conversion rate.

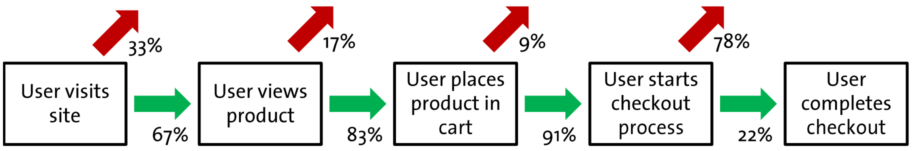


**Figure 18. Conversion rates between different steps towards a successful transaction can be measured and visualized.**

[10] http://www.google.com/analytics/

However, this method is not applicable for Web sites that are not transaction-based but have an informational focus because it is not possible to identify a sequence of navigational steps that represent a successful transaction or site visit. The analysis of dwell times is also difficult, because a short stay on a site might indicate either that a user quickly realized that he will not find what he is looking for or the opposite – which is that a user was able to find the information he was seeking for in a short time. Aligning clickstream information with mined COs would allow interpreting the user behavior on informational sites. By overlaying the CO with clickstreams it could be identified, for example, whether users quickly find a straight path to information they are seeking or whether they are lost on the site.

**(4) Focused Crawling:** Due to the rapid growth of the WWW and the enormous number Web documents, only few industrial Web crawlers that feed the major search engines such as Google.com or Bing.com are able to maintain an index that covers a significant proportion of today's Web. Other dedicated industrial crawling solutions or academic Web crawlers with limited resources are only able to retrieve and analyze small fractions of the Web. Which fraction of the Web the crawler downloads is crucial in most cases. Chakrabarti et al. [CHBD99] developed the concept of focused Web crawlers. The crawling strategy of focused Web crawelers aims at downloading only documents that are relevant to a predefined topic. The topic is usually specified by providing sample documents. Whether a document is relevant for a given topic can only be verified after the document was downloaded. Thus, the number of irrelevant documents among the downloaded documents must be minimized in order to increase the efficiency of a focused crawler, or, in other words, a focused crawler must be able to predict whether a document is relevant before downloading it. Once relevant pages are found on a Web site, information about the CO of the site can help Web crawlers to proceed to other relevant pages. Ying et al. [YZYH12] have proposed a focused Web crawler that is based on breadcrumb navigations (cf. Section 2.3.2). Breadcrumb navigations allow deriving the hierarchical COs (called "semantic trees" by Ying et al.). The idea is that each subtree of a CO contains pages with similar topics. Thus, once a relevant page was found, it is likely that other relevant pages can be found in the same subtree. In a first step, the crawler retrieves sample pages to determine relevant subtrees and then only those subtrees are downloaded. However, Ying et al. evaluated the method only for a single site and did not consider the problem of automatically extracting hierarchical COs or breadcrumb navigations – a necessary prerequisite for the deployability of their method. Furche et al. [FGKS12] argue that sequential COs can also be utilized to direct focused crawlers. In case of long sequences such as the results of a Web search or a product search, the results are paginated. If a crawler is interested in the entire result list but not in the entire content of the site, an efficient focused crawler must be able to decode the sequential CO in order to traverse it.

**(5) Contextual advertising:** Contextual advertising is the source of revenue of many content providers in today's Web and an important factor in the digital economy. Contextual advertising means that Web pages are enriched with advertisements that fit the content. For example, an advertisement of a car company might be displayed in conjunction with a news article about a Formula One race. This approach aims at increasing the efficiency, i.e. at leading to more clicks per ad. To apply this concept on a large scale, the advertisements must be matched automatically with the content. Thus, the advertiser usually provides target keywords for his campaign, which can be matched with page keywords to place ads on topically related pages. However, if no or too few keywords are assigned to a page, the approach cannot be applied. To achieve a higher keyword coverage and improve the efficiency of contextual advertising, Kumar et al. [GLPG11] utilized the hierarchical CO of a site. Their approach is based on the idea that subtrees contain topically similar pages and thus, keywords can be propagated buttom-up and top-down along the edges of the CO. However, Kumar et al. rely on the URL folder hierarchy as a model for the hierarchical CO. The URL folder hierarchy often induces a tree that approximates the hierarchical CO, but this does not apply to the general case [KUPT06]. More advanced methods of CO-mining will overcome this this limitation and make CO-based keyword enrichment applicable to more Web sites in practice.

**Website transcoding for mobile clients:** While Web standards such as HTML and CSS were developed from the beginning to be adaptive to different screen sizes and resolutions, the concept of resizable layouts has its limits if the range of target resolutions is too broad. On the one hand, screen sizes and average resolutions have dramatically increased [NEMN11] while on the other hand, mobile devices with small screen sizes are more and more used as Web clients. Maximizing the usability for both scenarios cannot be achieved by simply resizing the page elements in most cases. As a result, the contents which fit on one large screen must be split up into multiple pages to be easily accessible on small screens or, at least, they should be rearranged. A few years ago, most Web sites were only optimized for large screens ([HYCD09], [HHMS07]) but today, many vendors provide two separate versions, one optimized for PCs and one optimized for mobile devices. To avoid the costs and effort of maintaining two site versions in parallel, research has focused on automatically transcoding PC-optimized sites for mobile devices. Those approaches focus on the challenge of automatically distinguishing different page blocks and determining their role and importance (e.g. [CZSZ01, HHMS07, YASH09]). But to generate mobile sites automatically that resemble manually coded sites in terms of quality and usability, navigation mechanisms must be analyzed and tailored to the use on mobile devices, too. This implies the necessity of CO-mining, because the site structure must be extracted in order to generate appropriate navigation elements. In other words, currently, the unsolved problem of CO mining sets the limits for automatically transcoding sites for mobile devices.

## 3.2 Methodological Challenges

Although many applications would benefit, CO mining has not yet attracted much interest from the research community and industry. Based on the $O^3$-model, we argue that there are two main methodological challenges that hindered research in this field:

- **Ground truth problem:** A CO represents a logical arrangement of labels (and the corresponding contents) as designed and perceived by humans. A corresponding data structure might be stored on the server to dynamically generate and manage navigation elements. But only the resulting navigational blocks and not the original CO data sources are available on the client-side. Thus, it is difficult to evaluate CO mining approaches at a large scale, because the original CO is locked in the content management database on the server-side.

  *To solve this problem, we rely on manually labeled data sets. This is the only way to test, whether extracted structures reflect the COs. We have developed dedicated tools that support human assessors and reduce the labeling effort. However, the manually generated data sets are too small to be used as training data for machine learning approaches if the features space is large. Thus, most of the presented mining approaches are rule-based.*

- **Visually encoded information:** Humans rely primarily on visual information, i.e. the presentational and structural layers of the PO, to derive information about the CO: (1) The presentational layer of the PO provides information about the navigation design pattern of a navigational block. For example, certain separator symbols between hyperlinks indicate breadcrumb navigations (cf. Figure 14). (2) The structural layer of the PO additionally supports the identification of navigation design patterns. For example, a breadcrumb trail is usually placed above the main content of the page. In the following Section 3.3, we will discuss in more detail the technical challenges that arise if we want to process visual information in order to automatically extract the CO. We conclude that vision-based approaches will be very difficult to implement in the near future given the current state of the art.

  *The approaches presented in this thesis are not vision-based and the presentational layer of the PO is not considered. According to the $O^3$-model, navigation design patterns determine not only parts of the PO but also the FO. We show that it is possible to decode the CO based on the FO and the structural layer of the PO. The structural layer information that we exploit can be derived directly from the HTML structure. The methods presented in this thesis require neither CSS information nor HTML rendering. Hence, our methods gain CO information in a different way than humans do. However, we demonstrate that the results correlate very well.*

## 3.3 Technical Challenges

For humans, the CO of a Web site is obvious because humans are able to interpret the navigational blocks and their relationships based on visual information. At the same time, automatically extracting this structural information is challenging. In this section, we will explain why and describe the necessary subtasks for retrieving the CO. For this, we analyze the cognitive steps humans conduct to gain information about the CO of a site. Again, the focus is not on analyzing human perception and reasoning but on decomposing the problem. Thus, our examinations might be naïve from the perspective of cognitive sciences, but nevertheless, they fulfill the purpose of deriving a basic mining process.

## 3.3.1 Basic CO-Mining Process

The CO is visually communicated to human users through the graphical appearance of the rendered page. Humans gain CO information from the displayed navigational blocks such as horizontal or vertical menus. (This is obvious because one can think of any hierarchically organized Web site and then imagine a page of this site with all other content removed except for the navigational blocks – then the CO-information is still available). Thus, navigational blocks and their types must be identified in order to extract the CO. This task is referred to as *navigational block classification* in this thesis. But before the blocks can be classified, their boundaries must be determined. Hence, the first step of the basic CO-mining process is the conduction of a page segmentation that resembles the human perception (cf. Figure 19). Both, page segmentation and page block classification are known problems of Web mining, for which no general solutions exist yet (see Chapter 4).

Once the navigational blocks are extracted and their types are identified, their semantics must be understood. For example, in order to recover the CO, the different levels of a hierarchy must be reassembled if each level is represented by a different menu. However, in our basic CO-Mining process, this is the fourth step (cf. Figure 19), not the third. An intermediate

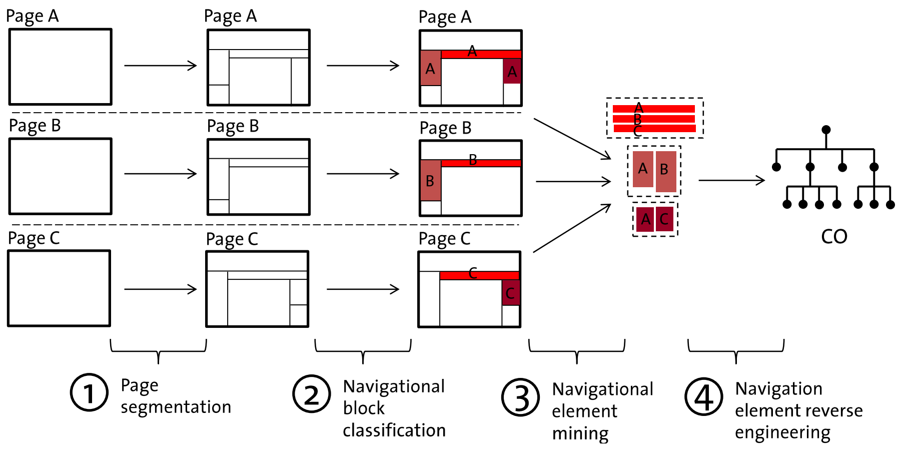

Page A
Page A
Page A
A
A
A
Page B
Page B
Page B
B
B
Page C
Page C
Page C
C
C
A
B
C
A
B
A
C
CO
1
Page segmentation
2
Navigational block classification
3
Navigational element mining
4
Navigation element reverse engineering

**Figure 19. The problem of extracting the CO can be divided into four subproblems: (1) Segmenting the page into visually distinct blocks, (2) identifying navigational blocks and the underlying navigation design pattern, (3) identifying navigational blocks that belong together and form navigation elements and (4) the interpretation of the navigation elements / navigation design patterns.**

step is necessary, because usually the navigational blocks of a page represent only fractions of the CO, e.g. in case of hierarchically organized sites, they show the subtree in which the active page is located (cf. Figure 13). But we are interested in the entire CO, and in order to retrieve it, not only the different levels but also the different subtrees must be put together. For this, the CO-information from different pages has to be merged. If we use the approach of reassembling the hierarchy levels on a per page basis first (cf. Figure 20(A)) and then merging the partial hierarchies, we have to ensure that the parts are assembled in the right way, i.e. that only subtrees are merged that really belong to the same hierarchy and that the subtrees are attached at the right positions. However, we can achieve the same result with a slightly different approach that makes the problem easier to handle: Instead of first extracting page-based partial hierarchies as depicted in Figure 20(A), we can first mine all appearances, i.e. navigational blocks, of a certain navigation element (Figure 20(B)), analyze them and then reassemble the global hierarchy. In case of hierarchical menus, one navigation element usually represents a single level of the global hierarchy. As illustrated in Figure 20, approach B) first extracts the hierarchy levels separately before the hierarchical relationships themselves are analyzed. Method A), in contrast, analyses the hierarchical relationships of the navigational blocks for each page individually before the partial hierarchies are put together. In this thesis, we assume a basic mining process that is based on method B), which means that navigational element mining is conducted before navigational element interpretation. This is reasonable since method B) brings an important advantage: We found that discovering the hierarchical relationships is easier if not conducted on a per-page level but on a per-site level because we can take more information into account.

The fours tasks will be discussed in more detail in the following sections.

## 3.3.2 Web Page Segmentation

Current Web pages can be thought of as graphical user interfaces (GUIs). User interactions are not solely based on textual information and textual input but on visual information and the manipulation of graphical symbols. As a result, human users have to distinguish page elements, i.e. page blocks, with different functionality based on visual signals in order to interact with a Web page. A user must be able, for example, to distinguish the main navigation, the local navigation, the page header and the main content of the page. If the main content consists of separate objects, e.g. a number of article teasers, the user also must be able to differentiate these objects. Thus, graphically clearly arranged and distinguishable page elements are an important requirement to ensure the usability of Web sites.

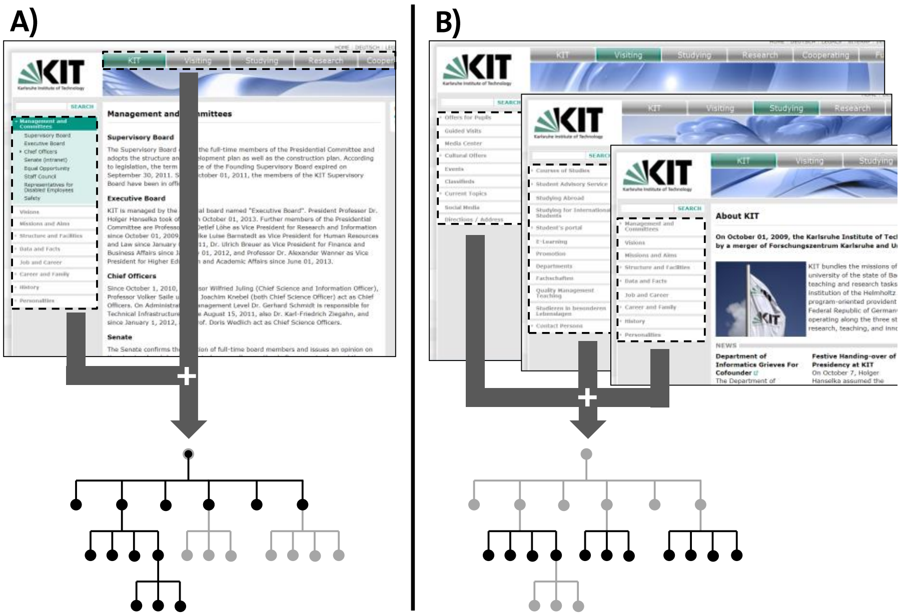


**Figure 20. There are two basic mining approaches: A) For all pages, the partial hierarchies are extracted first before the partial hierarchies are joined into the global hierarchy. B) The hierarchy levels can be joined first across all pages, before the hierarchical relationships are discovered on a site-wide basis. We found that approach B) is easier to implement.**

However, automated page segmentation by imitating the way humans discern page elements is a complex problem, for which a general solution is not in sight. Recognizing and distinguishing objects is regarded as an extremely difficult computational problem up until now [PICD08]. The human ability of discerning and identifying objects is the key subject of *Gestalt Theory* – a psychological school with roots in the beginning of the 20th century that still influences research in visual perception in general [PICD08] and automated Web page segmentation in particular, today [YASH09]. The idea behind Gestalt theory is that humans process visual input rather based on forms and figures than based on more fine-grained, fundamental visual characteristics such as dots, lines and colors. In other words, Gestalt theory assumes that our visual system has the "build-in" capability of recognizing objects. *"I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have '327'? No. I have sky, house, and trees."* Gestalt theory pioneer Max Wertheimer wrote in 1923 [WERT38]. We perceive objects even before we are aware of its individual parts and their visual properties.

As a result, Gestalt theory has focused researching the rules that are applied by the human visual system in order to discover closed forms and figures, i.e. grouping basic visual elements such as dots into more complex shapes. Among others, Wertheimer proposed the rules of similarity and proximity in [WERT38]. The rule of similarity says that objects with similar visual properties such as color, size or shape tend to appear as being grouped together. The rule of proximity postulates that objects that are located closer to each other compared to surrounding objects tend to be perceived as a group. Other rules have been proposed more recently, e.g. the rule of common region, which assumes that objects within the same bounded region tend to be perceived as a single, larger form [PALM92]. A machine that is able to conduct a page segmentation that comes close to the one perceived by humans must be able to apply this kind of rules. However, even in case of pages with little content and minimalistic design, humans seem to derive the page structure by applying combinations of different rules. In addition, although the grouping rules seem simple at first sight, each of it is fuzzy in practice and cannot be easily utilized in the field of computer vision. For example, consider the page depicted in Figure 21. Humans are able to distinguish e.g. the header, the menu and the different article teasers. We can identify to which text snippets the presented images belong. The resulting perceived page segmentation is outlined below the screenshot. The different blocks can be explained by Gestalt rules, which we have annotated. For example, we can associate the pictures with the appropriate text snippets because of the law of proximity. In the navigation bar at the bottom of the page, we can identify two blocks of hyperlinks that differ in color – the rule of similarity applies here. The header is confined by the boundaries of a black area, i.e. a common region.

The reason that automated visual page segmentation is so challenging is that the rules depend on the context and that they interfere. For example, there is no absolute distance at which the proximity rule starts to take effect. It depends on the distances to the surrounding objects and the entire page layout. Rules can also conflict or confirm each other. Close objects may be attributed to different groups, if they are placed inside areas with different backgrounds and so on. To derive an adequate page segmentation automatically, a holistic analysis is necessary that considers all complex interrelations. Although approaches for vision-based Web page segmentation exist (e.g. [CYWM03]), the results still differ from the human perception (cf. Section 4.3).

The page segmentation can also not be derived directly from the HTML structure. While the tree structure induced by the HTML-tags defines a hierarchical page segmentation, this structure is much more fine grained. In fact, the visual segmentation is contained within the HTML-induced segmentation because each block with distinct visual properties needs to be enclosed in separate HTML-tags to set those properties. However, the HTML elements that define visual blocks cannot be easily distinguished from other blocks.

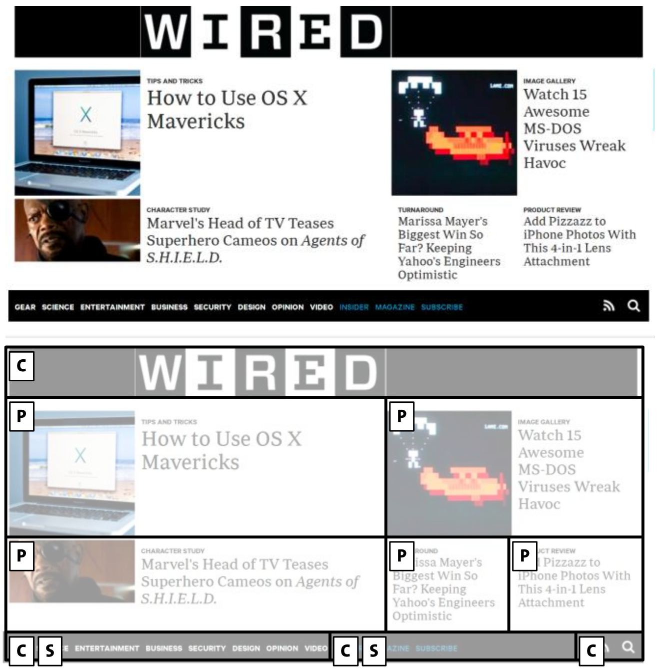

WIRED
TIPS AND TRICKS
How to Use OS X Mavericks
IMAGE GALLERY
Watch 15 Awesome MS-DOS Viruses Wreak Havoc
CHARACTER STUDY
Marvel's Head of TV Teases Superhero Cameos on Agents of S.H.I.E.L.D.
TURNAROUND
Marissa Mayer's Biggest Win So Far? Keeping Yahoo's Engineers Optimistic
PRODUCT REVIEW
Add Pizzazz to iPhone Photos With This 4-in-1 Lens Attachment
GEAR SCIENCE ENTERTAINMENT BUSINESS SECURITY DESIGN OPINION VIDEO INSIDER MAGAZINE SUBSCRIBE
C
P
P
P
P
P
C
S
C
S
C

Figure 21. The discussed Gestalt laws are intuitive, but when real-world documents are considered, the complexity becomes obvious: Even in this simple example, multiple Gestalt laws seem to take effect, e.g. law of similarity (S), law of proximity (P) and law of common region (C).

### 3.3.3 Navigational Block Classification

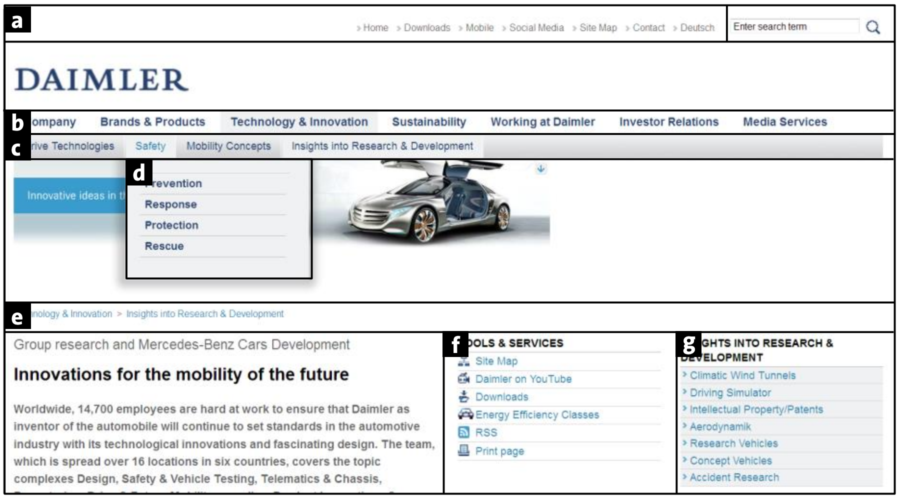
In the basic CO-mining process, the page segmentation task is followed by the navigational block classification task. At this stage, the page is already divided into blocks that resemble the different visual page elements as perceived by humans (cf. Figure 22). The objective is to identify the navigational blocks and to determine their types. For example, humans can distinguish several navigational blocks with different purposes on the page shown in Figure 22. Block (a) is a supplementary navigation that contains several shortcut links to key pages. Block (b) represents the main navigation (i.e. the first level of the global hierarchy), while the blocks (c) and (d) are implementing the second and third level of the hierarchy, respectively. Block (e) consists of a breadcrumb navigation that visualizes the current position in the global hierarchy. The blocks (f) and (g) result from contextual navigation elements, which display different hyperlinks in dependence of the page content.

The correct identification of navigational blocks and their roles is a prerequisite for human users to be able to interact with the Web site. We will now analyze, which attributes of navigational blocks indicate the block types to users, because these attributes must potentially be considered in computer-based classification, too. We will do this based on the example page shown in Figure 22. We are aware that the list of attributes might be incomplete and could be extended by the conduction of an empirical study. However, the observations based on the example page are sufficient to illustrate the complexity of the navigational block identification problem. We argue that at least the following attributes support the block classification by the users:

**Accentuation:** First, we can ask, why it is obvious to viewers of the page that not block (a) represents the main navigation but block (b). We can conclude that block (b) is more accentuated. In this case, the font size is larger, the font-style is bold and the font color is stronger. In general, the background color is another factor and also the contrast between the background and font color.

**Relative and absolute position:** Although the font size of block (b) is larger than the font size of block (c), it is not apparent, which of the blocks is more accentuated, because block (c) has a stronger background color in turn. Still, a user will not confuse the hierarchy levels because of the order in which the blocks appear on the page. Thus, relative position is another aspect. If both bars were located on the bottom of the page, it would be much harder to recognize them as the site's main navigation. Thus, absolute position matters, too.

**Characteristic visual symbols:** Because of the separator symbol ">" between the links of block (e), users can clearly identify the block as a breadcrumb navigation and can distinguish


a
› Home › Downloads › Mobile › Social Media › Site Map › Contact › Deutsch
Enter search term
DAIMLER
b
ompany Brands & Products Technology & Innovation Sustainability Working at Daimler Investor Relations Media Services
c
rive Technologies Safety Mobility Concepts Insights into Research & Development
Innovative ideas in t
d
revention
Response
Protection
Rescue
e
nology & Innovation > Insights into Research & Development
f
OLS & SERVICES
Site Map
Daimler on YouTube
Downloads
Energy Efficiency Classes
RSS
Print page
g
GHTS INTO RESEARCH & EVELOPMENT
› Climatic Wind Tunnels
› Driving Simulator
› Intellectual Property/Patents
› Aerodynamik
› Research Vehicles
› Concept Vehicles
› Accident Research

Group research and Mercedes-Benz Cars Development

Innovations for the mobility of the future

Worldwide, 14,700 employees are hard at work to ensure that Daimler as inventor of the automobile will continue to set standards in the automotive industry with its technological innovations and fascinating design. The team, which is spread over 16 locations in six countries, covers the topic complexes Design, Safety & Vehicle Testing, Telematics & Chassis,

**Figure 22. Navigational block classification aims at identifying the navigational blocks and their types, e.g. (a) supplementary navigation, (b) main navigation, (c) second level local navigation, (d) third level local navigation, (e) breadcrumb navigation, (f)+(g) contextual navigation.**

it from a potential fourth level navigation. The symbol “>” has evolved as a typical characteristic of breadcrumb navigations. Thus, visual symbols can play an important role in navigational block classification.

**Textual information:** On the presented sample page, the separator symbol “>” is not only used in block (e) but also in block (a). At first sight, a user could mistake block (a) for the breadcrumb navigation. However, the link names (e.g., “downloads”, “sitemap”, “contact”) indicate that block (a) fulfills a different purpose and serves as a supplementary navigation. Another example of textual information that has influence on the interpretation of navigational blocks is the caption of block (g). It indicates that the navigational block provides links to related content. Thus, textual contents can be considered by humans, too.

**Functional aspects:** If the link names in block (a) would not allow distinguishing the block from a breadcrumb navigation, a user can still learn the block type if he browses the site and visits other pages. The user would observe how the navigational blocks behave and would notice that the links in block (a) are not changing at page transitions. Thus, it cannot be a breadcrumb navigation. He would also see that block (e) changes in dependence of the active page. Thus, he learns that this page block represents a breadcrumb navigation.

Hence, a navigational block classification solution that imitates the way humans recognize navigational block types must consider at least all the listed aspects: font-size, font-style, font-color, background-color, absolute and relative position, visual symbols, textual information and even functional aspects. But the major challenge is that it is not possible to conduct the classification for each block in isolation. A holistic analysis of the entire page layout must be conducted. For example, if the navigational block (b) would not exist, users would assume that block (c) represents the first level of the hierarchy. Thus, the interpretation of a block can depend on the existence of other blocks. Even if block (b) would be on the page but the font size would be very small in comparison to block (c), users would take block (c) for the main navigation. Thus, the interpretation of a block can depend on the characteristics of other blocks. As a result, it is very difficult to decode how all these aspects intertwine. We also cannot easily apply machine learning methods, because it is very challenging if not impossible to find an appropriate set of features that captures all these aspects.

## 3.3.4 Navigation Element Mining

In order to extract the side-wide hierarchy, we must identify all navigational blocks from different pages that belong to the same navigation element. In other words, if a menu appears on multiple pages, we want to identify all its occurrences. Figure 23 shows a screenshots of two pages from the same site. A viewer can easily identify four navigation elements that are shared by both pages. Deciding whether two navigational blocks from different pages belong to the same navigation element automatically is the problem to which we refer to as navigation element mining. Even this seemingly simple task is challenging, because all straightforward solutions have its limits:

- **Comparing the navigational blocks' visual appearances:** Relying on the visual features such as colors, font-sizes, decorative elements, etc. for finding recurring navigation elements is prone to errors and requires an elaborated model. Pixel-wise comparison of two rendered blocks will not work because the hyperlinks that a navigational block displays can differ (cf. Figure 23). Also the font colors or background colors of a menu often change in dependence of the active page. Sometimes different page templates are used and the visual appearance of a navigation element unintentionally varies slightly. At the same time, different navigational elements might have a very similar visual appearance, e.g. the second and the third menu in Figure 23.

- **Matching the spatial position:** The position of a menu on the page can provide hints but it does not allow reliably identifying navigational blocks that belong to the same navigation element. For example, the position of the lower menu in Figure 23 differs because of additional content on the right-hand page.

- **Comparing HTML code:** Matching the code strings of two blocks to determine whether they belong to the same navigational element will also not work in general. There are several reasons, why the navigational blocks of one and the same navigational element often differ. First, depending on the page, different hyperlinks and fractions of the CO might be displayed. Second, certain attributes of the HTML-elements might be generated in dependence of the page, e.g. CSS class names. Third, different templates might be used and the code might differ, even if the visual presentation matches.

- **Comparing element ids:** To avoid the difficulties related to visual features, one could consider using identifiers assigned to HTML-elements in the DOM-tree to match navigational blocks. Usually, a specific identifier exists that is associated with all navigational blocks of a certain navigation element because such identifiers can be referenced in CSS-Code to attach dedicated presentational attributes. However, either the id-

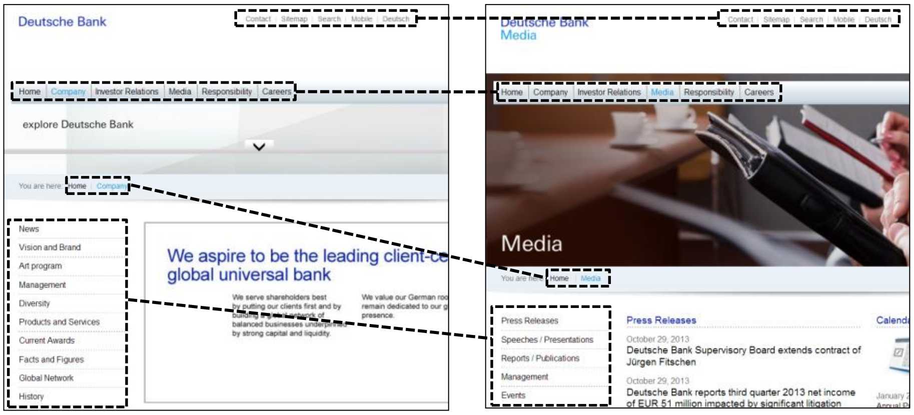

Deutsche Bank
Contact | Sitemap | Search | Mobile | Deutsch
Home | Company | Investor Relations | Media | Responsibility | Careers
explore Deutsche Bank
You are here: Home | Company
News
Vision and Brand
Art program
Management
Diversity
Products and Services
Current Awards
Facts and Figures
Global Network
History
We aspire to be the leading client-ce
global universal bank
Deutsche Bank
Media
Contact | Sitemap | Search | Mobile | Deutsch
Home | Company | Investor Relations | Media | Responsibility | Careers
Media
You are here: Home | Media
Press Releases
Speeches / Presentations
Reports / Publications
Management
Events
Press Releases
October 29, 2013
Deutsche Bank Supervisory Board extends contract of Jürgen Fitschen
October 29, 2013
Deutsche Bank reports third quarter 2013 net income of EUR 51 million impacted by significant litigation
Calenda

Figure 23. Navigation elements that are shared by two pages of the same site.

attribute or the class-attribute can both be used to define such identifiers. The identifiers could also be attached not directly to a navigational block but to a parent element. It is not evident from the source code, whether an identifier denotes all blocks of a navigational element or whether it fulfills another purpose.

- **Matching the DOM-tree position:** Aligning the DOM-trees of two pages and matching navigational blocks that are located at the same position in the trees does also not deliver reliable information. As Figure 23 illustrates, the page layout may differ and so does the structure of the DOM-tree. A navigational block with a different visual appearance that human users will not consider as the same menu might also be located at the same position in the DOM-tree on another page.
- **Comparing the hyperlink overlap:** Comparing the hyperlink-URLs of two navigational blocks does not reliably indicate whether the blocks belong to the same navigational element because the hyperlinks often change in dependence of the active subtree (cf. Figure 23)

A reliable navigation element mining solution must be based on a combination of multiple features. But how to combine and weight the features is a challenging problem on its own that has not yet been solved in previous work (cf. Section 4.4).

### 3.3.5 Navigation Element Reverse Engineering

To solve the problem of CO mining, it is not sufficient to mine navigation elements reliably and to determine their types. The internal structure of the navigation elements must be understood and analyzed. Often, the state of a navigational element differs in dependence of the active page. In other words, the links in the navigational blocks that belong to the same navigational element are varying. This is the case, for instance, if the state of the navigational block represents not the entire hierarchy but only the active subtree. Then, all partial trees must be joined accurately to retrieve the global hierarchy. But even extracting the partial tree represented by an individual navigational block is challenging because the tree structure of the underlying code diverges from the content structure. Again, the visual structure is crucial and the challenges of analyzing it described in the previous sections apply here, too. Sometimes, even visual clues are not sufficient and the internal structure of a menu can only be understood if the menu behavior is observed or the label semantics are considered as Figure 24 illustrates.

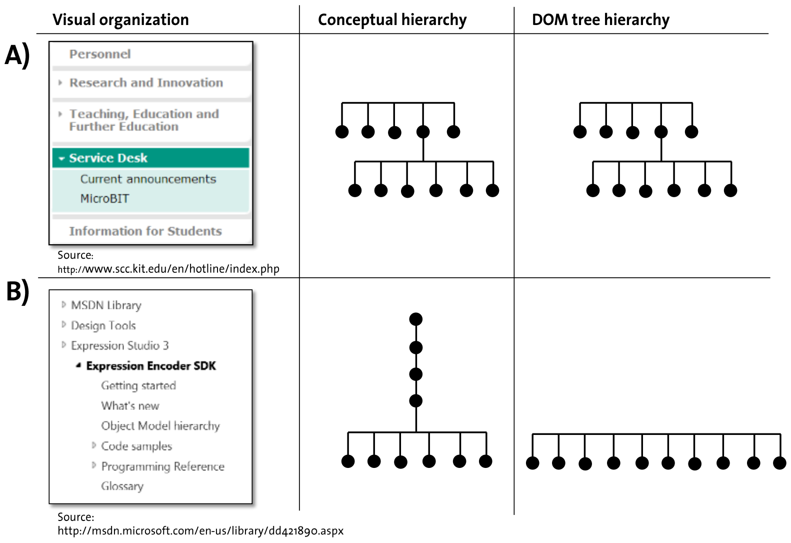


Figure 24. The subtree of the CO represented by menu A) is obvious because of the visual presentation but it is also reflected in the structure of the HTML code. Thus, it can easily be extracted automatically. However, this is often not the case, as menu B) illustrates. Here, the visual presentation indicates a tree structure similar to menu A) while it is not. In menu A), the first four items are siblings, in menu B) they are descendants (for example, the item “Design Tools” is a child item of ”MSDN Library”). This is not obvious from the visual presentation but it is also not reflected in the HTML code structure where all menu items are located at the same level. In contrast, humans are able to understand the structure once they browse the menu and explore different states.

# 4 Related Work

Before we review related work, we explain the organization and scope of this chapter in Section 4.1.

## 4.1 Organization and Scope of Related Work Analysis

The organization of this chapter is illustrated in Figure 25. We start by reviewing the terminologies used in related publications (Section 4.2) and conclude that wording inconsistencies indicate limited scholarly exchange between the involved parties. As a result of the wording inconsistencies, in depth analysis of related work is at the same time complicated and helpful. The basic CO-mining process outlined in Section 3.3.1, in which four different subtasks are distinguished, is used to structure the related work chapter as well. In addition, we consider methods for site structure analysis in general that are not based on reverse engineering menus but share the objective of extracting hierarchies (Section 4.5).

A lot of work has been done in the field of **Web page segmentation (Section 4.3)** but the contributions are difficult to overview and to assess. There seems to be no straight line of development but a large number of more or less isolated contributions that are not evaluated in comparison with previous solutions. However, page segmentation is only one partial problem to solve in order to implement CO-mining solutions and not the main focus of this thesis. Since, at the same time, the implementations of the existing methods are not available (with the vision-based page segmentation method, VIPS, being an exception), we neither can nor want to fill this research gap by conducting a comparative empirical evaluation of previous work in this thesis. Instead, the review of related work in this field aims at *a)* showing that a general solution reproducing the human perception does not exist and *b)* giving an overview over the used features, techniques and algorithms. Due to space limitations, we can only include a selection of previous work in our review. We choose the most

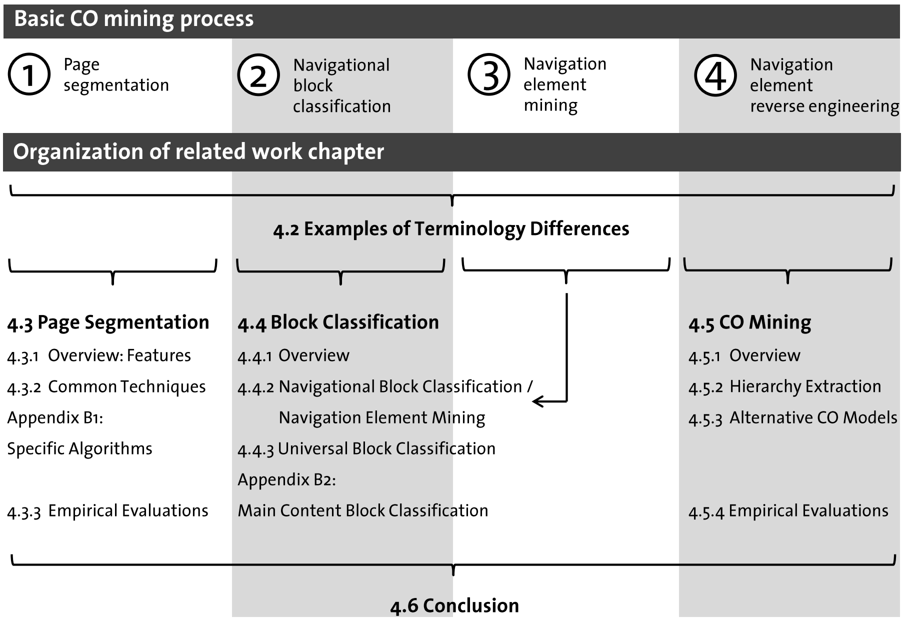

Basic CO mining process
1 Page segmentation
2 Navigational block classification
3 Navigation element mining
4 Navigation element reverse engineering
Organization of related work chapter
4.2 Examples of Terminology Differences
4.3 Page Segmentation
4.3.1 Overview: Features
4.3.2 Common Techniques
Appendix B1: Specific Algorithms
4.3.3 Empirical Evaluations
4.4 Block Classification
4.4.1 Overview
4.4.2 Navigational Block Classification / Navigation Element Mining
4.4.3 Universal Block Classification
Appendix B2: Main Content Block Classification
4.5 CO Mining
4.5.1 Overview
4.5.2 Hierarchy Extraction
4.5.3 Alternative CO Models
4.5.4 Empirical Evaluations
4.6 Conclusion

**Figure 25. This chapter is organized according to the basic CO mining process (cf. Section 3.3.1)**

popular work (based on citations) and work that introduced unique aspects with regard to the applied methodology.

The second task in the basic CO-mining process is navigational block classification, which aims at identifying the blocks related to navigation among all page blocks. Since not much work with focus on identifying navigational blocks exists, we include solutions for determining other block types as well and review solutions for **Web page block classification** in general in **Section 4.4**. The majority of the solutions in this field aim not at extracting navigational blocks but at identifying main content blocks. For the sake of completeness, a review of main content block extraction methods is provided in Appendix B.2. There are a couple of solutions for classifying all blocks of a page. However, existing work differs in the set of block types that are distinguished and even in the type granularity. The problem of a missing straight line of development and a lack of incremental improvements also applies to this field. This might be partly due to the evaluation difficulties described in Section 3.2. There is only little work that considers navigation element mining, the third task in the basic CO-mining process, which aims at finding all occurrences of navigational blocks belonging to the same navigation element. These solutions will not be discussed in a separate section but in conjunction with navigational block classification in Section 4.4.2 (cf. Figure 25).

The fourth task in the basic CO-mining process is reverse engineering extracted navigation elements in order to retrieve the CO. This specific problem has attracted not much attention from the research community yet. But there is some other work on extracting hierarchical COs that is not strictly based on menu interpretation. Thus, instead of focusing on navigation element interpretation in particular, we review work that is related to analyzing site structures in general in Section 4.5, no matter if it involves reverse engineering menus or not.

*We will not consider work that is related to the extraction of data records:* The extraction of *data records* from Web pages (also called wrapper induction) has attracted much attention from the research community in the recent years ([CKGS06] and [SLCO12] provide good overviews). These methods focus on retrieving data objects from repetitive page structures such as table rows. Actually, data record extraction is an application of CO-mining but it aims at analyzing sites that follow the database schema of organization (cf. Section 2.2.5) and not the hierarchical or sequential schema. Naturally, data record extractors are based on discovering repetitive content structures and not on navigation elements. Hence, data record extraction is a topically related field but the applied methods differ and are only marginally relevant to research problems discussed in this thesis.

## 4.2 Examples of Terminology Differences

Web page segmentation, block classification and CO-mining have been described in previous publications, but, for the most part, not as research problems on their own but as preprocessing tasks in combination with other problem statements. As a result, previous methods related to our research problem are often isolated solutions, which are not built on each other. Hence, it is not only difficult to discover related methods but also to compare methods because a common terminology has not yet evolved. However, we found that, although the terms and their definitions vary, we can map most of the underlying concepts on corresponding concepts of the $O^3$-model. In this section, we will exemplify the wording differences, to illustrate the challenges in retrieving and interpreting related work. To discuss related work in the following sections, we will translate the alternative terms used in other publications into the terms introduced in Section 2, whenever possible.

The idea that a Web page does not consist of a single homogenous region but is composed of small blocks is quite common in the field of Web mining. In an early work by Chen et al. [CZSZ01], the page blocks are called “objects”. These objects are nested and fulfill different purposes. In the eyes of the authors, the objects correspond to the HTML structure. Cai et al. [CYWM03] use the term “blocks” and consider blocks as semantic units, which can be identified by “visual and spatial clues”. They explicitly state that the “HTML structure is far different from the layout structure”. This contrasts with the work by Debnath et al. [DMPG05], published two years later, which assumes that blocks correspond to HTML elements again. Zou et al. [ZOLT06] refer to page blocks as “zones”, although their “zones” resemble the block concept of Cai et al. [CYWM03] closely. According to Wang et al. [WALZ07] a Web site consist of “information blocks” with logically related content, while Hattori et al. [HHMS07] state that a page is composed of “small objects”. Fernandes et al.

[FMSR11] use the terms “structural block” and “segment” synonymously for a “self-contained logical region” of a page that is not nested. Most related work assumes that page blocks can indeed be nested. Cai et al. [CYWM03] believe that page blocks are organized into a hierarchical “semantic structure”. Similar interpretations can be found in [YAZH01] or [XIYS06]. Chen et al. [CZSZ01] use the term “Function-based Object Model” for the object hierarchy, i.e. block hierarchy. Other terms for similar concepts are e.g. “zone tree” [ZOLT06], “partition tree” [GMBS07] or “tree of areas” [BURU09]. There is also no uniform wording for the task of classifying page blocks. Besides “block classification” (e.g., [JIJO00]), the terms “block identification” [DMPG05], “Web page element classification” [BURU09] or “role identification” [AKYE13] have been used. Most block classification methods focus on extracting the main content block, which is also referred to as “Information object [CZSZ01]”, “Body text” [FIKS01], “Informative content block” [LIHO02]”, simply “Content block” [CHMZ03], “Primary content section” [DMPG05] or “Article text” [PARO09]. Wang et al. [WALZ07] extract navigational blocks, which they call “key information”. Structures that correspond to a hierarchical CO are referred to as “Thesaurus” [CLWP03], “Website skeleton“ [LINL04], “Web-site topic hierarchy” [YALI09] or “Sitemap” [LICC11]. Another example of the sometimes confusing terminology in this field is the term “navigational link”, which is used in [CLWP03] to denote non-semantical and non-hierarchal links. This is exactly the opposite meaning of the term as in [LINL04], where navigational links denote hierarchical links.

## 4.3 Page Segmentation

In this section, we review existing solutions for segmenting Web pages into smaller units, i.e. blocks. Page segmentation can enhance the quality of Web search, e.g., by improving deduplication (e.g., [KONE08], [CHKP08]) and keyword-based Web search performance (e.g., [CYWM03], [KONE08], [FMSR11]). Furthermore, page segmentation is a prerequisite for reorganizing Web pages to be accessible on smaller screens (e.g. [CZSZ01], [CHMZ03], [BALU06], [HHMS07]). Most of the methods either generate a set of blocks (e.g. [HHMS07], [CHKP08], [ALCO11]) or a hierarchy of blocks (e.g. [CYWM03], [XIYS06], [ZOLT06], [GMBS07] ) for each page that reflects the visual or semantic structure. But not all solutions fall into one of those categories: The method by Fernandes et al. [FMSR11] produces not individual blocks but classes of blocks with similar tag paths. Baluja’s solution [BALU06] segments a Web page into exactly 9 tiles for displaying the page on small screens. Chen et al. [CHMZ03] work with a predefined block set and mine for 5 specific block types (header, footer, body, left side bar, right side bar). Most solutions have in common that a subset of the DOM elements, which defines the block boundaries, is returned as output. Hence, the segmentation cuts usually follow the boundaries of the DOM elements. An exception is the method by Cao et al. [CAML10], which is a pure image processing approach and does not take the DOM structure into account.

In Section 4.3.1, we analyze the features that are exploited by different methods, before we discuss common techniques in general (Section 4.3.2). A more detailed description of selected algorithms is provided in Appendix B.1. In Section 4.3.3, we review how the existing methods have been evaluated and how they performed.

### 4.3.1 Overview: Exploited Features

In this section, we focus on the features that are exploited by the reviewed solutions in order to conduct page segmentation. The feature selection is crucial not only for the classification performance but also for the runtime performance and compatibility with future HTML standards. Table 2 provides an overview over the features used by the solutions included in the survey. The table illustrates that most Web page segmentation methods are dedicated solutions that differ from traditional image segmentation algorithms. While traditional solutions are typically based on pixel bitmaps (e.g. [SHMA00]), only two of the reviewed solutions consider color information at pixel-level at all: the method of Cao et al. [CAML10], which is a pure image-based method and Baluja's solution [BALU06] which uses pixel information only to fine tune the segmentation cuts.

All other methods utilize the DOM-structure in one way or the other. Commonly, the positions and sizes of the individual DOM-elements are evaluated – this is done by 11 out of 18 reviewed solutions and can improve segmentation quality in comparison with traditional pixel-based solutions. Some of the solutions (e.g., [CYWM03], [CHMZ03]) use relative distance and position metrics instead of absolute values in order to deal with pages that are optimized for different screen resolutions.

However, exploiting size and position information requires visual rendering. Methods that do not need the page to be visually rendered bring the advantage of reduced resource consumption. For this reason, Hattori et al. [HHMS07] only estimate the actual element sizes based on the length of the contained text or explicit height- and width-attributes. But there are also a couple of alternative features that do not require page rendering. The most common non-visual feature is given by the types of the HTML elements. While tag types are usually used as one feature among others (e.g. [CYWM03], [CHMZ03], [KONE08] or [GMBS07]), the solutions by Debnath et al. [DMPG05] and Lin et al. [LICC11] conduct page segmentation solely based on this feature. However, the dependency on HTML elements has a major drawback: When the Web standards and coding conventions change over time such methods might not work anymore in the same way. For example, most of the type-depended solutions exploit the *table*-element to compute the blocks (e.g., [CYWM03], [CHMZ03], [DMPG05]). Previously, this element was playing an important role for organizing the page layout. But, in current practice, the *div*-element is used for this purpose and the *table*-element is only applied to format actual data tables.

Features that are based not on the element types but on the structure of the DOM-tree are also popular. For example, Cai et al. [CYWM03] divide nodes with a single child, while Chakrabarti et al. [CHKP08], Hattori et al. [HHMS07] as well as Alcic and Conrad [ALCO11] use the DOM tree structure to compute distance metrics between two elements. Fernandes et

**Table 2. Comparison of the features that are exploited by the reviewed solutions**

| | Element types | Back-ground-colors | Font features | Element position and size | DOM-structure | Site-wide structure | Text fea-tures[11] | Pixel data |
|---|---|---|---|---|---|---|---|---|
| Yang and Zhang [YAZH01] | x | | x | | | | x | |
| VIPS, Cai et al. [CYWM03] | x | x | x | x | x | | | |
| Chen et al. [CHMZ03] | x | x | x | x | x | | | |
| Debnath et al. [DMPG05] | x | | | | | | | |
| Baluja [BALU06] | x | x | | x | x | | | x |
| Xiang et al. [XIYS06] | x | x | x | x | x | | | |
| Zou et al. [ZOLT06] | x | x | x | x | | | | |
| Guo et al. [GMBS07] | x | | x | x | | | | |
| Hattori et al. [HHMS07] | x | | | x[12] | x | | | |
| Kohlschütter and Nejdl [KONE08] | (x)[13] | | | | | | x | |
| Chakrabarti et al. [CHKP08] | x | x | x | x | x | | x | |
| Burget [BURU09] | | x | x | x | | | | |
| Yang and Shi [YASH09] | x | x | x | x | x | | x | |
| Vineel [VINE09] | x | | | | x | | x | |
| Cao et al. [CAML10] | | | | | | | | x |
| Alcic et al. [ALCO11] | | | | x | x | | x | |
| Lin et al. [LICC11] | x | | | | | | | |
| Fernandes et al. [FMSR11] | | | | | | x | | |

[11] E.g., sentence length or text density
[12] Estimated, the page is not visually rendered
[13] Tag-agnostic and tag-aware methods are used, but tag-aware methods achieve significantly better segmentation performance

al. [FMSR11] use a different approach: they align the DOM structure of multiple pages to conduct the segmentation on a per-site basis instead of on a per-page basis.

Some approaches make use of text-based features, which do also not require page rendering. For example, Kohlschütter and Neidl [KONE08] use text-density as their main criterion for segmentation, while Chakrabarti et al. [CHKP08] consider sentence length as one component in their feature pool.

## 4.3.2 Common Segmentation Techniques

This section provides an overview over techniques that are commonly applied in previous work to conduct page segmentation. Specific algorithms are described in Appendix A. The techniques are not mutually exclusive – page segmentation solutions may involve multiple of the described techniques.

**DOM element removal / selection:** The most primitive way of conducting page segmentation is to simply consider specific HTML element types as blocks and ignore all other elements. Early segmentation solutions (e.g., the method used in [DLCT00]) are solely based on this principle[14]. However, some newer solutions remove specific tags in a preprocessing phase (e.g., br-elements in the solution presented by Xiang et al. [XIYS06]), before more sophisticated segmentation techniques are applied.

**DOM tree traversal:** A very common technique is to either traverse the DOM tree in a top-down manner to iteratively split the tree or to traverse it in a bottom-up manner to iteratively combine leafs into larger blocks. The top-down approach is taken, e.g., in the first phase of the VIPS algorithm [CYWM03], in both segmentation phases of the method by Chen et al. [CHMZ03], by the method described by Debnath et al. [DMPG05] and by the solution of Zou et al. [ZOLT06]. Examples for the bottom-up method can be found in [CYWM03] (second phase), in [XIYS06], where it is used for preprocessing, and in [GMBS07].

**Identification of recurring pattern:** The analysis of recurrent structures within a page can also yield segmentation information. This is because a block often consists of child blocks with similar structure. For example, the main content block of a homepage of a site from the news domain may be composed of a number of similar structured article teasers, each with a headline, a short abstract and a link. If such patterns can be discovered, each individual pattern occurrence indicates a separate block and the pattern sequence indicates a container block. The methods described in [YAZH01], [XIYS06] and [GMBS07] are based on this principle but they differ in the considered features and pattern recognition algorithms. The solution by Lin et al. [LICC11] exploits information about recurring patterns in a different way by separating page regions with high-complexity from regions with low-complexity, i.e. high repetitiveness.

[14] We did not include these methods in our review because their applicability to today's Web sites is limited and we consider them as obsolete.

**Segmentation as 1-dimensional problem:** Kohlschütter and Nejdl [KONE08] as well as Lin et al. [LICC11] interpret Web pages as sequences of tokens and disregard the hierarchical structure of HTML-documents. Interestingly, both apply solutions from other domains to address the page segmentation problem. The solution by Kohlschütter and Nejdl [KONE08] is motivated by a linguistic perspective. The authors mine changes in the text density in order to separate different page regions. Lin et al. [LICC11] approach the problem with a bioinformatical background and consider Web pages as protein sequences by mapping tags onto amino acids. A sequence splitting algorithm from the bioinformatics domain is applied that separates regions with high complexity from regions with low complexity.

**Graph-clustering:** With an appropriate distance metric, the DOM-tree elements can be considered as vertices in a weighted graph to map the page segmentation problem onto a graph-clustering problem, as shown in [CHKP08] and [ALCO11]. The resulting clusters represent page blocks. Alcic and Conrad combine three different distance metrics with three different clustering algorithms and present the results in [ALCO11].

## 4.3.3 Empirical Evaluations

In this Section, we provide an overview over empirical experiments, in which the reviewed page segmentation methods were evaluated. Precision, recall, F1-measure and accuracy are measures typically used to evaluate information retrieval tasks that involve discovering relevant items, i.e. positives in a set of relevant and irrelevant items (negatives). Each item can be classified into one of the four types: 1) true positive (TP) if the item is relevant and was retrieved, 2) false positive (FP) if the item is irrelevant but was retrieved, 3) true negative (TN) if the item is irrelevant and was not retrieved and 4) false negative (FN) if the item is relevant but was not retrieved. Precision, recall, F1-measure and accuracy are based on this typology:

$$Precision = \frac{TP}{TP+FP} \tag{1}$$

$$Recall = \frac{TP}{TP+FN} \tag{2}$$

$$F1 - measure = \frac{2 \cdot TP}{2 \cdot TP+FP+FN} \tag{3}$$

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{4}$$

The F1 measure combines precision and recall into one value by using the harmonic mean. Accuracy expresses the ratio of the number of correctly classified items (either true positive or true negative) to the number of all items. Although these measures have been used to evaluate page segmentation methods, there is no straightforward way of applying them in this scenario. This is due to the fact that a perfect agreement of two different segmentations is unlikely (when comparing machine-generated segmentation to human labeled segments as well as when comparing segmentation conducted by two different human assessors). Hence, it is reasonable to tolerate small deviations and impractical to distinguish true and false positives without allowing for some fuzziness. But unclear and undocumented criteria for these distinctions undermine the comparability of some of the reviewed solutions.

We distinguish 1) experiments that aim at directly measuring the segmentation quality in comparison with manually labeled data, 2) experiments that involve a previous method as baseline and 3) experiments that involve page segmentation as preprocessing and only indirectly capture the segmentation quality.

**1) Evaluation against manually labeled data:**

- In the original paper [CYWM03], the VIPS-method is applied to 600 selected Web pages and the results are manually assessed by 5 volunteers. The authors report 93% at least satisfactory results. Although VIPS is probably the most popular method, later evaluations confirmed (e.g. [YASH09] and [FAHB09]) that the results of the VIPS-method are often inaccurate.
- The method by Zou et al. [ZOLT06] was only evaluated for Web pages from the medical domain. 104 pages from 11 journals were manually segmented. The authors report an accuracy of 96.40% but do not report the criteria for true positives.
- Hattori et al. [HHMS07] used 100 selected Web sites to evaluate their method. They excluded sites with much CSS and Javascript, which means that their evaluation set does not represent current Web sites very well. The sites were manually segmented. Precision, Recall and F1-measure are provided. For sites from the US, a F1-measure of 0.75 was achieved. According to the authors, a true positive means that the segmentation positions of the automated method and the human labeling “are in agreement”.
- Chakrabarti et al. [CHKP08] manually label blocks from 105 randomly sampled pages. The evaluation methodology is well-documented and plausible. They consider each page block as a cluster of DOM-elements. Thus, a page segmentation is considered as a clustering of DOM elements and standard metrics for measuring cluster similarities (Adjusted RAND [HUAR85], Normalized Mutual Information [STGH03]) are applied. However, the measured agreement between the manually labeled segmentation and the automatically determined partitions is low (AdjustedRAND: 0.6, NMI: 0.76).
- Kohlschütter and Nejdl [KONE08] adopt the evaluation methodology of Chakrabart et al. [CHKP08] but the clustering agreement is measured based on tokens (words) but not on DOM-elements. However, the authors assume comparability and report that their tag-agnostic method has a very similar classification performance as the method by Chakrabart et al. [CHKP08]. A rule-based extension that involves tag-types boosts the classification performance by still leaving much room for improvements. Remarkably, a pure tag-based variant that does nothing else than selecting specific tags as segments did not perform significantly worse in the evaluation.

- Vineel [VINE09] uses 400 manually segmented Web pages to evaluate his solution. According to the author, precision and recall are around 0.9 and 0.8 respectively. The paper does not provide details on how true positives are defined.
- Alcic and Conrad [ALCO11] also apply cluster comparison to evaluate their algorithms similar to [CHKP08] and [KONE08]. Their data set contained 78 manually labeled pages. However, their results are not comparable to [CHKP08] and [KONE08] because the original version of the RAND index [RAND71] and not the corrected-for-chance version [HUAR85] is used. With a maximal RAND agreement of 0.61, the segmentation performance of the solution is not convincing.
- Lin et al. [LICC11] use manually labeled samples from three different sites. They address the problem of block overlap and define their own interpretations of precision and recall metrics. But it is very hard to assess their results because, although their metrics seem sound, they are non-standard and no baseline method is evaluated.

2) **Evaluation against a baseline method:**

- Fernandes et al. [FMSR11] generate four large baseline data-sets from different sites by applying the VIPS algorithm [CYWM03] with manual parameter settings. The quality of this baseline segmentation is not empirically evaluated. The data sets are used to measure cluster agreement based on terms. In comparison to the approach by Kohlschütter and Nejdl [KONE08], the method by Fernandes et al. [FMSR11] achieved a significantly higher agreement with the baseline segmentation. However, a substantial difference between the authors' method and the baseline method remains.
- Xiang et al. [XIYS06] use a collection of 40 manually labeled pages from 13 selected sites to compare their method to the VIPS algorithm [CYWM03]. The authors measure only recall and do not publish precision values. Hence, it is difficult to assess the quality of their segmentation results.
- Yang and Shi [YASH09] apply their method and the VIPS-method [CYWM03] to segment 160 Web pages from different domains. Human assessors assigned one of the three labels "error", "not-bad" and "perfect" to each retrieved block. The results are normalized according to the block areas. It is reported that with the method proposed by the authors in average 88.22% of the page area is segmented at least not-bad, while VIPS only reaches a value of 74.14%.

3) **Indirect evaluation of segmentation quality**

Often, the quality of page segmentation solutions is evaluated indirectly by assessing the ability of the solution for improving other tasks related to information retrieval. Cai et al. [CYWM03] show that their VIPS-method is able to slightly improve retrieval results if query expansion is conducted in a block-aware way. Fernandes et al. [FMSR11] apply a block-aware ranking model, which is described in [MFRS10], and report that their automated approach increases result quality and is only slightly inferior to a block-aware model based on manual segmentation. Chakrabarti et al. [CHKP08] show that the detection of duplicate pages can be significantly improved if not conducted based on the full textual content but on their segmentation method. Kohlschütter and Nejdl [KONE08] adopt the evaluation methodology of Chakrabarti et al. [CHKP08] and demonstrate that their own segmentation methods are more suitable for duplicate detection. Hattori et al. [HHMS07] use their segmentation solution to transcode sites for mobile applications. For evaluation they estimate the time a user needs to access information, solely based on screen distances. Cao et al. [CAML10] show that their method can be used to recognize phishing sites. However, they use a non-segmenting method as baseline and do not consider previous segmentation solutions.

## 4.4 Block Classification and Navigation Element Mining

In this Section, we distinguish methods for *navigational block classification*, *informative content block classification* (discussed in detail in Appendix B.2) and *general block classification*:

- **Methods for navigational block classification / navigation element mining:** There are only few solutions that involve block classification with specific focus on navigational blocks. As a result, there is also not much research on finding recurring navigational blocks that represent the same navigation element. The majority of the existing methods that involve navigational block classification, use it to generate some kind of hierarchical Web site model (e.g., [LINL04], [WALZ07], [YALI09] and [LICC11]) .
- **Informative content block classification methods:** Most work on block classification focusses on detecting a single block type, the informative content block. The informative content block is the block that contains the actual page content, e.g., an article. Distinguishing informative content blocks from “noisy” blocks, e.g., advertisements, navigational blocks, etc., is important for the performance of search engines. First, only the informative content block is relevant to the users’ search queries and the noisy blocks can distort the results. Hence, only the informative content block should be indexed. Second, to de-duplicate URLs effectively, i.e. to detect different URLs that point to documents with the same content, the noisy blocks should be filtered because dynamically inserted advertisements can prevent de-duplication. Third, considering only the informative content blocks does not only increase the quality of the index but also decreases its size. Fourth, as a result of a search query, a fragment of the informative content block should be presented to the user that he can effectively judge whether the page contains the information he is looking for or not. Since most methods for informative content extraction cannot be applied to mine navigational blocks, we discuss these methods in Appendix B.2.

- **General block classification methods:** There are a couple of methods that aim at deriving the types of all blocks. Existing methods do not agree on the classification schema or the classification granularity (e.g., Chen et al. [CHMZ03] distinguish 5 block classes, while Akpinar and Yesilada [AKYE13] use a schema with 27 block classes). The main motivation behind general block classification solutions is transcoding Web sites (e.g., in the case of [CZSZ01], [CHMZ03]) for small screen devices. The idea is to use knowledge about page block types to rearrange pages and possibly split them up into multiple screens. Besides transcoding, other discussed applications include ad blocking, information extraction in general and improvement of the accessibility with regard to disabled people (e.g., [LEKL04], [AKYE13]).

In the next section, we first provide an overview over the reviewed solutions, before methods for navigational block classification and general block classification are discussed separately in Section 4.4.2 and Section 4.4.3, respectively. Methods for informative content block classification are discussed in Appendix B.2.

## 4.4.1 Overview

In this section, we summarize the key characteristics of the considered related work in the field of block classification. Table 3, Table 4 and Table 5 provide an overview over the reviewed solutions for navigational block classification, informative content block classification and general block classification respectively. We present the classes that are distinguished (in the terminology of the authors), a brief summary of the method, the exploited features and the evaluation methodology. Regarding the evaluation methodology, we only mention experiments that focus on the classification performance, i.e. the quality of the results.

**Table 3. Navigational block classification methods**

| | Classes | Method | Features | Evaluation |
|---|---|---|---|---|
| Liu et al. [LINL04] | Navigational link sets (= hierarchical menus) / other blocks | Links are first clustered and the resulting sets are ranked; highest ranked set is considered as main menu | Link texts and URLs | Leave-one-out evaluation for 5 news sites, overall classification performance not evaluated |
| Rodrigues et al. [ENMG06][MEMF07] | Different types of navigation elements: s-nodes, c-nodes, i-nodes | Site-oriented method; rule-based extraction and classification of navigational blocks | Number of block occurrences, ULRs | Classification performance not evaluated |
| Wang et al. [WALZ07] | "Key information" (breadcrumb or active menu item) / all other blocks | Site-oriented method; simple rule-based method | Number of words, tag paths, occurrence entropy | 5 selected sites, rank of key information in result list |
| Lin et al. [LICC11] | Structure blocks (= navigational blocks) / content blocks | Rule-based method, Web pages are considered as protein sequences | Block complexity based on tag sequence, ratio of link text to all text | Evaluated on 2 sites; precision, recall and F1-measure calculated in a non-standard way |

**Table 4. Informative content block classification methods**

| | Classes | Method | Features | Evaluation |
|---|---|---|---|---|
| Finn et al. [FIKS01] | Body text / clutter | Cumulative distribution of tags is analyzed | Two types of tokens: words and tags | Classification performance is not evaluated |
| Lin and Ho [LIHO02] | Informative block / redundant blocks | Word entropy is computed; blocks with low average word entropies are classified as informative blocks | Word frequencies | Classification performance is not directly evaluated |
| Yi et al. [YILL03] | Main content block / noisy block | Site-oriented method; Entropy is computed based on text and DOM-attributes; low entropy values indicate main content blocks | DOM tree structure and DOM element attributes | Classification performance is not evaluated |
| Kao et al. [KAHC05] | Informative blocks (articles and TOCs) / redundant blocks | Method is based on term entropy and text-length entropy; entropies are aggregated; classification based on thresholds | Word frequencies, text length distribution, link text length distribution, link text and target page text overlap | Classification performance evaluated for selected news sites; optimal thresholds estimated in advance |
| Song et al. [SLWM04] | Three levels of importance | Method is based on machine learning (SVM and Neural Network); manually labeled training set and classic features | Absolute and relative heights / positions, image numbers and sizes, link numbers, text and link text lengths, numbers and sizes of form elements | 5-fold cross validation, 600 pages from 405 sites (manually labeled) |
| Debnath et al. [DMPG05] | Primary content blocks / noninformative content blocks | Three different algorithms: Content extractor (site-oriented), Feature extractor, k-feature extractor | Content extractor: Block similarity<br>Feature extractor, k-feature extractor: Word number, tag number, list number, etc. | Precision and recall based on manually labeled samples from 15 sites |
| Gottron et al. [GOTT08] | Informative content block / other blocks | Distribution of tags is analyzed | Number of words and tags, number of characters in words and tags | 13 selected sites, precision and recall for longest common subsequence |
| Burget et al. [BURU09] | Sub-blocks of main informative content area, e.g., h1, h2, subtitle, author, date, etc. | Decision tree (C4.5) classifier trained on a small feature set | Font sizes and weights, number of blocks below, left, right and above, contrast, 5 more textual features | Evaluated on only 4 (unseen) sites; precision, recall and F1-measure |
| Pasternack and Roth [PARO09] | Article text (= informative content block) / other blocks | Naïve bayes classifier trained on two features; supervised and unsupervised methods | Token trigrams and most-recent unclosed tags | Evaluated on 5 news sites and 40 sites from other domains; precision, recall and F1-measure |

**Table 5. General block classification methods**

| | Classes | Method | Features | Evaluation |
|---|---|---|---|---|
| Chen et al. [CZSZ01] | Information object, navigation bar, navigation list, independent navigation guide, interaction object, decoration object, special function object, index page, content page | Simple, rule-based method; only algorithms for navigation bar and page objects described | Navigation bar: classes of child nodes, uniformity of child nodes, text length, hyperlink targets, index page, content page: out- / in-degree ratio | Classification performance is not evaluated. |
| Chen et al. [CHMZ03] | Header, footer, left side bar, right side bar, body | Only positions of blocks on the page are considered to determine type | Block positions | Classification performance is not directly evaluated |
| Lee et al. [LEKL04] | 17 different classes: Main content, title, navigation, search, supporting content, site content, advertisement, etc. | Co-trained machine learning method | Stylistic (table cell flow, table cell word density, relative position, font features, etc.) and lexical (bag of words, part-of-speech features, etc.) | Evaluated on 50 manually labeled samples, satisfying results only for the 3 most frequent classes |
| Akpinar and Yesilada [AKYE13] | 27 different classes: advertisement, article, logo, copyright, header, etc. | Rule-based method | Diverse features including tag names, element sizes, element attributes, CSS styles, keywords, etc. | Survey with 30 sample pages |

## 4.4.2 Navigational Block Classification / Navigation Element Mining

We first describe previous work on navigational block classification and navigation element mining in Section 4.4.2.1, before we analyze in Section 4.4.2.2 how these methods have been evaluated.

#### 4.4.2.1 Methods

The hierarchy extraction framework of Liu et al. [LINL04] contains a method for identifying navigational link sets, which are understood as those menus that contain links to child nodes of a page. The authors first cluster the links of a page by their position (depth and path) in the DOM tree. These candidate link sets are further cleaned by analyzing link attributes. Link types (images or plain text), style uniformity of the links, physical proximity of the links and text lengths are considered in this procedure. The cleaned list of candidate link sets is then ranked by evaluating six features solely based on the link texts and the URLs (e.g. text length, text length variance, etc.). The authors use manually labeled navigational link sets to precompute the average values for each feature. The derivation from these averages is used to rank the candidate set and to estimate the probability that a candidate actually represents a navigational link set.

Rodrigues et al. propose a way of extracting and classifying navigational blocks in [ENMG06] and [MEMF07]. Since they are only interested in navigational blocks, they simply extract continuous lists of hyperlinks from pages instead of conducting real page segmentation. Isolated links are aggregated into virtual blocks, so called i-nodes. Similar blocks from different pages are grouped if they share at least 60% of the hyperlinks. This block grouping task corresponds to a simple navigation element mining solution as defined in our basic CO extraction process. The resulting navigation elements are classified into three classes: besides i-nodes, c-nodes and s-nodes are distinguished. A navigation element is classified as s-node if and only if more than 60% of its links point to pages that contain this node as well. In other words, s-nodes are “fixed” menus in a way that when a user clicks on a link, the menu is still present after the page transition. According to the authors, these navigational elements underpin the organization of the site.

Wang et al. [WALZ07] proposed a method for extracting the “key information” of Web pages. In their paper, the term key information refers to navigation elements that allow locating the position of a Web page in the hierarchical CO. Breadcrumb navigations and navigation elements that include links to the active page are mentioned but the authors do not further specify the types or properties of the blocks that they are interested in. Although the approach aims at distinguishing the navigation elements of interest from “noisy information menus” and “accidental menus”, it does not recognize the type of the navigational block, e.g. whether it is a breadcrumb navigation or not. Wang et al. use a very simple rule-based approach: First, they extract DOM elements whose child nodes only contain text with less than five words. These blocks are merged across all pages by joining two elements if the tag path and the text of the first child elements are identical. The merged blocks are further grouped if *either* the tag paths *or* the texts of the first child elements match. This results in a set of navigation elements, each of which consisting of a set of instances and each instance associated with a number of source pages. Based on the instance / page number distribution, the entropy is calculated for each navigation element. The authors argue that navigation elements for which a higher entropy value is computed are more likely to contain key information.

The hierarchy extraction solution of Yang and Liu [YALI09] (cf. Section 4.5) includes a primitive method for identifying navigational blocks: DOM elements for which the ratio of the length of linked text to the length of all text exceeds a threshold of 0.8 are considered as navigation bars, i.e. navigational blocks.

The framework of Lin et al. [LICC11] contains a solution for identifying navigational blocks as well as a method for merging them into navigation elements. The unique characteristic of their approach is the fact that Web pages are considered as protein sequences in which the element types represent different amino acids. A method from the field of bioinformatics is used to estimate the complexity, i.e. repetitiveness, of a block based on the elements it contains. Simple rules based on the complexity values and on the ratio of link text to all text in a block are used to identify navigational blocks. If the rules cannot be applied, a previous block classification framework [LIHO02] is used as fallback solution. Another algorithm from the field of bioinformatics is used to discover similar navigational blocks, i.e. to find blocks that represent the same navigation element. A predefined similarity threshold allows for some fuzziness.

#### 4.4.2.2 Performance Evaluations

Liu et al. [LINL04] use 5 manually labeled Web sites from the news domain to evaluate their framework. The evaluation includes analyzing the performance of detecting those menus that represent the hierarchy levels. A two-step menu extraction method is used, which first generates menu candidates and then selects the most-likely main menu. Both steps are evaluated individually but combined classification performance is not listed. However, very good precision and recall are measured for both tasks.

Rodrigues et al. [ENMG06] and [MEMF07] do not evaluate the classification performance of their method for identifying different navigation elements.

Wang et al. [WALZ07] evaluate their method for 5 selected Web sites. Their algorithm returns a sorted list of navigation elements. As evaluation metric, solely the position of the first correctly identified navigation element type in the list is provided for each Web site. Only in two cases, the top-ranked navigation element actually contained “key information”, i.e. a hierarchical menu or breadcrumb navigation. However, in all five cases, a navigation element considered as key information was among the top 3 results.

Yang and Liu’s simple method for extracting navigational blocks is not evaluated in their paper [YALI09].

Lin et al. [LICC11] evaluate the performance of their method for extracting navigational blocks only for three selected sites. Based on a manually labeled sample set, an average precision of 0.88 and an average recall of 0.83 are measured. However, even if their interpretation of precision and recall seems sound in the evaluation scenario, it differs from the common usage and the results are difficult to assess.

### 4.4.3 Universal Block Classification Methods

Jinlin Chen et al. [CZSZ01] were among the first to propose a method for classifying different kinds of page segments. They argue that the semantic information provided by the HTML element types does not express the category of the element (or object in the author’s terminology) and the Webpage author’s intention very well. Chen et al. derive a list of block classes from a single sample page. They distinguish page blocks that present content, different classes of navigational blocks, blocks for decorative purposes, et cetera (cf. Table 5). The authors only describe algorithms for the detection of three different classes, namely navigation bars, index pages and content pages and do not provide solutions for other block types. The algorithms are simple and rule-based. The threshold values used to set up the algorithms are not mentioned in the paper.

A method for adapting Web pages to small screen devices by Yu Chen et al. [CHMZ03] distinguishes five different high-level elements: header, footer, left side bar, right side bar and body. The body block corresponds to the informative content area at the center of the page. To distinguish the other four types, only the block position is considered. For example, blocks located in the left-most quarter of the page are classified as left side bar. To detect the header and the footer a dynamic threshold depending on the block with/height ratio is used.

Lee et al. [LEKL04] use a very fine-grained set of 17 different block classes. The set of classes ranges from common types such as informative content or navigation to classes that are not considered as separate blocks by other approaches such as article date or article title. Consequently, the authors understand blocks as much smaller units compared to previous approaches (the 20 manually labeled documents result in 1625 labeled blocks). Lee et al. apply co-training [BLMI98] in order to boost classification performance at the basis of only a small training set (cf. Figure 26). In co-training, the feature set is split into two halves to train two independent classifiers. Both classifiers are trained initially on a small labeled set. They are then applied on the unlabeled samples. From each classifier, the k most confident newly labeled samples are added to the labeled training set. The idea is that one classifier often is able to determine the class with high confidence, while the other is not able to classify the same sample. Thus, this sample can be used as additional training sample for improving the second classifier. The labeling/feedback/re-training cycle is iteratively repeated. Lee et al. train one classifier on a set of stylistic features (e.g. font properties, positions, etc.) and a second one on lexical properties (bag of words, part-of-speech tags, etc.) and perform up to 20 co-training iterations.

Akpinar and Yesilada describe a method for detecting fine-grained block classes in [AKYE13]. They distinguish 27 different classes, ranging from common high-level types such as article, header, footer, sidebar or menu to very specific classes such as logo, figure or separator. The used class library also contains very general concepts such as container. The method is based on the evaluation of manually specified rules. The authors do not give details about the used rules and how they were generated in the paper.

#### 4.4.3.1 Evaluations of Universal Block Classification Methods

Jinlin Chen et al. [CZSZ01] do not empirically evaluate their universal method but exemplify the performance for a single sample page.

Yu Chen et al. [CHMZ03] do not evaluate the block classification separately but the result of the small screen adaption with mixed results (ranging from complete fails for some sites to perfect adaption for others).

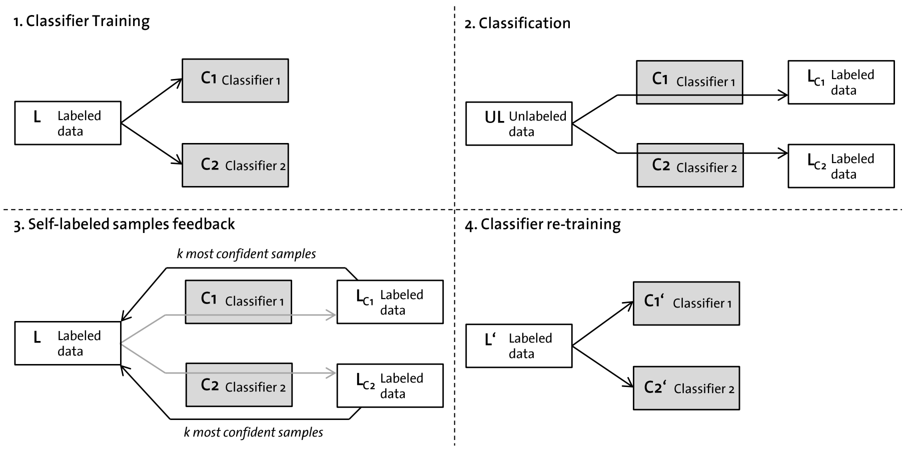


**Figure 26. Principle of co-traininig: (1) Two different classifiers are trained with the labeled data set, while each classifier uses different features; (2) both classifiers are applied on the unlabeled training data; (3) from each of the self-labeled sets, the k most confident samples are added to the labeled data set L; (4) in the next iteration, the extended set of labeled data is used to re-train the classifiers.**

Lee et al. [LEKL04] train classifiers for 17 different block types and use a sample set of 50 manually labeled pages from the news domain for evaluation. Their method shows a satisfying performance for only the three most frequent block types (main content, navigation and search) but performs poorly in classifying the remaining block types. The authors compare the classification performance of their method to the results of the main content extraction method by Song et al. [SLWM04] and find that their own, co-trained method results in higher error rates.

Akpinar and Yesilada describe an evaluation of their rule-based approach in [AKYE13]. Only 30 different pages were considered in their evaluation. The interpretation of the results is difficult because the paper does not contain enough details on how the used accuracy measure was calculated. Furthermore, the participants of the survey used a different classification system, which was ex post matched with the classes of the authors' solution. In addition, the authors report that in only about 33% of the cases more than 50% of the participants agreed on the block label. This might indicate that the used classification system is inappropriate.

## 4.5 Navigation Element Reverse Engineering and Alternative CO-Mining Approaches

While the focus of the previous sections was on analyzing the intra-page structure, we now review methods that aim at analyzing inter-page structures.

### 4.5.1 Overview

We consider only solutions for analysis on site-level and do not include structure mining methods that are typically applied in a cross-domain scenarios such as PageRank [PBMW99]. We can distinguish solutions that *extract* structures from solutions that *generate* structures. Solutions that *extract* structures are compliant with the $O^3$-model in a way that they assume an existing CO that is obvious to humans but concealed to machines. These solutions are considered as related work in this section.

Solutions that *generate* structures do not make the assumption of a embedded CO. Instead, they induce new structures and, hence, the underlying research questions are different. Many authors seem not to be aware of the existence of predefined COs, while other authors assume COs as a matter of course. Since many authors are not aware of the two different perspectives themselves, the distinction is often not clearly formulated in the publications in this field. However, the authors' argumentations usually reveal whether the focus is on hierarchy extraction or hierarchy generation. In addition, solutions that are not evaluated against manually labeled data sets clearly indicate the hierarchy generation perspective. Examples for solutions that are not considered as related work in this thesis because the focus is on hierarchy *generation* include link-based methods (e.g., [BORS92]), text-based methods (e.g. [MLGR98], [DUCH00]), hybrid text-/link-based methods (e.g., [LYHL10][HOEX12]) and usage-based methods (e.g., [CLZC05]). Since the terminology used in this field is very heterogeneous and not yet settled, the title of a publication is not always a good indicator, whether the work is related to the research questions addressed in this thesis or not. For example, the paper by Yang and Liu titled “Web Site Topic-Hierarchy Generation Based on Link Structure” [YALI09] is indeed relevant to our research. A very similar title of a different work is “Hierarchical Topic Segmentation of Websites” [KUPT06] but here, the addressed research problem is different. The authors of the latter publication focus on finding homogenous sections in a given tree (URL-induced) whose nodes are labeled with topics. An overview over the approaches that are considered in this section is provided in Table 6. Since little work has been done that aims at mining COs, we included not only methods that focus on hierarchical COs but also two solutions that extract non-hierarchical Web-site structures (the work done by Chen et al. [CLWP03] and by Rodrigues et al. [ENMG06, MEMF07]).

**Table 6. Overview over related CO-mining approaches**

| | Extracted model | Method | Features | Evaluation |
|---|---|---|---|---|
| Chen et al. [CLWP03] | Thesaurus for sites from the same domain (hierarchical CO with associative links) | Removal of navigation links | URL structure, block types using [CZSZ01], number of navigational block occurrences | 13 sites from the shopping domain, only precision |
| Liu et al. [LINL04] | Hierarchical CO | Main menu extraction | Style, URL and text features for main menu extraction | 5 sites from the news domain, precision, recall and f1 measure |
| Rodrigues et al. [ENMG06, MEMF07] | Abstracted, non-hierarchical site model | Navigation element mining | DOM structure | 2 sites, method is applied to detect entry pages. |
| Yang and Liu [YALI09] | Hierarchical CO | Training of classifiers for parent-child links and standard graph-based algorithms | URL structure, content relevance (tf-idf-based), navigation bar extraction, co-reference, link position, link text length, font size | Leave-one-out evaluation, 5 sites |
| Yang et al. [YJZN10] | Hierarchical CO | Parsing of HTML lists | DOM structure, HTML element types | Indirectly (in conjunction with related entity finding) |
| Lin et el. [LICC11] | Hierarchical CO | Recursive hierarchy construction by iteratively selecting the most important navigation element starting from the entry page | Number of navigation element occurrences, HITS-based [KLEI99] hub score | Precision, recall and F1 measure, actual sitemaps as baseline, 3 sites |

### 4.5.2 Hierarchy Extraction

Although Liu et al. [LINL04] use a very uncommon terminology and claim to extract the "skeleton" of a Web site, their method is actually one of the first solutions for mining hierarchical COs. Like the methods presented in this thesis, Liu et al. aim at reverse engineering menus. The authors' framework relies on the identification of "navigational link sets", which represent those menus that contain the links to the immediate child nodes of a page (the applied menu extraction approach is discussed in Section 4.4.2). The hierarchy construction method is quite simple: First, the navigational link set of the entry page is extracted. The contained links define the first hierarchy level. Then, the navigational link sets for all first-level pages are mined to generate the second level of the hierarchy and so on. At each level, the navigational link sets of the previous levels are ignored to ensure that the links point to lower-level pages.

Yang and Liu present another method for extracting hierarchical structures from Web sites in [YALI09]. Although a different terminology is used, the covered research problem resembles the CO-mining problem as formulated in this thesis very closely. The authors distinguish two different link types, aggregation links and shortcut links. According to the definition by Yang and Liu, aggregation links represent parent-child relationships, while shortcut links do not. The set of aggregation links represents the hierarchical CO, i.e. the topic-hierarchy in the terminology of the authors. A machine-learning approach is taken to train classifiers for the two link types. The authors train and evaluate three different classifier types: a decision tree classifier, a naïve Bayes classifier and a logistic regression classifier. The classifiers are trained on a broad range of features which capture the URL hierarchy, the existence of an index page, the content relevance estimated using a tf-idf-model [SAWY75], the presence of a navigation bar, the number of co-occurrences of a link, the position of a link within a text, the link text length and the font size. Interestingly, the method does not simply output the links classified as aggregation links. Instead, the probabilities of links being aggregation links, which are estimated by the classifiers, are transformed into edge weights. Then, two different graph-based algorithms that make use of edge weights are applied to generate a tree structure rooted at the entry page of the site. Although not explicitly mentioned by the authors, this approach might be motived by the fact that the classification results are not accurate enough to produce tree-like structures. In addition to the two weight-sensitive algorithms, the authors also generate a hierarchical model by traversing the Web site in breadth-first manner.

Yang et al. claim to reconstruct the “logical hierarchical sitemap”, which can be understood as the hierarchical CO, in [YJZN10]. However, the proposed method is very primitive and basically consists of parsing HTML lists. Since there are often many different HTML lists on a single page and not all lists represent the hierarchical site structure, the approach is not applicable in most scenarios. However, the focus of the authors is not on accurately mining the CO but on gaining some hierarchy information to enhance related entity finding.

Lin et al. [LICC11] propose a sophisticated system for extracting hierarchical COs. In the authors’ terminology, they automatically generate sitemaps. The system includes a page segmentation module (discussed in Section 4.3) and a module for identifying navigation elements (discussed in Section 4.4). The actual sitemap building is based on the number of times a navigation element occurs on the entire site and the hub values that are computed for each navigation element with a modified version of the HITS algorithm [KLEI99]. Both values aim at estimating the importance of a navigation element. The product of both values is used to combine them into a single measure. The sitemap construction starts by selecting the top-ranked navigation element from the entry page, which is considered as the first sitemap level. Navigation elements at the bottom of a page are excluded a priori in order to ignore copyright statements or similar blocks placed in the page footer. In a second step, the top-ranked navigation elements are selected for each of the first-level pages to construct the third level of the hierarchy and so on. In each recursion step, the navigation elements that were previously selected are not considered any more.

### 4.5.3 Extraction of Alternative Site Models

Chen et al. [CLWP03] do not extract hierarchical COs but thesauri. While hierarchies only involve a single kind of relationship, which are parent-child-relations, thesauri model additional associative connections to link terms with similar meanings. A thesaurus contains the same information as two separate COs, one being hierarchical and one being associative (cf. Section 2.2.5). The objective of the authors is to build a combined thesaurus, i.e. a combined CO, for Web sites from the same domain. To achieve this, first, the hierarchical CO is extracted individually for each site and, then, the COs are merged into a single thesaurus. Thus, the research problem of extracting hierarchical COs on site-level is addressed in the author's work. The authors distinguish navigation links and semantic links. With the terminology that we have introduced in the $O^3$-model, we can say that semantic links are those links that match the edges in the hierarchical CO, while all other links are navigational links (e.g., links from a child back to the parent node). The approach of Chen et al. is to filter all navigational links. To achieve this, the authors 1) remove links that point to a URL directory higher than the directory of the current page because it is assumed that these links represent child-parent relationships, 2) remove "high-level" navigation bars discovered with the method described in [CZSZ01] because it is assumed that they do not contain downward links and 3) remove navigation lists that occur multiple times because it is assumed that they are not semantically related to the page on which they appear. The used block classification method [CZSZ01] also distinguishes index and content pages. This information is used to separate hierarchical links from associative links: all links on index-pages are considered as being hierarchical, while the associative links are given by the links on the content pages.

Although Rodrigues et al. ([ENMG06, MEMF07]) propose extracting a site structure model based on navigation elements, the resulting model, called link structure graph (LSG), is not hierarchical. The authors distinguish two kinds of navigation elements: so-called s-nodes that are stable in a way that they reoccur on the linked pages and c-nodes that do not fulfil this condition. Obviously, menus that represent hierarchy levels are s-nodes because they usually allow navigating from each sibling to all other siblings in the same hierarchy level. Thus, the hierarchical CO of a site should have a major influence on the LSG. However, the LSG representation is not hierarchical because an edge between two nodes is always drawn if the source node contains a hyperlink to a page on which the target node is present. In addition, the nodes in the LSG represent navigation elements and not pages. Thus, the LSG representation has a different focus and does not correspond to the CO as it is understood in this thesis.

## 4.5.4 Evaluations of CO-Mining Solutions

Chen et al. [CLWP03] remove “navigational” links in order to recover the CO structure. They evaluated the performance of navigational link detection on 13 sites from the shopping domain. For each site, 25 pages were manually labeled. The authors only list precision (about 93% in average), recall values are not provided. How well the remaining links could be separated into hierarchical and associative links is not measured as well. Their method assumes that the semantic links on a page either consist of only hierarchical links (“index page”) or of only associative links (“content pages”). Although this might apply to some pages from the shopping domain, the assumption does obviously not hold in the general case.

Liu et al. [LINL04] conduct a leave-one-out evaluation based on five, manually-labeled sites from the news domain. The interpretation of their result is difficult because they do not directly evaluate the quality of the extracted CO. Instead, they measure the number of correctly identified “navigation pages”, which are hubs that contain links to content pages or other navigation pages located deeper in the hierarchy. However, the results indicate that their approach works very well for the first hierarchy levels of the analyzed pages but performs poorly on the extraction of deeper hierarchy levels.

Yang and Liu [YALI09] evaluate different combinations of graph-generating algorithms and classifiers for estimating the used edge weights. They perform a leave-one-out evaluation with five selected Web sites. Since the evaluation was conducted more than five years ago, the complexity of the tested sites at time of the experiments is difficult to assess. However, a screenshot of one of the sites, which is included in the paper, indicates that the considered sites had a low-complexity and do not represent current Web sites well. Parts of the CO were manually labeled for each Web site. As evaluation metric, the percentage of nodes in the benchmark trees for which the manually assigned parents and the extracted parent nodes match is used. According to this metric, the combination of the decision tree learner and the directed minimum-spanning tree algorithm performed best with an average result of 91.9%.

Yang et al. [YJZN10] as well as Rodrigues et al. [ENMG06, MEMF07] evaluate their methods only indirectly in combination with other tasks.

The hierarchy extraction method of Lin et al. [LICC11] is weakly evaluated. Only three selected sites are considered. All three sites contained sitemap pages, so it was possible to evaluate against the actual CO without labeling effort. An average precision of 0.69 and an average recall of 0.63 are reported by the authors.

## 4.6 Conclusion

We conclude that CO-mining is an unsolved research problem. Although there is some previous work that addressed the problem, practical applicability of previous solutions has not been proven. All reviewed solutions that are directly related to CO-mining (cf. Section 4.5) have in common that they are weakly evaluated. Only one method is evaluated for more than 5 sites and [CLWP03], which considers 13 different sites, reports precision values without the corresponding recall results. In Section 2.3, we have discussed that the CO and the link structure differ significantly and, furthermore, that the application of multiple navigation design patterns defines the FO on top of a CO. In Chapter 3, we have also argued that effective CO-mining requires decoding the navigation design patterns. Based on these premises, we can conclude that the existing CO-mining solutions are limited *by design*. Chen et al. [CLWP03] and Yang et al. [YALI09] work on the level of individual hyperlinks and do not consider link aggregations, i.e. navigational blocks at all. Liu et al. [LINL04] and Lin et al. [LICC11] do mine specific navigation elements but ignore that sites often contain a broad range of different navigation elements and only few of them hold hierarchy information. In addition, their frameworks are obviously not capable of handling the subtle differences between different navigation design patterns that we demonstrated in Section 3.3.

Having found that the CO-mining problem has not yet been solved by previous solutions, we can focus on the research questions formulated in Section 1 and review them under the light of the surveyed work[15]:

1. *Which navigation design patterns can be used for CO mining?*

   Although the methods by Liu et al. [LINL04] and Lin et al. [LICC11] aim at mining navigation elements that contain links to child nodes, the specific characteristics of such navigation elements remain vague in both publications. There are multiple ways of implementing hierarchical navigation and an effective CO-mining approach requires a deeper analysis of the applied navigation design patterns, which is not covered by previous work.

2. *How can those navigation design patterns that allow CO extraction be mined? Are the methods accurate enough to produce valuable hierarchy information?*

   A reliable method for extracting navigation elements that could serve as foundation for an applicable CO-mining method does not yet exit. There is little previous work on this problem statement. None of the reviewed existing solutions has been evaluated for more than five different Web sites and even for those selected sites, the solutions did not work perfectly (cf. Section 4.4.3). There is also no convincing universal block classification method with satisfying performance that delivers specific navigation elements. Methods with focus on main content extraction (cf. Ap-

[15] Work related to the fourth research question is discussed in Section 3.1.

pendix B.2) have been applied with more success, but the extraction techniques are different and cannot be generalized to mine navigation elements.

3. *How can we evaluate the correctness of CO mining solutions?*

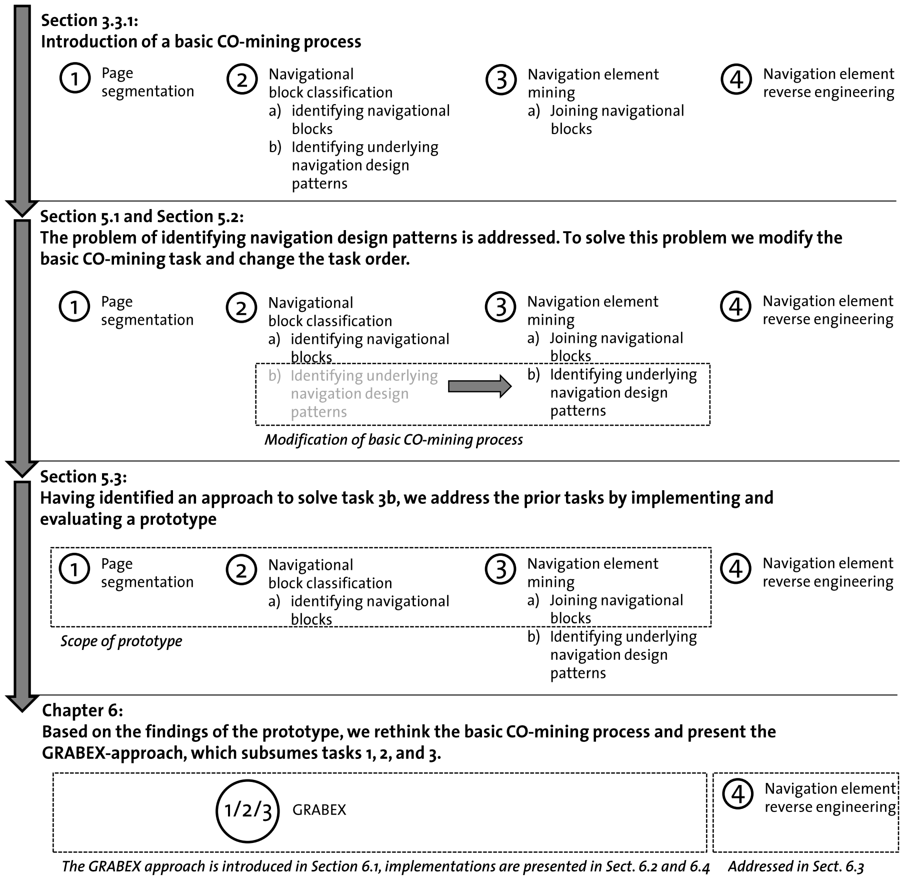
   Related work in general and existing CO mining solutions in particular have been evaluated on usually small, manually labeled data sets. There seems to be no alternative to time-consuming human assessments. However, evaluation data sets that involve less than 5 different sites are too small to draw conclusions on the generalizability of an approach.

# 5 Towards a Solution

In this section, we lay the foundations for an applicable CO-mining solution. The organization of this chapter is depicted in Figure 27. In Section 5.1 and Section 5.2, we introduce a solution that allows identifying the navigation design patterns of navigation elements. For this, we analyze graph representations of the links in the navigational blocks of a navigation element. We refer to these graphs as block graphs. Our approach requires conducting the identification of navigation design patterns not as part of task 2 in the basic CO-miniing process but after the navigational blocks have been joined to navigation elements (task 3 in the basic CO-mining process). In Section 5.3, we focus on the generation of block graphs by implementing a prototypical solution with conventional methods. The evaluation of the prototype reveals problems that are difficult to address with traditional Web mining methods. (Hence, in Chapter 6, we rethink the basic CO-mining process and present the GRABEX-method that solves the first three tasks simultaneously in a single step.)

## 5.1 Representing Navigation Elements as Block Graphs

In this section, we introduce the block graph model. The block graph model is characterized by a fundamental navigation element model and graph structures that are defined based on this fundamental model. In the fundamental navigation element model, navigation elements are modeled as sets of blocks, while blocks are modeled as sequences of links. To actually represent a site with the block graph model, it is required that 1) page segmentation is conducted successfully, 2) all navigational blocks are extracted but their types are not identified and 3) navigational blocks that belong to the same navigation element are grouped. This means that the first three tasks in the basic CO-mining process are completed at least partially: As Figure 28 illustrates, it is assumed that the boundaries of navigational blocks are known. Assumption 1) and 3) correspond to tasks 1 and 3 respectively of the basic CO-mining process (cf. Section 3.3.1). The second task, navigational block classification, is completed only partially because navigational blocks can be distinguished from other blocks



Section 3.3.1:
Introduction of a basic CO-mining process
1 Page segmentation
2 Navigational block classification
a) identifying navigational blocks
b) Identifying underlying navigation design patterns
3 Navigation element mining
a) Joining navigational blocks
4 Navigation element reverse engineering
Section 5.1 and Section 5.2:
The problem of identifying navigation design patterns is addressed. To solve this problem we modify the basic CO-mining task and change the task order.
1 Page segmentation
2 Navigational block classification
a) identifying navigational blocks
b) Identifying underlying navigation design patterns
3 Navigation element mining
a) Joining navigational blocks
b) Identifying underlying navigation design patterns
4 Navigation element reverse engineering
Modification of basic CO-mining process
Section 5.3:
Having identified an approach to solve task 3b, we address the prior tasks by implementing and evaluating a prototype
1 Page segmentation
2 Navigational block classification
a) identifying navigational blocks
3 Navigation element mining
a) Joining navigational blocks
b) Identifying underlying navigation design patterns
4 Navigation element reverse engineering
Scope of prototype
Chapter 6:
Based on the findings of the prototype, we rethink the basic CO-mining process and present the GRABEX-approach, which subsumes tasks 1, 2, and 3.
1/2/3 GRABEX
4 Navigation element reverse engineering
The GRABEX approach is introduced in Section 6.1, implementations are presented in Sect. 6.2 and 6.4
Addressed in Sect. 6.3

**Figure 27. Organization of Chapters 5 and 6.**

but the types of the navigational blocks, i.e. the underlying navigation design patterns, are not known.

By assuming that the first three tasks of the basic CO-mining process are executed successfully and the fourth task is not yet solved, we approach the CO-mining problem in a reverse direction. As we will see in Section 6.1, this makes sense because navigation elements result in characteristic block graph patterns and on this basis we can translate the first three mining task into a single task that is easier to solve – the task of discovering characteristic block graph patterns in the set of all possible page segmentations and navigation element combinations.

In the following, we will assume that a site *S* consists of a set of *z* pages:

$$S = \{P_1, P_2, \dots, P_z\} \tag{5}$$

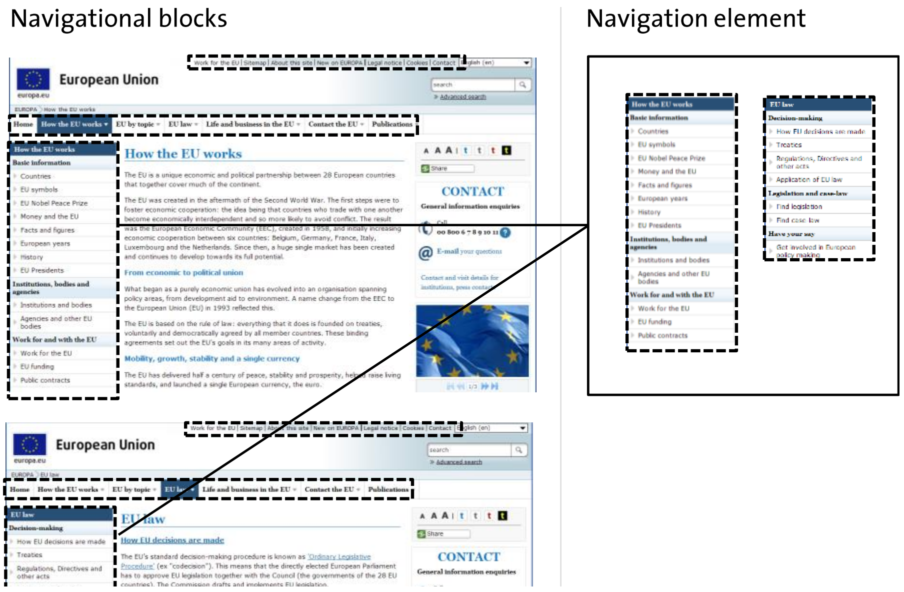


**Figure 28. Requirements of the block graph model. In the block graph model, we assume that the navigational blocks are extracted but their types are not determined (left side). Furthermore, blocks from different pages that belong to the same navigation element are grouped (right side).**

Each page $P_i$ contains a (possibly empty) set of $k_i$ navigational blocks:

$$P_i = \{B_{i,1}, B_{i,2}, \dots\ B_{i,k_i}\} \tag{6}$$

The navigational block $B_{i,l}$, which is the *l*-th block of page *i,* is modeled as a sequence of $m_{i,l}$ linked pages:

$$B_{i,l} = (u_{i,l}^1, u_{i,l}^2, \dots, u_{i,l}^{m_{i,l}}) \text{ with } u_{i,l}^1, u_{i,l}^2, \dots, u_{i,l}^{m_{i,l}} \in S \tag{7}$$

Since we assume a successful completion of the navigation element mining task, the grouping of navigational blocks into navigation elements is known as well and we model a navigation element $N_j$ as a set of navigational blocks:

$$N_j = \{B_{x,y} \in \cup_{P_i \in S} P_i \,|B_{x,y}\, results\ from\ navigation\ element\ N_j\} \tag{8}$$

These are the components of the fundamental navigation element model, as illustrated in Figure 29. In the depicted example, the site $S$ consists of only three pages. On each page, three navigational blocks are found. The first block on each page ($B_{1,1}$, $B_{1,2}$ and $B_{1,3}$ respectively) contains links to the other two pages and, hence, those blocks are modeled as sequence of two pages. All three blocks belong to the same navigation element $N_1$.

The Web graph $W = (V_W, E_W)$, whose vertices represent pages and whose edges are defined by the hyperlinks, is contained within the block graph model. The set of vertices $V_W$ is

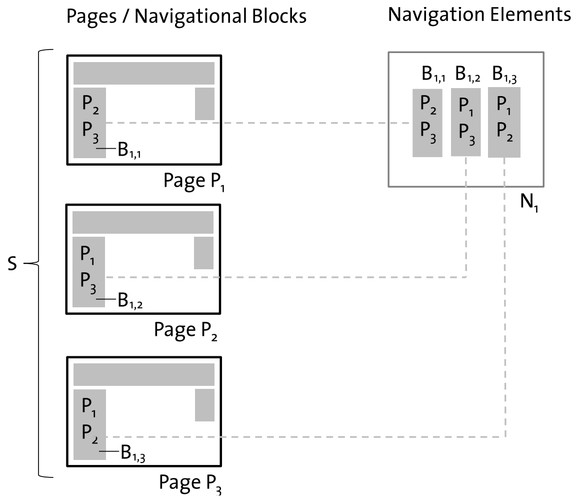


**Figure 29. Illustration of the block graph model.The navigational blocks (gray boxes) of navigation element *N*1 contain links to the other pages of the site (e.g., P2 and P3 on page P1).**

given by the set of pages $S$ and the set of edges can be formulated as follows if $u_{x,l}^{h}$ is used according to formula (7) and denotes the *h*-th linked page of the *l*-th block of page *x*:

$$E_W = \{(P_x, P_y) \in S^2 \mid \exists l, h \in \mathbb{N}: u_{x,l}^{h} = P_y\} \tag{9}$$

This means that $E_W$ contains an edge from page $P_x$ to page $P_y$ if $P_x$ contains a block $B_{x,l}$ of which the *h*-th element is a link to page $P_y$. While the Web graph is contained within the block graph model, the latter provides additional information because it allows splitting the Web graph into separate components that originate from different navigation elements. Hence, we define the *partial Web graph* $W_j = (V_j, E_j)$ that corresponds to the subset of links which result from the navigation element $N_j$. Again, the vertices are given by the set of pages $S$. The set of edges $E_j$ only includes those edges that originate from a block that belongs to $N_j$:

$$E_j = \{(P_x, P_y) \in S^2 \mid \exists l, h \in \mathbb{N}: u_{x,l}^{h} = P_y \wedge B_{x,l} \in N_j\} \tag{10}$$

As we will see in the next section, partial Web graphs allow drawing conclusions about the underlying navigation design pattern. In this context, another graph representation is useful. We call this graph the *partial link graph* $L_j = (V'_j, E'_j)$. As before, the set of vertices $V'_j$ corresponds to the set of pages $S$ and only links within blocks that belong to the navigation element $N_j$ are considered. An edge from a page $P_x$ to a page $P_y$ is drawn, if a block of $N_j$ exists, in whose sequence of linked pages, page $P_y$ immediately follows page $P_x$. For example, the block $B_{i,l} = (u_{i,l}^{1}, u_{i,l}^{2}, \dots, u_{i,l}^{m_{i,l}})$ defines the edges $(u_{i,l}^{1}, u_{i,l}^{2})$, $(u_{i,l}^{2}, u_{i,l}^{3})$, $(u_{i,l}^{3}, u_{i,l}^{4})$ and so on.

Formally, the set of edges $E'_j$ can be specified as:

$$E'_j = \{(P_x, P_y) \in S^2 | \exists z, l, h \in \mathbb{N}: u_{z,l}^{h} = P_x \wedge u_{z,l}^{h+1} = P_y \wedge B_{z,l} \in N_j\} \quad (11)$$

## 5.2 Block Graph Patterns Reveal Navigation Design Patterns

In this section, we demonstrate by examples that the block graph model allows to derive information about the used navigation design patterns.

### 5.2.1 Methodology and Visualization

In the following, we present a selection of patterns that is neither based on 1) a specific design pattern catalogue nor 2) on an empirical evaluation because:

1) There is no final, empirically grounded design pattern catalogue. Existing design pattern catalogues typically represent “best practices” rather than actual snapshots of commonly used patterns. In addition, the granularity of navigation design patterns and block graph patterns differs. Naturally, some navigation design patterns are not distinguishable based on their block graph representation, while in other cases, different block graph patterns are variations of a single navigation design pattern.

2) The block graph model assumes successful page segmentation and detection of navigational blocks as well as navigation elements (but not their types). These premises are open issues (we address them in Section 5.3 and Section 6.1) that hinder an empirical analysis at this stage.

Hence, we discuss a selection of navigation design patterns that are either based on a hierarchical CO or that result in distinctive block graph patterns illustrating the expressiveness of the model. We use the methodology for describing navigation design patterns that was introduced in Section 2.3.2. Since the block graph model is not able to capture information about the PO dimension, patterns that only differ in this dimension result in the same block graph pattern. Although, we mention alternative navigation design patterns that obviously share a presented block graph pattern, we do not claim to present an exhaustive list. The list of examples is a revised version of the examples that we presented in [KENU11]. For each example, we plot the graphs $W_j$ and $L_j$ (cf. previous section). In order to reveal further characteristics, we distinguish four classes of nodes in $W_j$. We define the set $SR_j$ that contains all pages on which a block belonging to the navigation element $N_j$ is present and the set $TG_j$ that contains all pages that are linked from within a block of $N_j$:

$$SR_j = \{P_x \in S | \exists y: (P_x, P_y) \in E_j\} \quad (12)$$

$$TG_j = \{P_x \in S | \exists y: (P_y, P_x) \in E_j\} \quad (13)$$

Since a page can belong to $SR_j$ and $TG_j$ at the same time, we can define four mutually exclusive sets:

1) $SR_j \setminus TG_j$, which contains pages on which the navigation element $N_j$ is present, and that are at the same time not linked from within $N_j$.

2) $SR_j \cap TG_j$, which contains pages on which the navigation element $N_j$ is present and that are linked from within $N_j$.

3) $TG_j \setminus SG_j$, which contains pages on which the navigation element $N_j$ is *not* present but that are linkied from within $N_j$.

4) $S \setminus (SG_j \cup TG_j)$, which contains pages on which the navigation element $N_j$ is *not* present and that are not linked from within $N_j$.´

***Visualization of block graph footprint***: To visualize $W_j$, we separate the canvas on which the graph is drawn into three zones that correspond to the sets 1), 2) and 3). We do not draw nodes that belong to set 4) because those nodes are not of interest for analyzing the navigation element $N_j$. Bidirectional edges are drawn black, in contrast to unidirectional edges, for which we have chosen a lighter gray. In order to not overload the presentation, arrows to illustrate edge directions are shown only if they provide interesting information. For patterns that are based on a hierarchical CO, we assume the CO depicted in Figure 30. We will refer to this kind of visualization as block graph footprint.

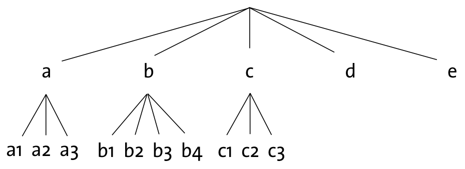


**Figure 30. Assumed hierarchical CO**

### 5.2.2 Pattern Examples

**Horizontal main menu**

First, we discuss the block graph footprint of the navigation design pattern “horizontal main menu” (as introduced in Section 2.3.2):

| *Navigation Design Pattern:* | **Horizontal main menu** |
|---|---|
| *CO-dimension* | Based on a hierarchical CO |
| *FO-dimension* | The pattern generates links to the first level of the hierarchy |
| *PO-dimension* | Links are arranged horizontally at the top of each page |

The resulting graphs $W_j$ and $L_j$ are depicted in Figure 31. We can assume that the main menu is visible on all pages and thus, all pages are source pages, i.e. are contained in $SR_j$. However, only the five first-level pages of the hierarchy are accessible. Hence, these pages belong to $SR_j \cap TG_j$. Since from each page in $SR_j \cap TG_j$, each other page in $SR_j \cap TG_j$ can be accessed, the partial graph $SR_j \cap TG_j$ forms a complete subgraph, i.e. a clique (Figure 31 (2)). The pages in $SR_j \setminus TG_j$ are not connected to each other. In $L_j$, there is a single linear graph consisting of the five first level pages, because only links to these pages appear and they appear always in the same order.

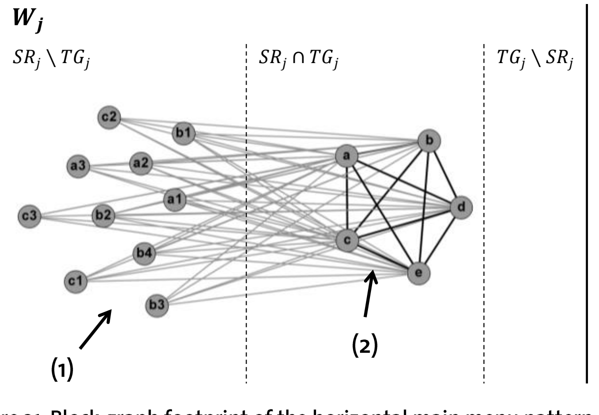


**Figure 31. Block graph footprint of the horizontal main menu pattern.**

*Discussion – navigation design patterns with similar block graphs:* Since the PO-dimension is not captured in the block graph model, horizontal main menus cannot be distinguished from main menus that are placed, e.g., vertically at the left side, solely based on this model. In addition, other, supplementary menus that appear on all pages of a site and always contain the same links result in similar block graph patterns.

### Vertical submenu

We now assume another pattern commonly used to implement hierarchies, which is a submenu placed in the left sidebar of a page:

| *Navigation Design Pattern:* | **Vertical submenu** |
|---|---|
| *CO-dimension* | Based on a hierarchical CO |
| *FO-dimension* | The pattern generates links to an active hierarchy level below the first level. In this context, *hierarchy level* denotes a set of pages that are child pages of the same parent. *Active* hierarchy level means that either the page that contains the menu or an ancestor page of this page belong to the hierarchy level. |
| *PO-dimension* | Links are arranged vertically in the left or right sidebar |

Since three nodes of the first level contain child nodes, there are three submenu “instances”, one for each subtree. Each menu instance generates a clique in $SR_j \cap TG_j$ because from each second level node, all of its siblings are accessible (Figure 32, (2)). There are three nodes, that contain the submenu but that are not accessible via the menu, which are the parents of the second level nodes (Figure 32, (1)). In $L_j$, there are three line graphs, each representing an instance of the submenu.

*Discussion – navigation design patterns with similar block graphs:* Menus that differ only in the PO-dimension from vertical submenus, e.g., horizontal submenus, result in similar patterns. However, the pattern is a strong indicator for a submenu within a hierarchy.

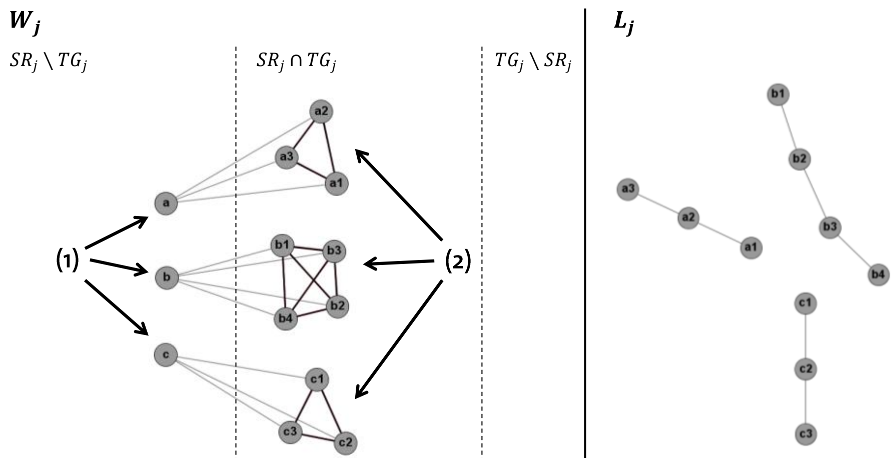


**Figure 32. Block graph footprint of the vertical submenu pattern**

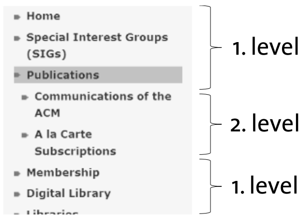


Figure 33. Example of a vertical multilevel menu. All first level nodes are expanded, while submenus are only expanded if the subtree is active. (Source: http://www.acm.org)

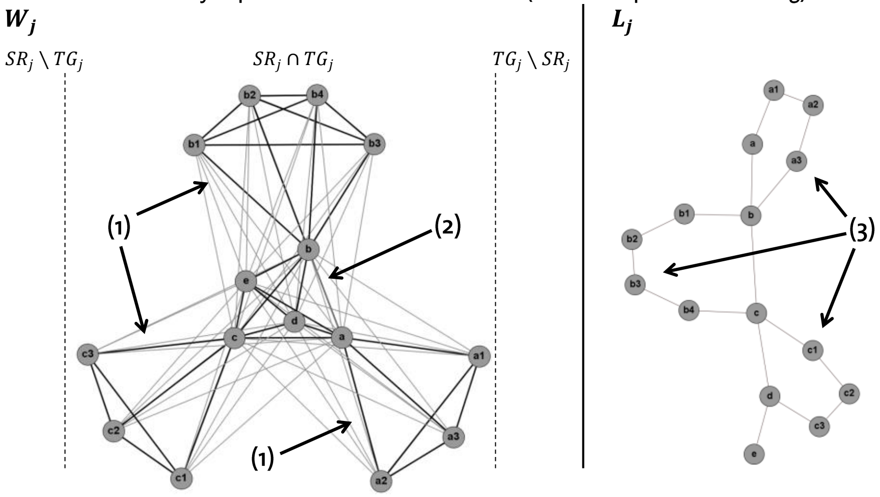


Figure 34. Block graph footprint of the vertical 2-level menu pattern.

### Vertical 2-level menu

Often, instead of separate main and submenus (cf. previous patterns), multiple hierarchy levels are accessible via a single menu. A typical example is a vertical menu in which the first hierarchy level is always visible while only the submenu of the active subtree is expanded (cf. Figure 33):

| *Navigation Design Pattern:* | **Vertical 2-level-menu** |
|---|---|
| *CO-dimension* | Based on a hierarchical CO |
| *FO-dimension* | The pattern generates links to the first hierarchy level and the second hierarchy level if the subtree is active. If the children of a page are expanded, they appear directly below the parent and before the next sibling page. |
| *PO-dimension* | Links are arranged vertically at the left or right side of a page. The submenus are indented or in another way visually separated from parent levels. |

Since the vertical 2-level-menu pattern is present on all pages of the site and allows accessing all pages at the same time, all nodes of $W_j$ belong to $SR_j \cap TG_j$. The pattern results in a flower-like structure consisting of multiple cliques (Figure 34). The central clique (Figure 34, (2)) originates from the first hierarchy level, while the second level pages produce the adjacent cliques (Figure 34, (1)). In $L_j$, the submenus add cycles (Figure 34, (3)) to an otherwise linear graph.

*Discussion – navigation design patterns with similar block graphs:* Based on our current considerations, we believe that this pattern typically appears only in conjunction with a vertical multilevel menu. In case of horizontally-arranged menus, the child nodes are usually not placed in between the first level nodes but below. Hence, $L_j$ would not show the characteristic cycles.

**Multilevel menus**

The vertical 2-level menu pattern is an instance of a broader class of navigation design patterns, which subsumes all menus (typically vertically aligned) that represent subtrees of the hierarchical CO that span multiple levels. The examples a) and b) from Figure 13 on page 25 fall into this category as well as the examples shown in Figure 24 on page 45. There are different patterns of behavior that are discussed in Section 6.3. As explained in Section 5.2.3, the block graph representations of the vertical 2-level menu pattern reflect the fact that the pattern is a combination of a main and a local menu. This generalizes to the multilevel menu pattern class. Multilevel menus can be considered as combination of a main menu and multiple local menus. Hence, they are characterized by multiple cliques in $W_j$ – this is an important aspect, which is utilized by the menu extraction method presented in Section 6.2.

## Breadcrumb navigation

Another navigation design pattern with a distinct block graph footprint is the pattern Breadcrumb navigation as introduce in Section 2.3.2:

| *Navigation Design Pattern* | **Breadcrumb navigation** |
|---|---|
| *CO-dimension* | Based on a hierarchical CO |
| *FO-dimension* | The pattern generates links to all ancestor pages in the hierarchical CO, i.e. pages on the path from the root node to the active page |
| *PO-dimension* | Links are arranged horizontally above the presented content chunk. Separator symbols between the links indicate breadcrumb navigations. Common separator symbols are > and \| |

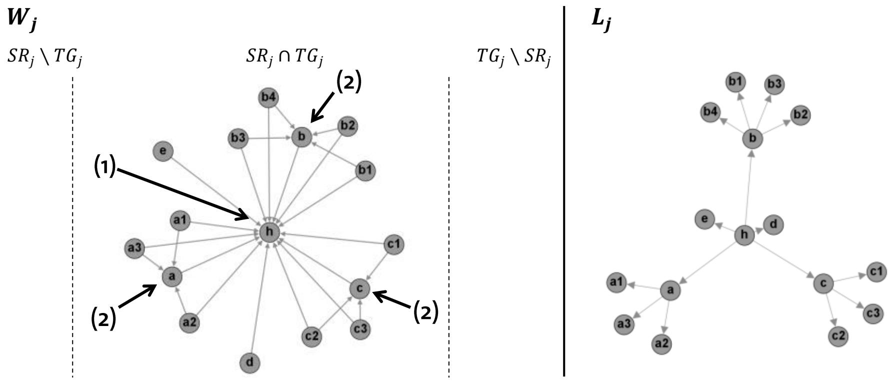


**Figure 35. Block graph footprint of the breadcrumb navigation pattern**

If a breadcrumb navigation shows the active path including the current page and the breadcrumb appears on all pages of a site, all nodes of $W_j$ belong to $SR_j \cap TG_j$. In this example, we assume that the site has an explicit homepage *h* that can be considered as the root node of the exemplary CO (cf. Figure 30)[16]. Since page *h* is an ancestor of all pages, the breadcrumb always contains a link to this page and *h* is characterized by the fact that it has the most in-links (Figure 30 (1)), followed by the first level pages that have child nodes (Figure 30(2)). In case of a breadcrumb navigation, $L_j$ is particularly interesting, since it corresponds to the original CO and thus shows a strict tree structure.

*Discussion – navigation design patterns with similar block graphs:* We believe that the block graph footprint of a breadcrumb navigation is unique and does not appear in combination with other common patterns.

## Sitemap navigation

Many Web sites contain so-called sitemaps, which visualize the hierarchical CO. Sitemaps can be considered as navigation design patterns as well. According to our schema, we can describe the sitemap pattern as follows:

| *Navigation Design Pattern* | **Sitemap navigation** |
|---|---|
| *CO-dimension* | Based on a hierarchical CO |
| *FO-dimension* | The sitemap appears usually on only a single page of the site. The sitemap pattern generates links to all pages in the hierarchy. The order in which the links appear usually reflects a depth-first tree traversal. |
| *PO-dimension* | The sitemap is presented in the main content area. The tree structure is visualized, e.g. by indenting lower tree levels. |

[16]For practical reasons, hierarchical COs are often modeled without an explicit root node in the field of Web information architecture. Instead, the homepage is considered as the first sibling of the first hierarchy level. For simplicity reasons, we adopt this view in this section, except for the Breadcrumb example, where we assume an additional root node to point out the pattern.

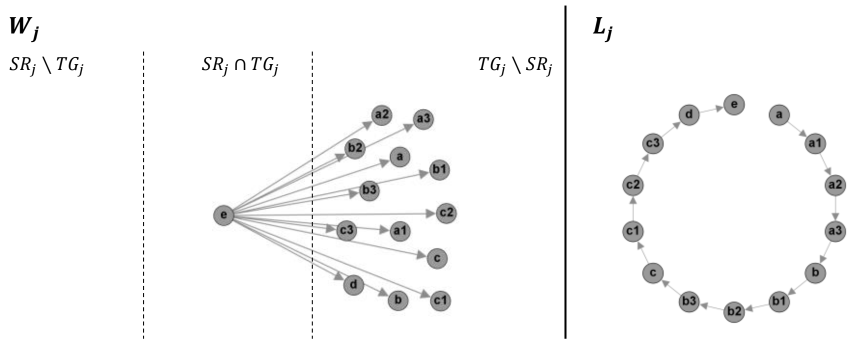


**Figure 36. Block graph footprint of the sitemap navigation pattern**

Since sitemap navigations typically appear only on a single page of each site, $SR_j$ contains only one node. A sitemap visualizes the entire CO, hence, $TG_j$ contains all pages of a site that are part of the hierarchical CO (Figure 36). $L_j$ consists of a single linear graph, defined by the order in which the links appear on the sitemap page. This order usually corresponds to the order of a depth-first traversal of the CO.

*Discussion – navigation design patterns with similar block graphs:* Without additional knowledge and judging only from $W_j$ and $L_j$, the sitemap pattern cannot be distinguished from other patterns, e.g., a list to related information that appears only once on the entire site. However, if the pages that are part of the hierarchical CO are known (not necessarily their hierarchical arrangement and order), the discussed $W_j$ and $L_j$ patterns should indeed allow identifying sitemap navigations.

## Language switch

Many Web sites today are multilingual. Due to maintenance reasons, each language version of a site contains typically the same contents and shares the structure with the other site versions. This means that for each page in one language, corresponding pages in other languages exist that provide the same information. Since users enter a site not only via the homepage but may arrive on an arbitrary page of the site indexed by a search engine, commonly, the language can be selected on every individual page. By "Language switch" we refer to the frequent navigation design pattern that allows switching between pages that contain the same information but in different languages (Figure 37). The pattern can be specified as follows:

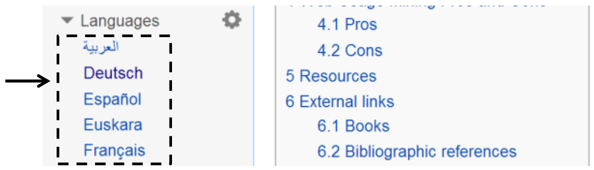


Figure 37. Language switch in Wikipedia

| *Navigation Design Pattern* | **Language switch** |
|---|---|
| *CO-dimension* | Based on multiple parallel COs (e.g. hierarchies) with similar structure. Each of the parallel COs represents one language. Each page in one CO has a single corresponding page in each of the other COs (which represent the alternative language version of the page). |
| *FO-dimension* | On each page, links to all alternative language versions of the page are presented. |
| *PO-dimension* | The language switch can appear in the header, in the footer or in a sidebar. |

Since the language switch is present on all pages and at the same time all pages are navigable from other pages, $SR_j \cap TG_j$ contains all pages of the site. $W_j$ is divided into a large number of isolated cliques. Each clique represents pages with the same content in different languages. This is due to the fact that a language switch alone allows users only to set the language of a page but not to proceed to pages with other information. $L_j$ exposes a similar structure, except for the edges, which are not bidirectional if there is a fixed order in which the languages are listed on each page.

*Discussion – navigation design patterns with similar block graphs:* Currently we are not aware of any confusable patterns and believe that the language switch pattern results in distinctive graph structures.

## Context switch

In the language switch pattern example, we assumed that users can switch between alternative language versions of a page. However, pages might not exist in all languages and sometimes, the entire site structure and content may differ in dependence of the language version. Hence, if users on an arbitrary page want to switch to another language, they are directed to the homepage of the alternative sub site because a corresponding page does not exist. This kind of pattern for switching between otherwise separated sub sites is

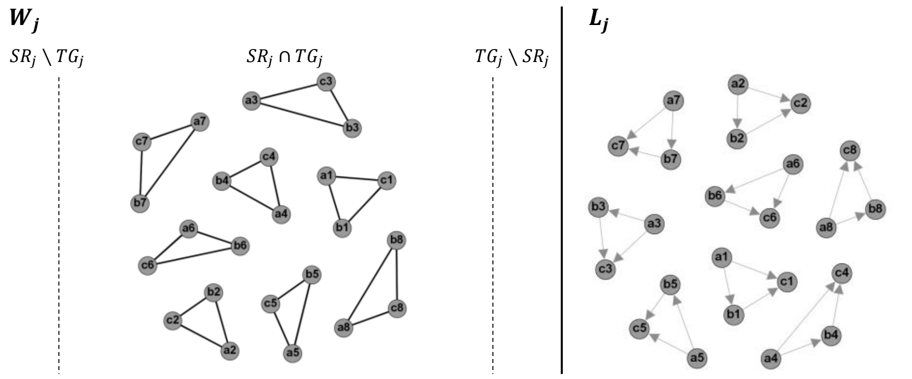


**Figure 38. Block graph footprint of the language switch pattern**

also applied in the context of use cases than language selection. For example, there might be separate sub sites for different brands of a company that are connected in this way. We will refer to this pattern as context switch. The detailed description is as follows:

| *Navigation Design Pattern* | **Context switch** |
|---|---|
| *CO-dimension* | Based on multiple parallel COs (e.g., hierarchies). The structure of the COs can differ. |
| *FO-dimension* | On each page, links to the entry page of each alternative CO are presented. The entry page is, e.g., the root node of a hierarchical CO or the first page in a sequential CO. |
| *PO-dimension* | Context switches typically appear in the header of a page |

Figure 39 visualizes the context switch pattern in case of two alternative sub sites, i.e. COs. The context switch is present on all pages of the site, hence, all pages belong to $SR_j$. We have two alternative COs, represented by the pages *a1-a7* and *b1-b7* respectively in the figure. On each page, the entry page of the alternative CO is linked (Figure 39(2)). Hence, both entry pages constitute the set $SR_j \cap TG_j$. In the case of only two alternative COs, $L_j$ is empty because the context switch contains only a single link on each page.

*Discussion – navigation design patterns with similar block graphs:* Currently, we are not aware of alternative patterns which result in similar block graphs and do not represent some kind of context switch. However, the patterns themselves do not reveal the type of a context switch, e.g., whether it is used for switching between different brands or languages.

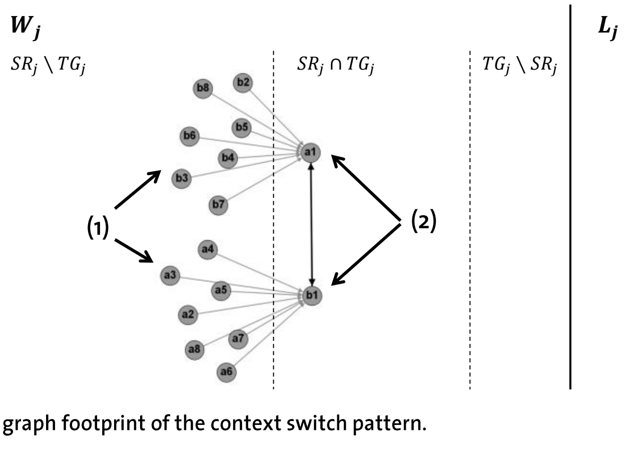


**Figure 39. Block graph footprint of the context switch pattern.**

## Pagination

In Section 2.3.2, we have introduced the pattern pagination as follows:

| *Navigation Design Pattern* | **Pagination** |
|---|---|
| *CO-dimension* | Based on a sequential CO |
| *FO-dimension* | The pattern generates links to a fixed number of direct predecessors and direct successors of the active element in the sequence. Optionally, a link to the first and last element of the sequence is provided. |
| *PO-dimension* | Links are arranged horizontally above or below the presented content chunk. |

We now assume an implementation of the pagination pattern in which links to the immediate predecessor and successor of the active page are displayed (typically labelled with previous / next). The resulting patterns in $W_j$ and $L_j$ for a sequence *a1-a10* are depicted in Figure 40. Since all pages contain the pagination and all pages of the sequence are accessible, all pages are contained in the set $SR_j \cap TG_j$. $W_j$ is a linear graph that reproduces the original linear CO. If not only links to the immediate neighbors of a page are rendered but, e.g., also to the two preceding and the two successive pages, $W_j$ will contain additional edges. However, these edges could be easily eliminated to reproduce the original order. $L_j$ shows two separate linear graphs, one containing pages with even numbers and one containing pages with odd numbers. This results from the fact that, for instance, on an even page, a link two the previous page is followed by the link to the succeeding page, both being odd.

*Discussion – navigation design patterns with similar block graphs:* We believe that the described pattern is a reliable indicator for navigation over sequential structures. However, this also includes multiple more specific patterns such as the “guided tour” or “purchase process” patterns, which have been discussed as individual types.[17]

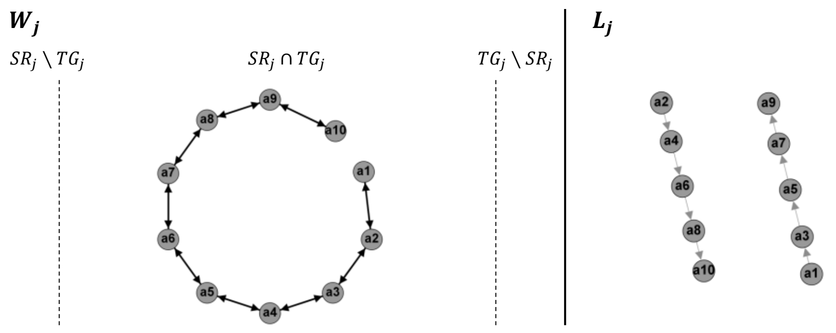


**Figure 40. Block graph footprint of a pagination pattern**

[17] See, e.g., http://www.welie.com/patterns

### 5.2.3 Compound Patterns

As demonstrated in the previous section, block graph patterns can reveal the underlying navigation design patterns. Since all block graphs of a site share the same underlying COs, the block graph patterns of a site are interrelated. Analyzing the relationships between different patterns – or studying patterns of patterns – can provide additional signals for the detection and classification of navigation design patterns. A detailed survey of interacting patterns is out of scope of this thesis. Instead, we provide two examples to illustrated how patterns relate. In the remainder of this thesis, interacting patterns are discussed in the context of specific problem statements.

**Example 1** – Given a main menu *j* and a submenu *i*, the combined graph $W_j \cup W_i$ equals the multilevel pattern[18] (Figure 41). This relationship is intuitive since a multilevel menu is in fact a combination of a main and a submenu. However, even trivial relationships can have implications on pattern analysis: If we assume that a submenu appears only in combination with a main menu, block graphs could be examined pairwise. Instead of searching separately for main and submenus, we can search for navigation element pairs that follow the multilevel menu pattern.

**Example 2** – Given a breadcrumb *j* and a multilevel menu *i*, then $L_j \subset W_i$ (Figure 42)[19]. Moreover, the structure of $L_j$ can be inferred from $W_i$, and $W_i$ is at least partially determined by $L_j$ (except for the nodes *d* and *e* that do not have child nodes). In other words, if we have found a possible multilevel menu pattern but some uncertainty remains, an identified breadcrumb pattern can confirm this interpretation – and the other way around. Hence, the consideration of interacting block graph pattern can increase classification performance. However, such a combined classification is out of scope of this thesis and considered as future work.

[18] We ignore the PO-dimension (e.g., vertical or horizontal alignment) in this context, since it has no influence on the block graph patterns

[19] In this example, we do not model an explicit root node (see footnote 16)

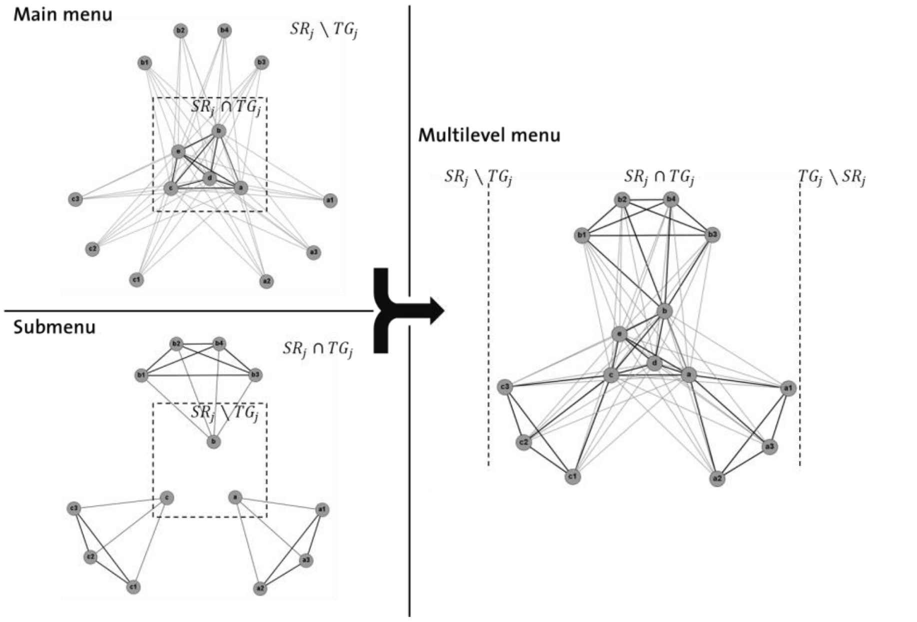


Figure 41. A multilevel menu corresponds to the combined block graph of a separate main and submenu.

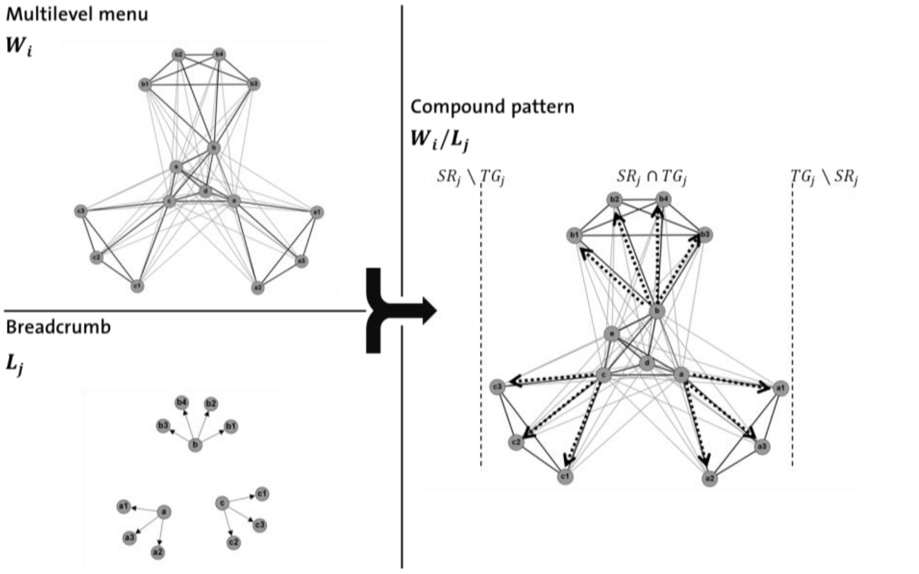


Figure 42. The link graph $L_j$ of a breadcrumb is a subset of the Web graph $W_i$ of a multilevel menu.

## 5.3 Exploring the Limits of Block Graph Generation with Conventional Methods

To generate the block graph model, navigational block classification has to be solved as well as page segmentation and navigation element mining (cf. Figure 27, page 74). In this section, we explore to which extent this can be achieved with conventional methods. We implement methods described in related work whenever it appears to be reasonable and rely on simple straightforward techniques if they seem to be more suitable. Following previous solutions, the prototype presented in this section requires visual rendering of Web pages to access specific attributes such as block widths and heights. The GRABEX-approach presented in Section 6.1 succeeds without page rendering, which boosts runtime performance.

Since main menus, submenus and multilevel menus are the most fundamental solutions for implementing hierarchical navigation, we focus on the corresponding block graph patterns. We subsume the navigation elements implementing the first two levels of a hierarchical CO under the term *main navigation system*, regardless of whether they are implemented as multilevel menu or as main-/submenu combination. Parts of the contributions presented in this section have been published in [KENU11]. In [KENU11], we have also evaluated a simple rule-based approach for navigation element reverse engineering (the fourth task of the basic CO-mining process). This approach will not be discussed in this section, since a revised and extended approach for rule-based reverse engineering of menus is presented in Section 6.3.

### 5.3.1 Implementation of a First Prototype

In Section 5.3.1.1, we explain that the block graph generation scenario results in relaxed requirements for page segmentation and navigational block extraction in comparison with more general scenarios. This has influence on implementations of page segmentation and block extraction, which are described in Section 5.3.1.2 and Section 5.3.1.3, respectively. In Section 5.3.1.4, details of a prototypical solution for navigation element mining are presented. An experiment to evaluate the accuracy of the solution is described in Section 5.3.2.

#### 5.3.1.1 Page Segmentation and Navigational Block Extraction Strategy

From the analysis of existing work on page segmentation in Section 4.3 and block classification in Section 4.4, we concluded that no general solutions to these problems exist up until now. Moreover, there are few comparative evaluations of existing solutions that target specific use cases. At the same time, evaluating existing methods is difficult due to the lack of published implementations and out of scope of this thesis. As a result, it is not possible to determine the most suitable page segmentation and block classification solution for our purpose in an easy way.

However, we are only interesting in isolating and recognizing navigational blocks correctly and do not require an accurate segmentation of the entire page and an accurate classification of all page blocks. Besides, we do not necessarily require the exact set of navigational blocks. Additional, “noisy” page blocks can be tolerated. This results from the assumption that the block graph model allows distinguishing the main navigation system from other navigation element types. Hence, additional, “faulty” page blocks do not have negative effects as long as they are not combined into navigation elements that accidently resemble block graph patterns of the main navigation system. In principle, we can work with a superset of page blocks as long as this superset contains the true navigational blocks. However, we should try to keep this superset as small as possible for three reasons: 1) to reduce runtime overhead of downstream processing, 2) to keep the effort of manual evaluation low, i.e. to limit the number of navigation elements that need to be inspected, and 3) to avoid negative effects on the accuracy of downstream tasks. Additional noisy page blocks could complicate joining navigational blocks to navigation elements and, as aforementioned, could lead to the accidental appearance of typical block graph pattern.

Since all reviewed page segmentation methods except for [CAML10] return subsets of the DOM elements, in which each DOM element represents a page segment, the set of all DOM elements of a page defines an adequate superset. Thus, the fundamental strategy behind both, our page segmentation solution and the navigational block extraction method, is to remove DOM elements that are unlikely to represent navigational blocks.

### 5.3.1.2 Page Segmentation Algorithm

Figure 43 visualizes the fine-grained page segmentation defined by the DOM elements of a sample page. This structure still contains much redundancy with respect to our purpose: We are only interested in navigational blocks and, hence, can ignore blocks that do not

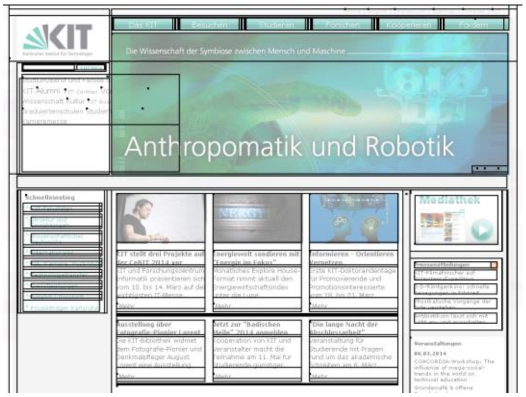

KIT
Das KIT
Besuchen
Studieren
Forschen
Kooperieren
Fördern
Die Wissenschaft der Symbiose zwischen Mensch und Maschine
Anthropomatik und Robotik
Schnelleinstieg
Mediathek

**Figure 43. Segmentation defined by the original DOM-tree of a page (source: http://www.kit.edu)**

contain hyperlinks. Furthermore, we can tolerate small variations of the assumed block boundaries as long as they do not affect the hyperlinks within the block (because the block graph presentation is only based on the hyperlinks in each block). Based on these observations, we can simplify the original DOM-tree to reduce the superset without eliminating the true navigational blocks. By this, we reduce the runtime of all downstream processing and limit the required effort of manually inspecting the results. The DOM-tree simplification is conducted by processing the tree bottom-up and applying the following rules:

- If a node is a leaf node and a hyperlink (a-element): keep node
- If a node is a leaf node and not a hyperlink: remove node
- If a node has a single child: remove node but append child node to parent node
- If a node has multiple child nodes: keep node

The effect of this simple graph transformation algorithm is illustrated in Figure 44. The algorithm prunes subtrees whose leaf nodes are not hyperlinks and keeps only those internal nodes at which the tree branches. In case of an internal node with a single child, the node and its child node both represent the same aggregation of links. Hence, we consider the node as redundant and remove it.

#### 5.3.1.3 Navigational Block Extraction

After the DOM-tree transformation, we can consider a-elements that share the same parent node as potential navigational blocks (e.g., blocks *a*, *b* and *c* in Figure 44). Furthermore, we can assume that very large blocks, blocks with much non-linked text and blocks in which the proportion of linked text is small are unlikely to be navigational blocks[20]. The evaluation framework presented in Section 5.3.2.1 was used to find appropriate thresholds and to derive the following rules for eliminating page blocks:

- **Elimination Rule 1:** Screen size of element exceeds 70% of the size of the entire page
- **Elimination Rule 2:** Length of non-hyperlinked text exceeds 150 characters
- **Elimination Rule 3:** More than 50 percent of the text is not hyperlinked

We found that by solely applying these three rules many obviously noisy blocks remain. In contrast to general block classification methods, we are only interested in specific blocks that aggregate a number of other items, the links. Since users must be able to recognize which links belong to the same navigation element, we can assume that all links of the same navigational block have a similar formatting (cf. law of similarity, [WERT38]). However, there are many visual attributes that could be considered for comparing the formatting similarity of two elements, e.g. font face, font size, colors, positions, et cetera. First experimental results showed that not all of these attributes are good indicators of whether two

[20] By linked text we refer to text that is part of a hyperlink, i.e. text below an *a*-element. All other text is referred to as non-linked text.

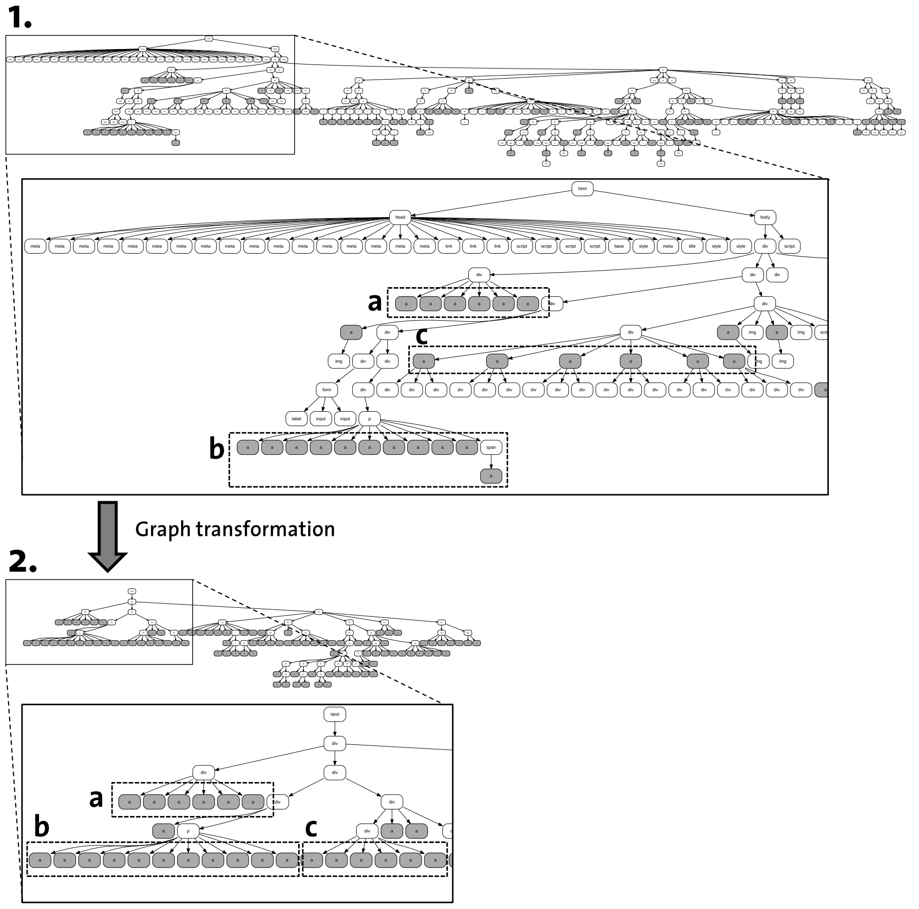


**Figure 44. DOM-tree simplification example**

links belong to the same navigational block or not. For example, in one and the same navigational block, link colors may indeed vary depending on the link target. In addition, the same font type and font size is often used throughout the entire page and, hence, both properties do not work well as distinguishing features. We found that position and size attributes are more reliable characteristics. Thus, we added a fourth rule which is based on comparing the heights, widths, vertical positions and horizontal positions of the links in a navigational block candidate. As we argued in Section 5.3.1.1, removing a true navigational block is worse than keeping a faulty block. For this reason, we do not require that all child nodes are equal in all four attributes. Instead, we formulate the fourth rule as follows:

– **Elimination Rule 4:** No pair of child nodes exists that are equal in at least two of the following properties: x-offset, y-offset, width and height.

#### 5.3.1.4 Navigation Element Mining

To generate block graph models, knowing navigational blocks is not sufficient. In addition, navigational blocks that belong to the same navigation element must be identified – this task was introduced as navigation element mining in our basic CO-mining process (Section 3.3.1). Analyzing related work (Section 4.4.2), we found that only Rodriguez et al. [ENMG06][MEMF07] addressed this problem previously. The authors' approach is based on link comparison. Page blocks are pairwise compared by applying a similarity-measure introduced by Jaccard [JACC12] and commonly referred to as Jaccard-index in other publications. The Jaccard-index is popular in the field of citation-analysis (e.g.,[SCZS06]) and allows comparing sets with different cardinalities. Given two sets A and B, the Jaccard-index is defined as:

$$SIM_{Jac}(A,B) = \frac{|A \cap B|}{|A \cup B|} \qquad (14)$$

Rodriguez et al. [ENMG06] consider two navigational blocks as similar if the Jaccard-index is 1, i.e. if both sets are similar. Obviously, this approach is not suitable for the navigation element mining task in our scenario since we do not assume that the navigational blocks of a navigation element all contain the same links. At least, we have to allow for some fuzziness and lower the threshold. Furthermore, we found that a modified similarity-measure is more appropriate:

$$SIM_{Mod}(A,B) = \frac{|A \cap B|}{\max(|A|,|B|)} \qquad (15)$$

If $A \subseteq B$ or $B \subseteq A$, then $SIM_{Jac} = SIM_{Mod}$ as we show for the case $A \subseteq B$, which implies $|A| \leq |B|$ and $|A \cup B| = |B|$:

$$SIM_{Jac}(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|B|} = \frac{|A \cap B|}{\max(|A|,|B|)} = SIM_{Mod}(A,B) \qquad (16)$$

The modified similarity measure is motivated by the multilevel menu pattern, in which the links of the navigational blocks change in dependence of the active page (cf. Section 5.2.2). This pattern is important for mining hierarchical COs because it is frequently used and provides rich hierarchy information. If we consider this pattern, we can say that the Jaccard-index and the modified index do not differ if a submenu is expanded in one navigational block and collapsed in the other (Figure 45-1). However, the Jaccard-index is lower than the

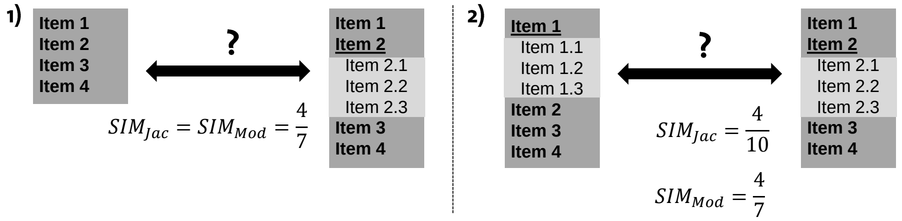


**Figure 45. The used modified similarity measure and the Jaccard-index differ if $|A \setminus B| > 0 \wedge |B \setminus A| > 0$**

modified index if different submenus are expanded in both navigational blocks (Figure 45-2). In such cases, the Jaccard-index tends to drop below any reasonable threshold while the modified version is more likely to reflect the similarity of both menus. Experimenting with different thresholds, we found that a value of 0.5 works well.

However, not all navigation design patterns result in navigational blocks that overlap with respect to the hyperlinks they contain. For example, the language switch pattern (cf. Section 5.2.2) generates different links on each page. While the links differ, the link texts stay the same (the names of the alternative languages). Hence, we apply the modified metric also to compare the link texts. In preliminary experiments, we found that a slightly higher threshold should be used and set the threshold to a value of 0.6.

First tests revealed that both criterions alone lead to faulty associations between navigational blocks because of accidental overlappings. Considering how humans identify reappearing navigation elements, we can say that users solely rely on visual features. This includes two aspects: 1) the consistent visual design of the navigation element on all pages and 2) a more or less fixed position of the navigation element on all pages. Again, both aspects are fuzzy: visual design may vary to some extent, e.g., because of topic-depended color-coding or additional hyperlinks and positions may change, for instance, because of different page layouts and contents. Dealing with this fuzziness is complicated by the number of visual attributes that matter (e.g., vertical and horizontal position, width, height, colors, fonts, border, etc.) and that interact as well (cf. Section 3.3.4). However, those dimensions can be reduced if common Web standards are taken into account and a rational designer is assumed. Rational designers separate content and presentation [WC00] and store both in different files. Moreover, they reuse HTML code across pages whenever possible by defining page templates or at least reusing specified formatting rules to reduce their own implementation effort. This allows us to assess whether two blocks from different pages are likely to be similar with respect to the visual design and position based on the HTML-structure alone, even if the actual visual characteristics are stored in a separate document. To achieve this, we compare what we call the selector token paths of two HTML elements. The selector path of an HTML-element is defined by all tag names, CSS-class-attributes and id-attributes of the element itself and all ancestor elements (Figure 46). Selector token paths reflect the most common types of CSS selectors (cf. [KENU10]). To compare two paths, tokens with the same tree positions are compared pairwise (e.g., using the labeling of Figure 46, the t0.0-token of a block is compared to the t0.0-token of another block, the t1.0-token to the other t1.0-token and so on). The number of discrepancies is summed up and used as similarity-measure. The lower the sum of discrepancies, the more likely both blocks have a similar visual appearance and are placed at similar page positions because:

a) **Position:** In general, there are many possible DOM tree structures for a single page design. However, we assume that rational designers reuse HTML structures and pages with similar visual design also have similar DOM trees. As a result, page blocks located at similar positions on different pages have similar ancestors.

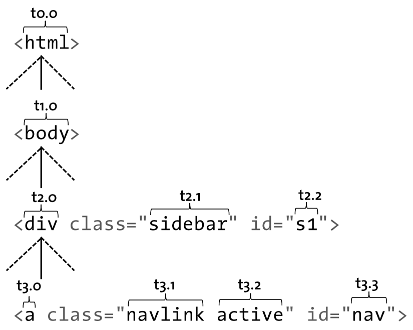

t0.0
<html>
t1.0
<body>
t2.0
t2.1
t2.2
<div class="sidebar" id="s1">
t3.0
t3.1
t3.2
t3.3
<a class="navlink active" id="nav">

**Figure 46. The element types, class-attributes and id-attributes of an element and its ancestors define the selector token path.**

**The individual tokens are labelled t0.0-t3.3.**

b) **Visual appearance:** CSS-definitions are abstract presentation rules that are not directly associated with specific HTML-elements. Instead, they describe how HTML-elements with specific attributes shall be rendered. The attributes of HTML-elements to which CSS-definitions refer are typically HTML-element types, class-attributes and id-attributes (the three attributes used in the token paths). Furthermore, CSS-definitions can refer to the DOM-hierarchy in order to specify the scope of rules. For instance, a CSS-definition may set the font color for all elements of type *a* that are descendants of elements with the class-attribute "maincontent". Hence, the element types, CSS-classes and ID-attributes of ancestors also influence the visual presentation of a page block and two page blocks are likely be formatted in a similar way if their selector token paths are similar.

Based on preliminary experiments, we set the tolerated number of discrepancies between the selector token paths of two navigational blocks that belong to the same navigation elements to 4. In summary, if we denote the selector token path of a navigational block *A* as $STP_A$and the function returning the number of discrepancies as $DIFF$, we consider two navigational blocks *A* and *B* as belonging to the same navigation element if:

$$(SIM_{Mod}(Links_A, Links_B) > 0.5 \vee SIM_{Mod}(Texts_A, Texts_B) > 0.6) \wedge DIFF(STP_A, STP_B) \leq 4 \qquad (17)$$

If the above expression is true for page blocks *A* and *B* as well as for *B* and *C*, it follows that also *A* and *C* belong to the same navigation element, even if the expression evaluates to false for *A* and *C*. To improve runtime performance, we only compare page blocks of pages that are connected by a hyperlink.

### 5.3.2 Evaluation

In Section 5.3.2.1, we describe a dedicated visualization tool that was developed to allow an efficient inspection of the quality of the extracted information. The experimental setup and the results of the empirical study are discussed in Section 5.3.2.2.

#### 5.3.2.1 Result Visualization

For evaluating the prototypical solution, we need to assess whether tasks 1, 2 and 3 of the basic CO-mining process have been conducted successfully for the main navigation system. In detail, we must be able to answer the following questions:

*A) Page segmentation:* Were the boundaries of the blocks belonging to the main navigation system correctly identified?

*B) Navigational block classification:* Were the blocks of the main navigation system recognized as navigational blocks?

C) *Navigation element mining*: Were the navigational blocks of the main navigation system grouped successfully?

Question A and question C require inspecting all page blocks belonging to the main navigation system. To assess whether the boundaries were correctly identified (question A), the inspection must be based on the visually rendered page blocks. In addition, we have to manually scan all navigation elements in search of the main navigation system to answer question B.

Hence, visual inspection of a large number of elements and their relationships is necessary to identify errors. Manually browsing the actual Web sites and comparing the visual presentation with the extracted data would be too time-consuming and hinder efficient development and evaluation. We avoided this limitation by developing a dedicated presentation that can be efficiently inspected with the use of a zoomable user interface. The used Web crawler was extended to render each crawled page and to save a screenshot. In addition, the position (in pixels) of every DOM-element was saved. This allowed us to generate a cropped screenshot of each page block ex post. To visualize the grouping of navigational blocks, we applied the graph visualization software Graphviz[21] for arranging the screenshots of the page blocks, which we used as node symbols. To allow assessors to quickly explore large graphs, the results were displayed with the graph browsing toolkit ZGRViewer[22].

Figure 47 depicts the structure of the generated diagrams. The mining results of an entire site are summarized on a large canvas (Figure 47-a). Using ZGRViewer, assessors can pan in any direction with the desired speed and adjust the zoom-level using the mouse-wheel. On the canvas, groups of page blocks are horizontally arranged. The groups, which represent navigation elements, are visualized by bounding boxes (Figure 47). In addition, a number plotted on each page block (not shown in the figure) indicates the number of times the page block appeared.

[21] Available from http://www.graphviz.org/
[22] Available from http://zvtm.sourceforge.net/zgrviewer.html

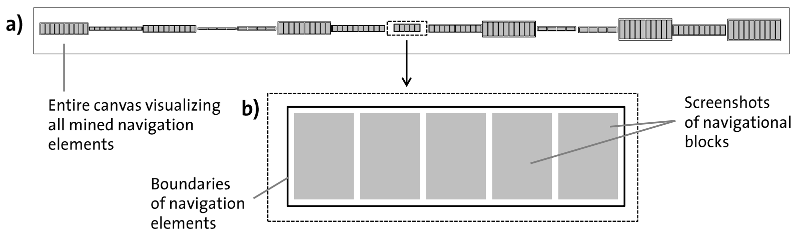

a)
b)
Entire canvas visualizing all mined navigation elements
Screenshots of navigational blocks
Boundaries of navigation elements

**Figure 47. Visualization of the navigation element extraction results. All mined navigation elements were arranged on a large canvas (a). Human assessors can zoom in to inspect individual navigation elements (b).**

#### 5.3.2.2 Experiment

To evaluate the prototypical implementation, we first generated a list of 50 domains as follows: An English dictionary was used to create a list of 50 random keywords. Each keyword is passed as search query to a search engine and the first site that did not violate one of the following criteria was manually selected:

- The site is hierarchically organized and the hierarchy has at least two levels.
- There are no other dominant types of organization schemas as, e.g., in wikis (associative schema) or directories (database schema)
- The site does not use frames or Javascript-based menus

This list of domains was then automatically crawled. For each site up to a maximum number of 250 pages were retrieved. We used the evaluation tool described in Section 5.3.2.1 to assess the quality of navigational block extraction (including the correctness of the boundaries) and navigation element mining (i.e. correctness of grouping of navigational blocks). We evaluated whether the tasks were conducted without any errors, with minor errors or with major errors:

- Errorless: The task was conducted without any errors for the entire sites
- Minor errors: The task was conducted with errors that would result in single additional or missing links in the hierarchical CO, while the fundamental structure of the main navigation system is intact.
- Major errors: The extraction of the main navigation system failed.

If one task leads to major errors, subsequent tasks will fail, too. Hence, sites with major navigational block extraction errors were excluded from navigation element mining. The results are summarized in Table 7. For the second task, we provide the percentage of major errors excluding the sites that produced major errors in the previous tasks at the one hand and the aggregated major errors, representing the overall rate of major errors.

**Table 7. Summary of evaluation results.**

| | | |
|---|---|---|
| **Page segmentation and navigational block extraction** | No errors | 78.0% |
| | Minor errors | 16.0% |
| | Major errors | 6.0% |
| **Navigation element mining** | No errors | 70.2% |
| | Minor errors | 21.3 % |
| | Major errors | 8.5% |
| **Both tasks combined** | Aggregated major errors | 14.0% |

### 5.3.3 Conclusions

The prototype was designed to evaluate how challenging the generation of block graph models is. The implemented solution represents the current state-of-the-art as analyzed in Chapter 4 with some extensions, whenever more appropriate techniques were available (e.g., dedicated page segmentation, comparison of selector token paths instead of visual attributes, etc.). The evaluation results are ambivalent: We made some restrictions on the evaluated sites that excluded challenging samples from the beginning, e.g., sites that are not dominated by a hierarchy at all or sites that rely on Javascript-based menus. And still, for almost 1/3 of the sites, the hierarchy extraction was erroneous. This error rate is too high for most applications that could make use of hierarchy information. A major problem is the fact that the errors of each individual task sum up in the end. Based on the empirical results and the experiences from the prototype development, we gained further insights:

- Page segmentation and navigational block extraction work well and caused few major errors. Most observed errors resulted from too large blocks causing additional links to be included in navigational blocks that are did not result from the same navigation design pattern.
- Navigation element mining seems to be much harder to solve. If a single pairwise comparison of navigational blocks results in an error, the entire results can become useless because two navigational elements are wrongly interpreted as a single one.

To summarize, the implemented page segmentation method can serve as foundation for future CO reverse-engineering solutions. In contrast, the used link-based similarity measures do not promise to solve the problem of navigation element mining well enough to be practically applied.

# 6 The GRABEX-Approach

In this chapter, we introduce the **Gra**ph-based **b**lock **ex**traction approach (GRABEX) for extracting navigation elements (Section 6.1). For applying the GRABEX-approach, we focus on the most prominent navigation design patterns that are based on hierarchical COs in order to extract hierarchy information from as many sites as possible. These patterns are:

1) *Hierarchical menus, which subsume the main/local menu and the multilevel menu pattern* –We describe a GRABEX-application to mine these patterns (Section 6.2) and a rule-based solution for reverse engineering the CO (Section 6.3).

2) *Breadcrumb navigations* – We describe a GRABEX-application (Section 6.4). Deriving the hierarchy is trivial once the breadcrumbs are extracted correctly.

Besides the fact that the GRABEX-method allows reverse-engineering COs, it has other advantages over current page segmentation and block classification methods, since current methods for page segmentation and block classification exploit different kinds of features. For example, 11 out of 18 reviewed page segmentation solutions (cf. Section 4.3) use features that are based on the position and size of elements and 14 solutions evaluate the types of HTML-elements (e.g., table, div, etc.). Both kind of features result in drawbacks: to compute sizes and positions, computationally expensive page rendering is required and relying on the semantics of HTML-elements to conduct page segmentation results in a strong dependency on current Web standards and their usage. Given the rapid change in trending technologies in the field of Web development, such solutions can become obsolete within a few years.

In contrast, the GRABEX-method is tag-agnostic and does not require page rendering. Furthermore, only HTML files are required. CSS files, script files or images are not download-ed.

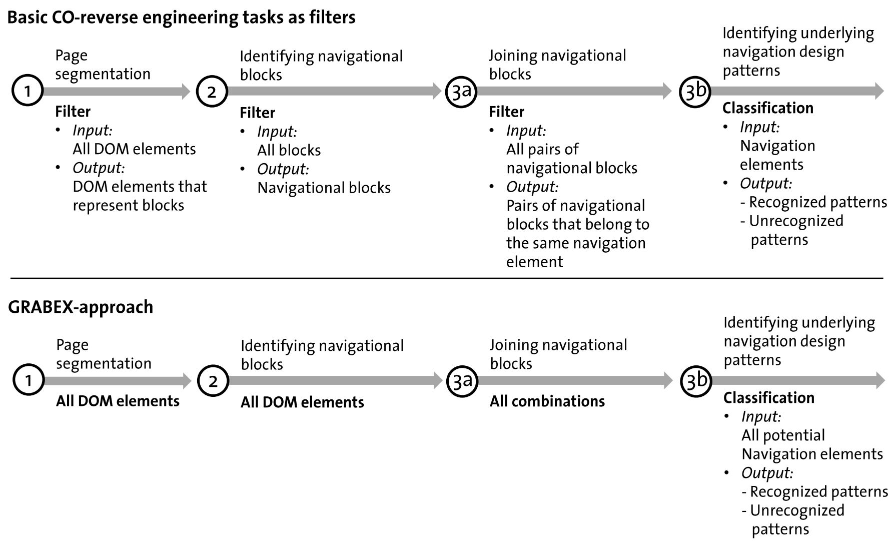

Basic CO-reverse engineering tasks as filters
1
Page segmentation
Filter
• Input: All DOM elements
• Output: DOM elements that represent blocks
2
Identifying navigational blocks
Filter
• Input: All blocks
• Output: Navigational blocks
3a
Joining navigational blocks
Filter
• Input: All pairs of navigational blocks
• Output: Pairs of navigational blocks that belong to the same navigation element
3b
Identifying underlying navigation design patterns
Classification
• Input: Navigation elements
• Output: - Recognized patterns - Unrecognized patterns
GRABEX-approach
1
Page segmentation
All DOM elements
2
Identifying navigational blocks
All DOM elements
3a
Joining navigational blocks
All combinations
3b
Identifying underlying navigation design patterns
Classification
• Input: All potential Navigation elements
• Output: - Recognized patterns - Unrecognized patterns

**Figure 48. GRABEX does not assume that all navigation elements have been correctly identified. Instead, it is assumed that specific block graph patterns reveal true navigation elements in a large set of candidates.**

## 6.1 GRABEX: Mining for Specific Block Graph Patterns

From the evaluation of the described prototype described in the previous section, we conclude that it is difficult to solve navigation element mining, i.e. the joining of navigational blocks, based on the applied similarity measures. However, searching for alternative similarity measures is not promising, since a suitable measure must not produce any error for a large number of comparisons to deliver useful results for a site. It is an open question, whether such similarity measure can be found at all. Hence, we rethink the basic CO-reverse-engineering process based on the assumption that block graphs allow to reveal the underlying navigation design pattern. For this, we address the CO-mining task in a different way and consider it as search for specific block graph patterns.

To explain the GRABEX-approach, the first three tasks (the tasks necessary to generate block graphs) can be thought of as filters (Figure 48). Task 1, page segmentation takes as input all DOM-elements and filters those DOM-elements that do not represent page blocks. Task 2, the identification of navigational blocks (block classification is considered as task 3a according to Figure 27) can be regarded as a filter that removes all non-navigational blocks. Task 3a, takes as input all pairs of navigational blocks and returns only those pairs that belong to the same navigation element (this is equivalent to grouping the navigational blocks to navigation elements). Finally, task 3b classifies navigation elements based on their block graphs into classes representing different underlying navigation design patterns. In contrast, an application that implements the GRABEX-approach is not based on such a general classifier but on a dedicated binary classifier that determines whether a block graph represents one specific design pattern or not. The GRABEX-approach is based on the idea of removing all prior filters and feeding the classifier with all navigation element candidates (i.e. all possible DOM element combinations) instead of requiring that the true navigation elements are identified a priori. The problem of identifying navigation elements is transferred to the classifier: We do not use a classifier that classifies navigation elements according to their types but a binary classifier that distinguishes 1) a specific type of navigation elements from 2) other types of navigation elements *and* random block combinations. The analysis of block graph patterns in Section 5.1 indicates that it is possible to implement such a classifier based on the block graph model and the solutions presented in this chapter provide the proof of concept.

Instead of trying to identify all navigation elements and then determining their types, the GRABEX-approach postulates to search for combinations of page blocks that result in characteristic block graph footprints (cf. Section 5.2.2). Each GRABEX-application for a specific navigation design pattern consists of the following components:

a) **A block graph representation that reveals characteristic patterns**

There are various ways of generating graph representations for the block graph model. As demonstrated in Section 5.2.2, different graph representations are suitable for different navigation design patterns.

b) **A classification method for the characteristic patterns**

In Section 5.2.2, we have only discussed the visualizations of block graph patterns but not how specific patterns can be automatically distinguished from other patterns or random structures.

c) **A method for reducing the navigation element candidate set**

Because of combinatorial explosion, analyzing all possible combinations of DOM-elements would not scale and hence, to pick up the filter metaphor, the filters are not entirely removed but relaxed. For instance, instead of trying to remove all non-navigational blocks in task 2, we only try to reduce the set by removing candidates that are very unlikely to represent navigational blocks.

## 6.2 Mining Hierarchical Menus with GRABEX

In this Section, we present an application of the GRABEX-framework, the MenuMiner-method. Parts of the contributions have been published previously in [KENU12A]. Following the GRABEX-framework, MenuMiner can be described by three components:

a) **A block graph presentation that reveals characteristic patterns (Section 6.2.1)**
MenuMiner extracts navigation elements that follow the main menu, local menu or multilevel menu pattern (cf. Section 5.2.2). The partial Web graph presentation (as introduced in Section 5.1) is analyzed in order to identify characteristic patterns. Characteristic patterns are complete subgraphs, so-called cliques .

b) **A classification method for the characteristic patterns (Section 6.2.2)**
Classification is conducted rule-based by searching for navigation element candidates whose partial Web graph presentations form maximal cliques.

c) **A method for reducing the navigation element candidate set (Section 6.2.3)**
Dedicated rule-based extraction algorithms are presented that reduce the navigation element candidate search space with greedy strategies. In addition, the DOM-tree is preprocessed to reduce the number of page block candidates.

In an empirical evaluation, we demonstrate that the MenuMiner-method is able to solve open Web mining problems (Section 6.2.4). It is also shown that the hierarchical CO can be reverse-engineering based on MenuMiner-data (Section 6.3).

### 6.2.1 Characteristic Patterns: Cliques

The partial Web graphs $W_j$ of several navigation design patterns analyzed in Section 5.2.2 contained cliques, i.e. complete subgraphs in which each pair of nodes is connected by an edge, with more than three nodes in $SR_j \cap TG_j$. Cliques were found to be characteristic for the main menu pattern, the local menu pattern, the multilevel menu pattern and the language switch pattern. If a Web site is organized based on a hierarchical CO, it typically contains either main- and submenus or a multilevel menu. Hence, mining for navigation elements that result in cliques should deliver all navigation elements that are necessary to reverse-engineer the hierarchical CO in subsequent processing steps (however, it must be considered that not all delivered navigation elements necessarily represent parts of the hierarchy). We can also argue in a different way that the navigation elements that define cliques in the Web graph usually represent the backbone of a site's navigation: These navigation elements allow users to navigate over a number of pages while the menu remains fixed. After a user has moved from one page to another, the menu is still present and contains the same links as before (cf. Figure 49). Hence, such menus define fixed landmarks from which users learn the CO of a site.

In Section 5.2.2, we observed a single clique in the partial Web graph of the main menu pattern and multiple cliques in the graphs of the local and multilevel patterns. We search for combinations of block candidates that result in single cliques only and do not consider the multi-clique case in the classification-rules. Hence, a single navigation element may be split up into multiple groups of navigational blocks. However, we can easily consolidate these groups by post-processing (cf. Section 6.2.3.3).

## 6.2.2 Classification Method

The large graph visualized in the background of Figure 49 is the complete Web graph of the site from which the four sample pages originate. The depicted clique of four is a subgraph of the Web graph. However, the three pages “Discover”, “How-to” and “Apps” alone, for instance, form a clique as well. But we can assume that navigation element candidates that form larger cliques are more likely to represent navigation elements than candidates resulting in smaller ones. Hence, we only need to search for cliques that are not part of larger ones, so called maximal cliques. In a single graph, there can be multiple maximal cliques, which may or may not overlap. For example, the page “Discover” could form a clique with some other nodes than the depicted ones, too. Although enumerating all maximal cliques of an undirected graph is known to be a NP-complete problem in the general case, it can be solved in O($\Delta^4$) time, if the maximum node degree $\Delta$ is fixed [MAUN04]. Since the Web graph can easily be transformed in an undirected representation by removing all edges that are not bidirectional and we can assume a fixed upper limit of links that can be placed on a page, it would be possible to enumerate the cliques in this way.

However, the problem to solve is more complex, since maximal cliques in the Web graph do not necessarily indicate navigation elements that form maximal cliques. The reason is that the edges in a Web graph clique may result from multiple different navigation elements. Hence, we have to search for maximal cliques under the constraint that only a single page block from each page is involved in defining the clique. In other words, we have to search for maximal cliques among all possible combinations of pages blocks (each from a different page). To not confuse this kind of cliques with cliques in the Web graph, we will

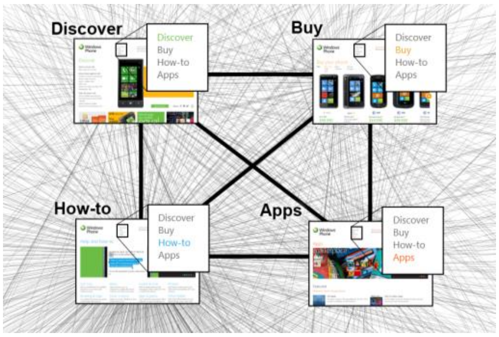


**Figure 49. Fixed menus define cliques [KENU12A]**

refer to them as *block cliques*. There is another constraint that we have to consider: Since each clique represents a menu and each link can only be part of a single menu, we should not attribute a single link to multiple cliques. However, a page block can indeed be part of multiple cliques, since in case of a multilevel menu, each menu level forms a separate clique (cf. Figure 34, page 81). Hence, an appropriate block clique extraction process is to iteratively search for the largest block graph clique in the candidate set and modify afterwards the candidate set by removing all clique links from the blocks that are involve in the clique (while maintaining the blocks themselves). If multiple cliques of the same size are found in an iteration, additional criteria can be considered in order to prioritize the cliques.

To formalize the block clique extraction process, we define the set $SC$ that contains all block cliques of a site (including non-maximal cliques). $SC$ contains all combinations of at least three blocks under the constraints that 1) each block belongs to a different page and 2) each block links the source pages of all other blocks.

To formally specify the set $SC$, we use the notation introduced in Section 5.1 and denote the *l*-th page block of page *i* as $B_{i,l}$. Let $BA$ be the set of all page blocks. $SC$ can be expressed as:

$$SC = \{C \in \mathcal{P}(BA) | \forall B_{i,l}, B_{j,k} \in C: (B_{i,l} \neq B_{j,k} \Rightarrow i \neq j) \wedge P_i \in B_{j,k} \wedge |C| > 2\} \tag{18}$$

The above definition expresses that if $C$ belongs to $SC$, and we take two different blocks from $C$, then both originate from different pages. Furthermore, the page, to which the first block belongs is linked in the second block (clique property). In addition, $C$ must have at least three elements.

MenuMiner extracts a set of block graph cliques $SC^* \subseteq SC$ by selecting the largest block graph cliques in a greedy way and taking the above constraints into account. The extraction rule can be formulated as an algorithm if *r* denotes a scoring function for prioritizing cliques of the same size:

1) Find $C_i \in SC$ that fulfills the following condition:
   $\forall C_j \in SC: |C_i| \geq |C_j| \wedge \left(|C_i| = |C_j| \Rightarrow r(C_i) \geq r(C_j)\right)$
   If no element of $SC$ fulfills the above condition: terminate and return $SC^*$

2) Add $C_i$ to $SC^*$

3) Update $SC$ by removing $C_i$ and all subsets of $C_i$:
   $SC \coloneqq SC \setminus \mathcal{P}(C_i)$

4) Let $P_{C_i}$ be the set of pages that form the clique $C_i$. Remove all block cliques from $SC$ that share an edge, i.e. hyperlink with $C_i$:
   $SC \coloneqq SC \setminus \left\{C_j \in SC: \left|P_{C_i} \cap P_{C_j}\right| \geq 2 \wedge |C_i \cap C_j| \geq 1\right\}$

5) If $|SC| > 0$ go to step 1, else stop and return $SC^*$

In step 1, the largest block clique is selected. If there are multiple cliques of the same size, the one with the highest value of the scoring function r is chosen. Step 3 ensures that smaller cliques embedded within $C_i$ are removed from the candidate set as well. Step 4 guarantees the condition that a hyperlink contributes only to a single clique. If $|C_i \cap C_j| \geq 1$, then both block cliques share a page block and at least one page that is part of the clique (the page to which the shared page block belongs). If $\left|P_{C_i} \cap P_{C_j}\right| \geq 2$, then at least one additional page is part of both cliques and, hence, the links to these additional shared pages in the shared page block contribute to both cliques, which would violate the constraint.

## 6.2.3 Efficient Computation

In this Section, we describe how the clique extraction rule can be implemented effectively. Cliques are processed with a local perspective (Section 6.2.3.1), a dedicated algorithm for reducing the search space of page block combinations is implemented (Section 6.2.3.2) and the set of page blocks is further decreased by applying preprocessing rules (Section 6.2.3.3).

### 6.2.3.1 Local Computation and Updating

If we would directly translate the extraction rule defined in Section 6.2.2 into an executable algorithm, we would have to operate on very large graphs. The first step of the extraction rule requires that we find the largest maximal block graph cliques. This could be done by generating a graph whose nodes are defined by all extracted page blocks and which contains an edge from page block *a* to page block *b* if block *a* contains a link to the source page of block *b*. Since we can assume an upper limit $\Delta$ of the maximal number of links in a block, all maximal cliques could be enumerated in $O(\Delta^4)$ [MAUN04] and the largest maximal cliques can be retrieved from the result set in linear time by sequential search. To avoid operating on this large graph structure, a more practical approach is to search for cliques locally, i.e. to process page after page and only search for cliques in which the processed page is involved. In addition, this approach has the advantage that MenuMiner can be applied while the crawling process is still running and the results can be fed back to direct the crawling strategy. For example, if not the entire domain should be crawled but only the most important pages, the extracted menus can be used to identify the upper levels of the hierarchy.

We also utilize the fact that block graph cliques are embedded in Web graph cliques. Hence, we mine for cliques in the Web graph, which is much smaller than the block graph, and then decompose the Web graph cliques to discover embedded block graph cliques. Figure 50 illustrates this approach. When we process page *P1*, we first expand the local Web graph, whose set of nodes consists of *P1* and all pages linked by *P1*. All links between these pages are considered as edges. We generate an undirected graph by removing all edges that do not have a counterpart in the reverse direction. Then we use an implementation of the Bron-Kerbosch-algorithm [BRKE73] to efficiently enumerate all maximal cliques. In the example depicted in Figure 50, the three maximal Web graph cliques *WC1*, *WC2* and *WC3* are returned. Now all cliques are decomposed to reveal embedded block graph cliques. This is achieved by iteratively applying the BlockCliqueFinder-algorithm described in the next section. The BlockCliqueFinder-algorithm delivers the largest block graph clique embedded within a Web graph clique. If there are multiple possible solutions, the scoring function *r* as motivated in Section 6.2.2 is taken into account. In our implementation, the similarity of the selector token paths (cf. Section 5.3.1.4) is used as scoring function *r*. After a block graph clique has been extracted, the involved links cannot be attributed to another clique anymore due to the constraint that one and the same link can only be part of a single menu. Hence, we have to update our model after each iteration of the BlockCliqueFinder-algorithm (cf. Figure 50). Figure 50 is a simplified illustration, since links are not actually deleted from the model. Instead, the links are marked as being already bound to a clique of a specific size. This is necessary to avoid that, for instance, the common edge of *WC1* and *WC2* is bound to a smaller block graph clique embedded in *WC2* and not to the block graph clique of size four embedded in *WC1*, if *WC2* is decomposed prior to *WC1*. Hence, BlockCliqueFinder operates always on the entire local Web graph and if a block graph clique is found that shares a link with another previously extracted block graph clique, the algorithm a) removes the link from

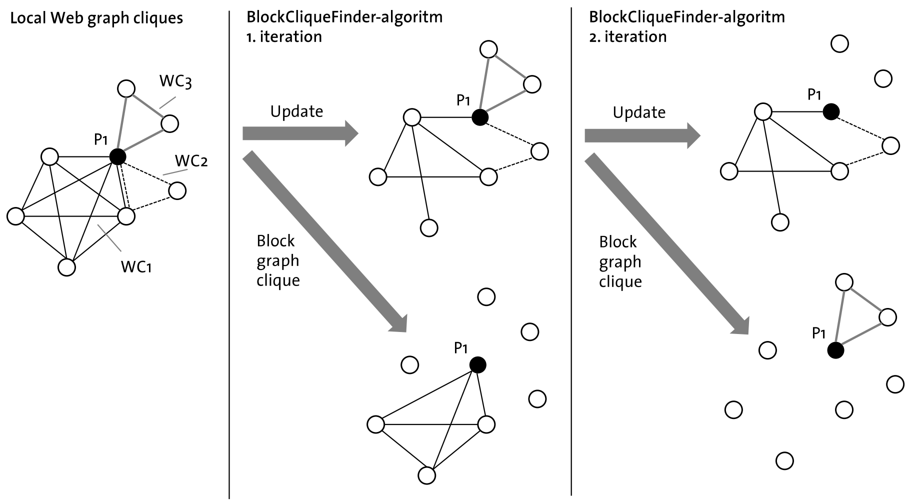


**Figure 50. Decomposing Web graph cliques**

the existing block graph clique if this clique is smaller or b) removes the link from the new block graph clique if the existing clique is larger. Note that to guarantee the correctness of the algorithm with respect to the extraction rule formulated in Section 6.2.2, we would have to propagate the changes because by reducing the size of the existing clique all of its links are now bound to a smaller clique than before and possibly should be attributed to another block graph clique. However, we found that these kinds of updates are seldom and tolerate a small chance of violating the extraction rule instead of allowing unmanageable complex cascades of updates.

### 6.2.3.2 Reduction of Search Space: BlockCliqueFinder-algorithm

In this section, the BlockCliqueFinder-algorithm, a fast method for decomposing Web graph cliques into block graph cliques is described. The BlockCliqueFinder-algorithm returns the largest block graph clique embedded within a Web graph clique by considering the scoring function *r*. To motivate the algorithm, we consider the exemplary clique of four shown in Figure 51. (Note that this is not a real-world example, since we are typically dealing with much more complex structures with respect to the number of involved pages, page blocks and links.) Each page in the example contains two page blocks. If there is an embedded block graph clique, there are three options for each of the pages P1, P2 and P3: 1) block 0 can be part of the clique, 2) block 1 can be part of the clique or 3) none of the blocks of the page is involved. There are only two options for page P0, since this is the page being currently processed and we are only interested in cliques adjacent to the current page. Hence, either block 0 or block 1 of page P0 is involved in an embedded block graph clique of interest.

All possible combinations can be thought of as leaf nodes of a search tree (Figure 52). The path from the root to the leaf describes the combination of blocks. For instance, solution S4 results from including block 0 of page P0, block 0 of page P1, block 1 of page P2 and block 0 of page P3. The numbers within the squared brackets of the solutions express which links appear in all blocks on the path. For instance, "[1, 0, 1, 0]" means that all involved blocks contain links to pages P0 and P2. Solutions with the maximum number of shared links (3 in the example) represent maximal block graph cliques (solutions S1-S4). We want to find the maximal block graph clique with the highest score, without having to expand the entire tree

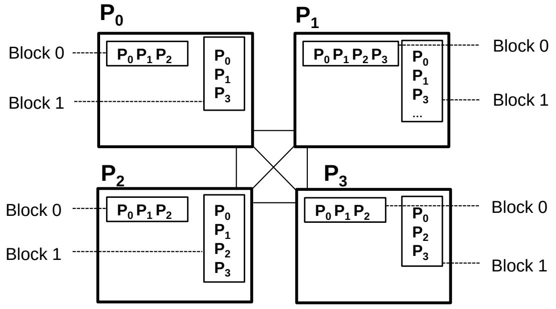


**Figure 51. A Web graph clique of four pages.**

in order to save runtime. To achieve this, we apply three strategies:

- **Pruning:** A transition from a parent node to a child node, i.e. considering the intersection of links with an additional block, will either not change the number of shared links or decrease it. Hence, if we have found a solution (leaf node) representing a clique of size k, we do not need to further expand branches that represent less than k shared links because we can be sure that they do not evaluate to larger maximal cliques. In our example, for instance, all branches with less than three shared links can be pruned once one of the solutions S1-S4 is discovered.
- **Greedy-Expansion:** In order to effectively reduce the search tree by pruning, solutions (i.e. leaf nodes) with many shared links should be discovered as quickly as possible. Hence, the tree is not expanded in a depth first or breadth first manner but in a greedy way. This is done by expanding the most promising branches, i.e. the branches with the largest number of shared links first.
- **Branch merging:** Branches located at the same level and representing the same set of shared links (for example A1 and A2 at processing level P2), are ancestors of the same set of solutions. In other words, if we have expanded the subtree A1, we can be sure that there will be no better solution below A2 with respect to the clique size. However, the solutions below A1 and below A2 represent different block combinations and the scoring function *r* can differ. We use a heuristic and apply the scoring function to the intermediate results A1 and A2. Then, only the more promising branch is expanded while the other one is pruned.

In addition to the tree expansion strategy, we have to check each time a new branch is generated, whether the shared links are not already bound to another, larger block graph clique. Links for which this applies are removed.

A pseudo-code formulation of the BlockCliqueFinder-algorithm, which was published in [KENU12A] can be found in Appendix A.

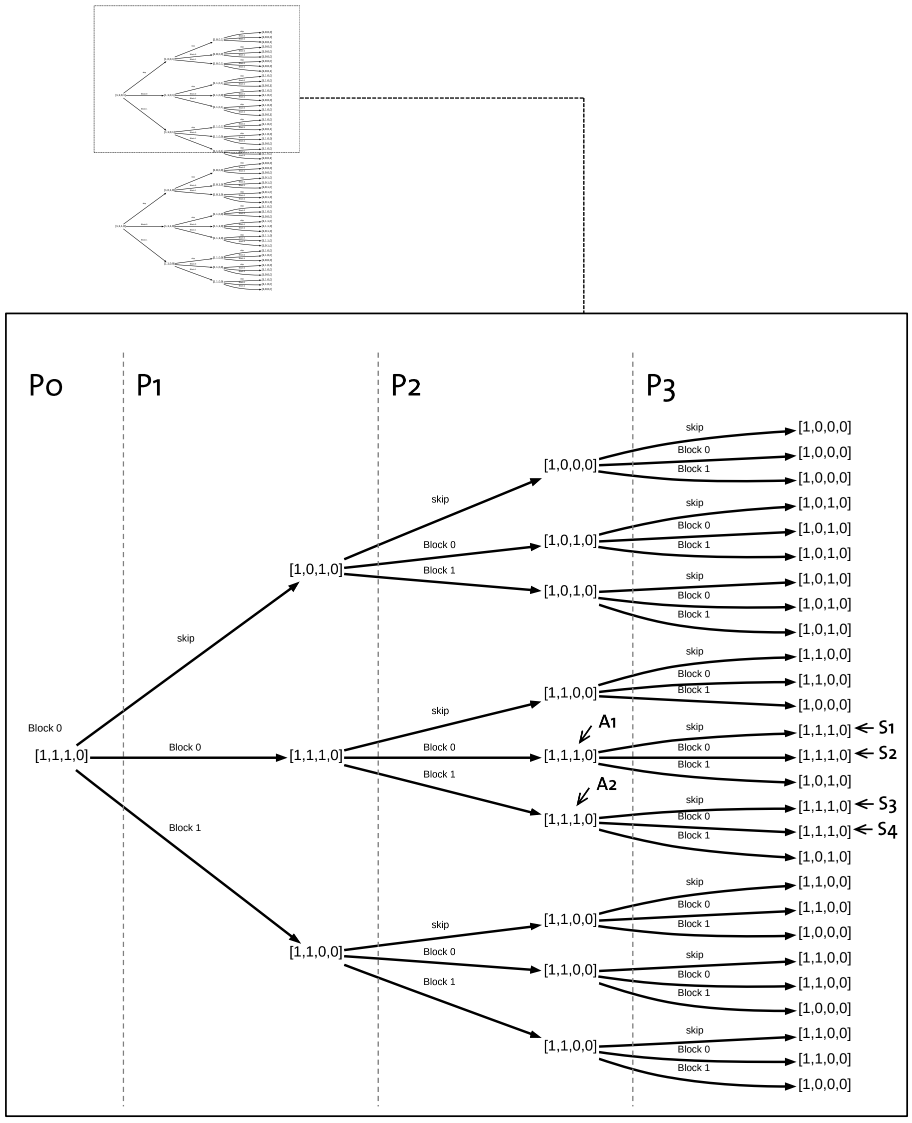


**Figure 52. The block graph clique search tree for the Web graph clique depicted in Figure 51.**

#### 6.2.3.3 Pre- and Postprocessing

In principle, every DOM-element of every page could be considered as page block candidate and the BlockCliqueFinder-algorithm could be run on this set. However, we can reduce the set of page block candidates considerably to save runtime by applying the DOM-tree transformation method presented in Section 5.3.1.2. This segmentation method is further improved by taking the particular use case into account and considering specific properties of the navigation elements of interest. The original transformation method was to process the DOM-tree bottom up, removing leafs that are no hyperlinks and internal nodes that have only a single child. Figure 53 illustrates the effect of these rules on a typical DOM-structure of a multilevel menu. In the example, four hyperlinks, each wrapped inside an individual *div*-element, represent the first level of the hierarchy. Below the third menu item, an expanded submenu is present. The submenu is implemented as sibling of the *a*-element and the second level links are wrapped inside a separate *div*-element. We found this kind of tree structure to be very common for implementing menus. However, if the original transformation rules are applied, the *div*-parents of the first, second and fourth *a*-element are removed because each has only a single child element, while the *div*-element wrapping the third *a*-element and the submenu remains. As a result, the logical structure of the menu in which the four links of the first level are all located at the same depth of the hierarchy is degraded after the tree transformation. To avoid this, we implemented an additional, menu-specific transformation rule that applies if an element has exactly two child elements and one is an *a*-element while the other is not. In this case, we treat the element similar to elements with a single child, by removing it and appending the children to the parent.

After the transformation of the DOM-tree, the internal nodes of the resulting tree represent the block candidates. Obviously, the block candidates are nested. Consider the example depicted in Figure 54: We do not know if the outer block containing Link 1 – Link 6 represents the boundaries of a menu or the inner block containing Link 3 – Link 5. We have to consider both candidates, but to reduce the size of the data structure and further improve runtime performance we remove the links of the inner block from the outer block candidate. If the

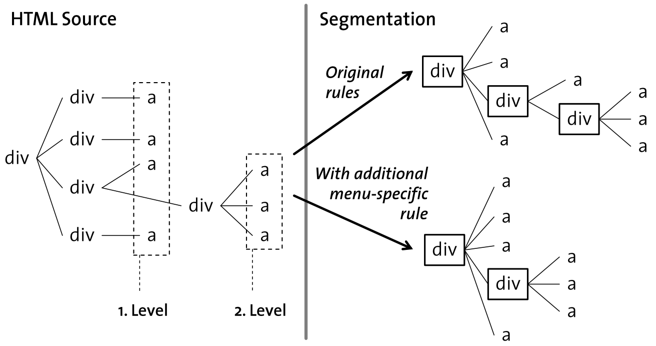


**Figure 53. Menu-specific rule to preserve logical structure ([KENU12A], revised).**

outer page block represents a navigational block, the inner block represents a submenu. As illustrated in Section 5.2.2, both levels define separate cliques and, hence, the clique defined by the outer block candidate remains intact which allows us to still recognize the menu. In a post-processing step, we can easily reassemble the page blocks by including all child blocks.

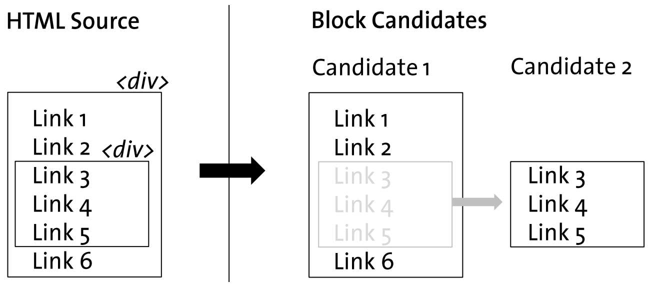


**Figure 54. Treatment of nested blocks**

### 6.2.4 Evaluation

To evaluate MenuMiner, a selected domain that contains a large number of diverse sites was crawled. The domain microsoft.com was chosen and we filtered for pages targeting the US-audience by testing for the substring "en-us" in the URL. We processed 10,000 pages with MenuMiner. A total number of 74,198 pages had to be downloaded because for each processed page, all its neighbors need to be retrieved as well[23].

#### 6.2.4.1 Runtime Performance

MenuMiner ran on a single Pentium D, 3 GHZ node with 3 GBs of RAM. The mean execution time of the Bron-Kerbosch algorithm was 0.15ms. For the BlockCliqueFinder-algorithm a mean execution time of 2.23ms was measured. Interestingly, there were few outliers that took up to 371ms to process while for 87% of the pages, the execution time was below 2ms. For each page, we also tracked the number of neighboring pages, the number of Web graph cliques, the number of block candidates and the number of returned block cliques. We found that the number of Web graph cliques correlates most strongly with the measured runtime (Figure 55). Although our sample contains few pages with more than 200 Web graph cliques, the plot indicates that MenuMiner is able to process more complex pages as well because the processing time seems to increase almost linearly with the number of maximal Web graph cliques. Since the processing of a page is not influenced by the number of previously processed pages, MenuMiner scales well in the number of processed pages.

[23] This overhead solely results from the fact that we were not processing the entire domain for practical reasons.

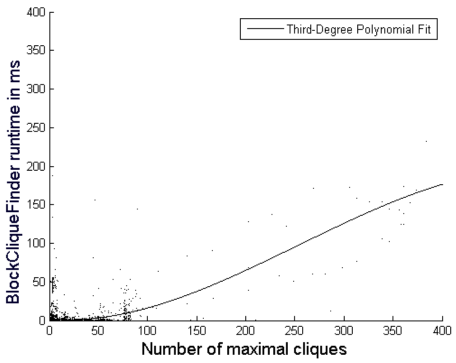


**Figure 55. The number of maximal Web graph cliques vs. BlockCliqueFinder runtime [KENU12A]**

#### 6.2.4.2 Evaluation of Correctness

MenuMiner extracts the fixed navigation elements of a site and, in particular, the navigation elements that represent the hierarchy levels. For reverse-engineering the CO, further processing of the results is necessary to reassemble the hierarchy. We focus on these aspects in Section 6.3, where we present and evaluate a navigation element reverse-engineering method based on MenuMiner.

The evaluation described in this section demonstrates that MenuMiner is a successful implementation of the GRABEX-approach because navigation elements can be extracted correctly and effectively by mining for specific graph patterns. In Section 5.3.3, we concluded that a main challenge that hindered the extraction of navigation elements is the problem of determining which navigational blocks belong to the same navigation elements and which do not. A single faulty association can corrupt the results of CO reverse-engineering entirely. Hence, the evaluation in this section focuses on that aspect and we apply the method for discovering the boundaries of Web sites. In doing so, we do not only evaluate whether navigational blocks are combined correctly into navigation elements but demonstrate at the same time that MenuMiner is a GRABEX-instance that allows to solve open Web mining problems.

**Web site boundary detection – problem statement**

Domains often host multiple different Web sites (sometimes called subsites). Humans can intuitively distinguish Web sites, since they differ in styles, navigation mechanisms and typically also the topical focus (cf. Section 2.1.1). In addition, the page headers indicate whether two pages belong to the same site or not. Recently there have been efforts to automatically mine Web site boundaries (e.g. [SENE05], [MEMF07],[ALCZ10], [ALCZ11]) because sites reflect the logical aggregation of Web pages better than domains. Web archiving solutions and ranking methods [SENE05] would benefit from the ability to determine site boundaries. Search result presentation could also be improved by organizing pages according to their source sites and not according to their source domains. However, none of the previously proposed solutions proved to be accurate enough to be applicable in practice.

**Our approach to Web site boundary detection**

In Section 2.1.1, a Web site was defined as a number of pages that have a shared CO, FO and PO. We argued that this reflects the intuitive and common understanding. For our evaluation, we utilize the fact that a common navigation element implies a shared CO and FO. It also implies a partially shared PO because both pages have at least the design and position of this navigation element in common. Hence, we can compute Web site boundaries based on the results delivered by MenuMiner and evaluate whether these boundaries reflect the Web site boundaries perceived by human users. MenuMiner outputs navigational blocks that form cliques. For instance, in case of a fixed menu with five hyperlinks, MenuMiner would return five navigational blocks, one from each linked page. However, the menu possibly appears on many more pages. To discover these occurrences, we use the selector token paths introduced in 5.3.1.4. First, the selector path tokens (element names, id-attributes and class attributes) that are shared by all navigational blocks of a navigation element are detected. The shared token path can be considered as template identifying navigational blocks belonging to this navigation element. Then, we search on the other pages of the site whether we can find additional blocks with the same selector token path pattern. We consider these blocks as occurrences of the navigation element as well. To compute site boundaries, we initially treat each page as a separate cluster, i.e. site. Then, all mined navigation elements are processed and the clusters of the source pages of all navigational blocks belonging to the same navigation element are merged successively.

**Evaluation metric**

To evaluate the quality of the computed site boundaries, we want to compare them to the boundaries perceived by humans. We can consider the computed site boundaries as clustering of the pages that we want to compare to the clustering perceived by humans. Metrics for comparing clusterings have been discussed and researched [WAWA07] but we are facing the problem that it is not practicable to manually cluster 10,000 pages. However, we can manually label a subset and estimate the Rand index [RAND71], a well-known measure for comparing clusterings. An advantage of the Rand index with respect to our scenario is the fact that it is based on pair-wise comparison. If there are two clusterings *C1* and *C2* for a set *O*, we can assign each possible pair of elements in *O* to the set of agreements *A* or the set of disagreements *D*. The set of agreements *A* contains all pairs on which both clusterings agree that either both elements belong to different clusters or both elements belong to the same cluster. The set of disagreements *D* contains pairs whose elements are assigned to the same cluster in one clustering but are placed in different clusters in the other clustering. The Rand index is defined as the fraction of agreements of *C1* and *C2*:

$$R_{C1,C2} = \frac{A}{A+D} \tag{19}$$

If we draw a random sample $S \subseteq O^2$ that is not too small, we can estimate the Rand index for both clusterings by computing the fraction of agreements based on the sample. Hence, if one of the clustering represents the site boundaries perceived by humans, we only need to determine for the sample pairs whether humans would attribute both pages to two different sites or not and not for all possible pairs. We labeled 845 pairs of pages in this way. In all cases the decision was obvious and undoubted.

**Results**

Based on MenuMiner, we computed 58 clusters and measured an almost perfect estimated Rand index of 0.996. A small error was expected since the crawling process was aborted after processing 10,000 pages and it is likely that some pages were included in the evaluation that originate from sites whose menus were not extracted since the clique members were not reached.

We implemented three different baseline methods

- Baseline method 1: To test whether the site boundaries could have been derived from the structure of the URL, we generated clusterings of three different granularities from the URL structure. First, we considered only subdomains as different sites, then, the first level of the folder hierarchy was included and finally, we also used the second level of the hierarchy to derive the clustering.
- Baseline methods 2 and 3: We implemented the methods for Web site boundary detection that performed best in a previous study [ALCZ10].[24] Both methods are based on a bisecting k-means algorithm which is used to iteratively split clusters into two parts. In contrast to MenuMiner, this approach requires specifying the number of clusters, i.e. iterations, a priori (a severe limitation with respect to the practical applicability). To evaluate if these methods could theoretically extract site boundaries correctly if the right number of clusters is specified, we varied the numbers of iterative splits. Baseline methods 2 and 3 differ only in the used features. Method 2 uses the set of internal hyperlinks that are present on a page as feature and method 3 uses words gained from tokenizing URLs ("/" and "." are considered as delimiters).

Figure 56 summarizes the results. There is no configuration of a baseline method that reaches an estimated Rand index as high as MenuMiner. The most accurate alternative method is the URL-hierarchy baseline in the configuration that solely considers subdomains. It achieves a Rand index of 0.97 but generates only 9 clusters, which is obviously too few. Hence, we conclude that MenuMiner is an applicable solution to the problem of Web site boundary detection that allows detecting site boundaries more accurately than previous solutions.

[24] Another method evaluated in the study relied on a combination of eight different features. However, it did not perform significantly better in the study and thus, we did not include it in our evaluation.

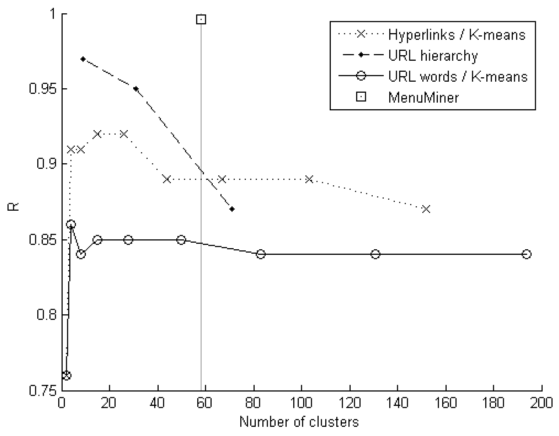


**Figure 56. Rand index R based on manually labeled samples for different boundary detection methods [KENU12A]**

## 6.3 Reverse Engineering Hierarchical Menus

MenuMiner delivers navigation elements but does not determine which parts of the hierarchy they represent. In this Section, we describe a rule-based approach to reverse engineer navigation elements delivered by MenuMiner in order to retrieve the hierarchical CO. In addition to the requirement of correctness in the sense that the CO as perceived by humans is returned, we developed this rule-based approach under consideration of the following requirements:

– MenuMiner is tag-agnostic and not bound to the semantics of specific HTML-tags. Thus, it neither depends on the tags included in current or future Web standards nor on the current practice of their usage. Downstream navigation element analysis should be sustainable in the same way, and, hence, a tag-agnostic solution is favorable.

– MenuMiner has an excellent runtime performance which allows for processing data at Web-scale. It does not require HTML-rendering, downloading additional resources such as images or CSS files and the execution of Javascript. Navigation element reverse engineering should not induce additional overhead by requiring any of these things.

– MenuMiner results can include navigation elements that do not follow the main-, sub- or multilevel-menu patterns because other navigation design patterns can also be characterized by cliques. An appropriate solution for navigation element reverse engineering should be fault tolerant with respect to these additional navigation elements.

## 6.3.1 Decomposing the Problem

We found that the problem can be decomposed into three subproblems (Figure 57), which will be discussed in the following sections:

1) Analyzing the internal hierarchy of individual menus (intra-menu hierarchies, Section 6.3.1.1)

2) Assigning additional pages to menu items (page assignment, Section 6.3.1.2). This is done by exploiting information that is available to users from visually highlighted menu items. Items in hierarchical menus that are highlighted, i.e. marked as active, indicate that the active page is arranged below this item in the hierarchical CO, even if the active page does not appear in the menu.

3) Discovering menu-submenu-relationships (inter-menu hierarchies, Section 6.3.1.3).

### 6.3.1.1 Analyzing Multilevel-Menus

In this section, we introduce the sub-problem of analyzing intra-menu hierarchies and describe our solution to this problem.

**Problem Description**

In Section 5.2.2, we have introduced multilevel menus as a class of navigation design patterns that represent subtrees of hierarchical COs. The examples a) and b) of Figure 13 on page 25 illustrate that multilevel menus can implement different strategies of expanding and collapsing menu levels in dependence of the active page. Human users are usually able to identify the logical tree structure of a multilevel menu because of the visual design. Users are able to learn the CO of the entire site by browsing and successively expanding different sections of the hierarchy. However, the CO is not reflected in a similar way in the structure of

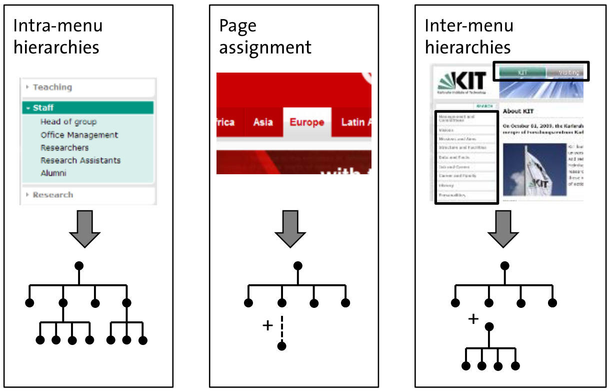


**Figure 57. The three subproblems of navigation element analysis: 1) Analysis of the internal hierarchy, 2) Identifying pages that are arranged under a menu item and 3) combing the hierarchy information of different menus**

the DOM-tree as motivated in Section 3.3.5 and exemplified in Figure 24 on page 45. From solely parsing the HTML-source, we cannot determine the logical structure of the menu items.

**Solution**

We distinguish two classes of multilevel menus: The first class subsumes menus whose state can be dynamically changed on the client-side via mouse actions ("pop-up menus"). The source code delivered by the server stays the same for all pages in case of this class. We will refer to such menus as *client menus*. The second class consists of multilevel menus for which the server returns a different state in dependence of the active page. Because the state is generated on the server side, we denote these menus as *server menus*.

*Processing of client menus.* The problem of crawling contents that are made available to users through Javascript manipulations of the DOM-tree is a challenging problem and a research field on its own (e.g., [MEBD08]). To our surprise, we found that retrieving the hierarchical CO from typical Javascript-based menus is not as challenging as expected. In most cases, the structure of the DOM-tree reflects the CO well. By applying a slightly modified version of the tree transformation method used by MenuMiner (Section 6.2), the original CO can be easily recovered. There are several explanations for the close match between the CO and the DOM-structure: First, the entire hierarchy is available on each page and not only specific subtrees. Second, it must be possible to address all levels of the hierarchy individually with Javascript to expand or collapse them. Hence, all levels are clearly separated by individual container elements. Third, usually Javascript frameworks are used to implement pop-up functionality. These frameworks are more mature than proprietary scripts and produce cleaner HTML-code with fewer presentation-related tags. Fourth, the structure of the hierarchy must be available to client-side scripts for allowing them to expand and collapse the correct levels. This hierarchy information is usually directly encoded in the DOM-structure and not gained from additional Javascript data structures that link parent and child elements. Hence, it is crucial for the functionality of a menu that the DOM-structure reflects the hierarchical CO. Using nested lists has emerged as a kind of standard for encoding the hierarchy of client-menus. To take account of this fact and to further improve the accuracy, we make an exception from the requirement of being tag-agnostic and include a specific rule for processing this kind of menus in our page transformation method: UL-elements are never discarded and LI-elements are always thrown away.

_Processing of Server Menus._ In case of server menus, the DOM-structure does not reliably reflect the structure of the CO. However, we can analyze which links are contained in the menu on each page and derive the original CO based on this information as we show in this section. We introduce the ListWalker-algorithm which achieves this by processing navigational blocks as sequences of hyperlinks. Due to space limitations and for readability, we focus on motivating the algorithm's functionality. A pseudo-code description can be found in [KEHA13A]. We assume that the state (by state, we mean the presented sequence of links) of a menu does not depend on the browsing history but solely on the active page. This reflects current best practices and exceptions are hard to find.

The ListWalker-algorithm covers solutions for three sub problems (Figure 58): 1) A way of identifying the root and the first level of the hierarchy, 2) a method for deriving the grandchildren if a parent-child relationship is known, and 3) a method for identifying cross-links. Solutions to the first two subproblems allow retrieving the entire hierarchy by starting from the first level and iteratively expanding the children. Solving the third subproblem is required to ensure applicability on real-world Web sites. We experienced that many hierarchical COs contain cross-links to pages located in other subtrees of the CO. Typically, pages that appear multiple times in the hierarchy have unique locations that can be identified by the state of the menu on the page itself. Hence, in hierarchy extraction, these pages should be placed at the location indicated by the menu state. Identifying this location is the third subproblem covered by the ListWalker-algorithm. In the following, we describe how the three subproblems are addressed:

1. *Identifying the root and the first level:* If the menu state on the root page of the hierarchical CO, i.e. on the homepage of the site, is known, we also know the first level of the hierarchy because we can assume that all deeper levels are collapsed. To identify this state, we proceed as follows: Let $l_1$ denote the first link in the link sequences, which is usually the same in all navigational blocks. We take the menu state on page $l_1$ (referred to as state $l_1$) and check whether another state exists, whose set of links is a subset of the links of state $l_1$. If such a state exists, we consider it as root state, if not we consider state $l_1$ as root state.

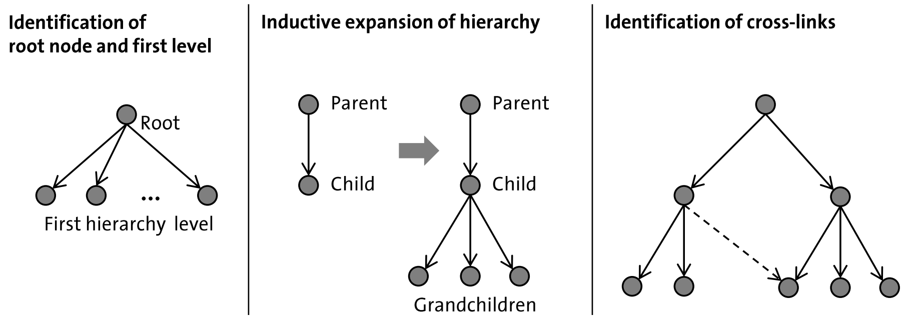

Identification of root node and first level
Root
...
First hierarchy level
Inductive expansion of hierarchy
Parent
Child
Parent
Child
Grandchildren
Identification of cross-links

**Figure 58. The three subproblems covered by the ListWalker-algorithm**

*Explanation:* Let $h$ denote the homepage and $a_1$, $a_2$,... $a_n$ the pages of the first level, i.e. the child nodes of the homepage. If $h$ is part of the menu, then we can assume that $l_1 = h$ because the homepage typically appears at the first position. In this case, state $l_1$ will have the form $(h, a_1, a_2, ..., a_n)$. Since all other levels except the first level are collapsed, we will not find another state whose links are a subset of this state and correctly identify it as root state. If $h$ is not part of the menu and $a_1$ does not have child nodes, state $l_1$ will have the form $(a_1, a_2, ..., a_n)$. This sequence equals the menu state on $h$ and we will correctly choose it because there is no other state with more collapsed levels. If $h$ is not part of the menu and $a_1$ does have child nodes, denoted as $b_1$, $b_2$,... ,$b_{m.}$, state $l_1$ will have the form $(a_1, b_1, b_2, b_3, a_2, ..., a_n)$. However, the root state on which the links $b_1$, $b_2$,... ,$b_m$ are collapsed is correctly identified because its links are a subset of the links in state $l_1$.

2. *Deriving grandchildren from parent-child relationships:* If page $b$ is a child node of page $a$, we assign all pages as child nodes to $b$ that are linked on page $b$ without being linked on page $a$. Hence, the rule can only be applied top down, starting from the root node and its children. The assumption behind this rule is that if a user navigates one level deeper in the hierarchy, only the immediate child nodes of the visited page appear additionally in the menu. Although multilevel menus can be implemented in many different ways, it is hard to find examples that violate this rule. The differences in the behavior of multilevel menus seem to lie only in the way intermediate levels collapse when descending the hierarchy (cf. Figure 59, also examples in Figure 13 on page 25 and Figure 24 on page 45).

3. *Identification of cross-links:* Another general rule that is hardly ever violated by multilevel menus is that the ancestors of the active page are always visible, even if intermediate levels collapse when descending the hierarchy. For instance in Figure 59-b, the sibling nodes of *Item 2.1*, which are visible on the parent page are hidden on the page rep-

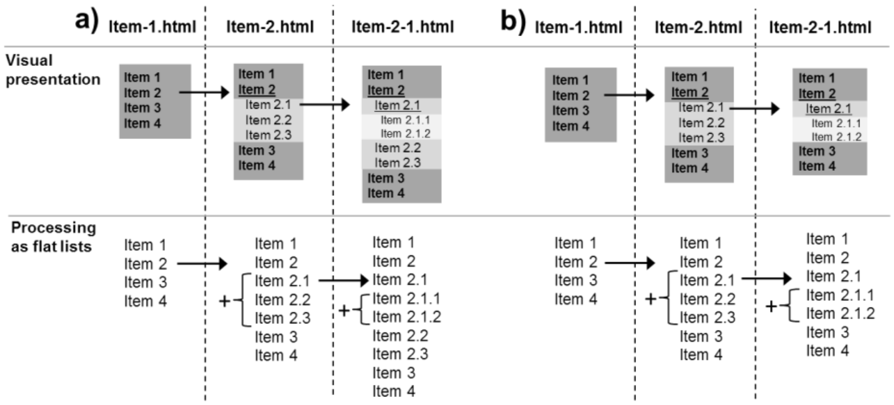


**Figure 59. The ListWalker-algorithm successively expands child items of menus. This allows recovering the original CO by processing menus as flat lists without considering the HTML-structure. [KEHA13A]**

resenting *Item 2.1* itself. However, even if the sibling nodes of the active page or its ancestors are hidden, the path from the root node of the CO to the active page is typically included in the menu. We hypothesize that this is caused by two reasons: First, users should be able to navigate back to higher levels of the hierarchy in an easy way and, second, the ancestors of the active page indicate the location of the page in the CO to the users. This allows us to test for crosslinks: If we identify a new child node by the rule described above, we check whether the path of the node to which we would append it is consistent with the menu state on the potential child. For this, we check whether all links of the path appear in the original order previous to the menu item representing the potential child itself. If this does not apply, we can assume a crosslink to another section of the CO hierarchy instead of a true child node.

#### 6.3.1.2 Page Assignment

In this section, we describe how additional pages can be arranged in the hierarchy, even if they are not linked in a menu.

**Problem Description**

Highlighted menu items indicate to users the location of a page in the hierarchy to users. Figure 60 shows two pages in which the menu item "Asia" is highlighted. Users learn from this visually encoded information that both pages are located below the node "Asia" in the global hierarchy. Hence, hierarchy information can be gained from marked menu items. Often, marked menu items are even the only way to derive the position of a page in the CO besides content semantics. The left-hand page shown in Figure 60 is the page linked by the menu item "Asia". If we know, for instance, that the menu containing the link represents the main menu of the site, we can conclude that the linked page is located at the first-level of the hierarchy in the CO. However, the right-hand page can only be reached by a link in the content area and there is no menu that contains a link to this page. Still, it is obvious to human users that the second page is also part of the section "Asia" because of the highlighted menu item[25]. In order to automatically reverse engineer the entire CO, we need ways of automatically identifying the position of pages in the hierarchical CO, even if the pages are not directly linked in hierarchical menus.

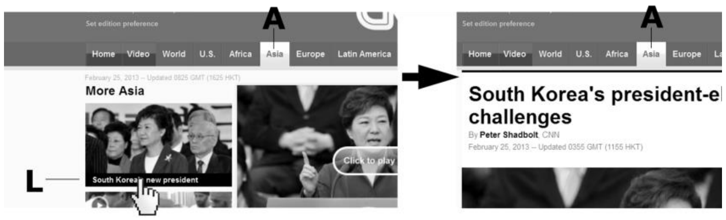


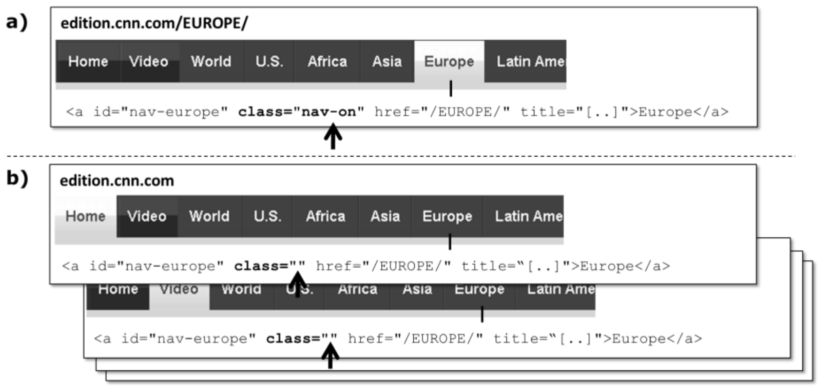
**Figure 60. The page on the right side is not linked by a menu item but by a link in the content area (L) only. Still, it is a child page of the active menu item "Asia" (A) [KEHA13A]**

[25] Even if the menu item "Asia" would not be highlighted, human users would probably attribute the right-hand page to the section "Asia" because of link context and semantic knowledge.

## Solution

Since the menu boundaries delivered by MenuMiner are known, we can extract features that allow determining the location of a page in the hierarchy. If we have two sample pages *a* and *b*, and we know that page *a* is part of a certain site section while page *b* is not, we can try to automatically extract distinguishing features for this section by comparing *a* to *b*. These distinguishing features can be used to identify other pages that are arranged under the same site section, i.e. CO subtree. In other words, we consider page *a* as positive sample for the specific site section and page *b* as negative sample. Finding positive samples is trivial because each page can be considered as positive sample for the site section rooted at the page itself. Knowing menu boundaries allows also identifying negative examples: If a page belongs to a menu, we can assume that all other pages in the menu represent sibling nodes and not child or successor nodes. Hence, they are part of other site sections. For deriving distinguishing features, we analyze (1) CSS-classes that are assigned to menu items and (2) the URL-structure of child nodes:

(1) The first method aims at determining visually highlighted menu items. Designers typically use specific CSS-classes for assigning the visual properties to highlight menu items. As Figure 61 illustrates, our method allows identifying these CSS-classes automatically. "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



```
<a id="nav-europe" class="nav-on" href="/EUROPE/" title="[..]">Europe</a>
```

```
<a id="nav-europe" class="" href="/EUROPE/" title="[..]">Europe</a>
```

```
<a id="nav-europe" class="" href="/EUROPE/" title="[..]">Europe</a>
```

**Figure 61. 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.[26] [KEHA13A]**

[26] CSS classes to highlight menu items can also be assigned to parent items of the a-elements, which must be considered in the implementation.

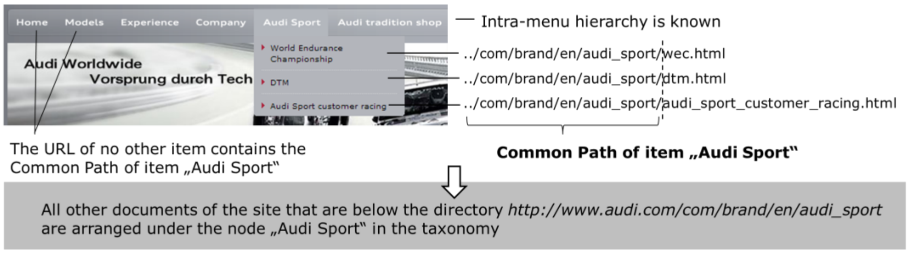

Home Models Experience Company Audi Sport Audi tradition shop
World Endurance Championship
DTM
Audi Sport customer racing
Audi Worldwide
Vorsprung durch Tech
Intra-menu hierarchy is known
../com/brand/en/audi_sport/wec.html
../com/brand/en/audi_sport/dtm.html
../com/brand/en/audi_sport/audi_sport_customer_racing.html
The URL of no other item contains the Common Path of item „Audi Sport"
Common Path of item „Audi Sport"
All other documents of the site that are below the directory http://www.audi.com/com/brand/en/audi_sport are arranged under the node „Audi Sport" in the taxonomy

**Figure 62. The child nodes have a common directory and in turn all other pages residing under this directory can be interpreted as child nodes of the same menu item [KEHA13A].**

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." [KEHA13A]

"(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." [KEHA13A] This approach is illustrated in Figure 62. "The URLs of the child nodes of the menu item 'Audi Sport' have a common directory prefix that is exclusive in the sense that no other pages linked by other menu items reside under this directory. All other pages of 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." [KEHA13A]

#### 6.3.1.3 Inter-Menu Hierarchy

In this section, we describe our findings on discovering menu-submenu-relationships.

**Problem Description**

A hierarchical CO can either be implemented as a single multi-level menu or as a combination of different main- and submenus. To cover the latter case, a CO reverse engineering solution must be able to detect which level of a hierarchy is represented by an individual menu.

**Solution**

We found that this subproblem is more difficult to address than the others. A submenu represents the child nodes of one of the pages in the main menu. Hence, there is only one page in the main menu on which the submenu is displayed. On the other hand, we can assume that all pages linked in the submenu also contain the main menu. If a pair of two navigation elements meets both criteria, we could regard them as main- and submenu. We applied this approach in [KENU11]. However, we found that this approach is not viable in practice because inconsistencies such as the same submenu being arranged under two different main menu items, appear to be too common. A possible solution would be to derive an appropriate set of graph-based features and train a classifier on a large data set as done in the BreadcrumbMiner-implementation (Section 6.4). The results of the other two sub-problems, i.e. intra-menu hierarchies and the set of pages arranged under a menu would be important features. Hence, we concentrate on those aspects in this chapter. Instead of a general solution to the problem of determining hierarchy levels, which is considered as future work, we focus on determining the main menu only. We experimented with different heuristics for identifying main menus and achieved good results with a simple scoring function $K_i$. The design of the scoring function is based on the consideration that a) more pages are arranged below the global main menu than below any local sub-menu and the average distance (cf. Figure 63) of the descendant nodes tends to be higher in case of a global menu and b) the order in which the menus appear in the HTML-source code reflects the importance in a way that higher menu levels tend to appear prior to lower level menus. For each menu, $K_i$ is computed as $K_i = A_i / P_i$, with $A_i$ being the average page distance and $P_i$ being the average menu position. To compute the average page distance $A_i$, the distance for pages that are arranged under the menu is specified as illustrated in Figure 63 and set to *0* for all other pages. $P_i$ is defined by the average position of the menu in the source code. For instance, if a menu is the first menu in the source code on all pages on which it appears, $P_i$ is 1. We select the menu with highest $K_i$ as the main menu if additional boundary conditions apply: a) the menu has less than 15 items, b) in average, not more than 30% of the textual content appear before the menu (this conditions ensures that footer menus are avoided) and c) there is no other menu with a lower $P_i$ value that appears on more pages. If a menu is identified in this way and pages of a domain do not contain this menu, we retrieve another main menu for these pages iteratively in the same way.

## 6.3.2 Implementation and Evaluation

The solutions to the three subproblems discussed above were implemented to process the menus extracted by MenuMiner. Navigational blocks of each navigation element were consolidated by joining similar menu states (in accordance with the block graph model, we consider the link sequence of a block as menu state). To discover similar menu states of a navigation element even if there is some noise (for instance, a single additional hyperlink appears on one page), the overlap coefficient is used as similarity measure [JOFU87]. First, intra-menu analysis is conducted by applying the methods for server-menu parsing and

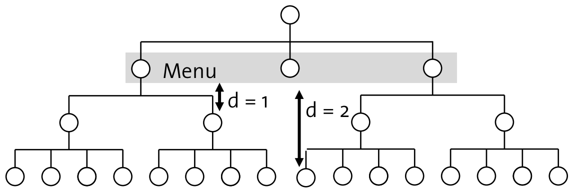


**Figure 63. Distance *d* of descendant nodes to a menu**

client-menu parsing for each menu. If both methods deliver hierarchy information, the result representing the largest tree structure is kept and the other one is thrown away. Second, page-assignment is conducted. Third, the heuristic for identifying main menus described in Section 6.3.1.3 is applied.

#### 6.3.2.1 Evaluation Methodology

MenuMiner scales well in the number of pages (cf. Section 6.2.4.1) and the number of delivered menus grows sublinear in the number of pages. At the same time, the rule-based solutions presented in this chapter consume only little resources. Hence, runtime performance is no major concern and we focus on evaluating the result quality.

To the best of our knowledge, a suitable labeled data set does not exist that could be used to evaluate the hierarchy extraction performance of our solution. Thus, we had to generate an own, manually labeled evaluation data set. For this, we crawled 350 randomly selected domains. We seeded the crawler with yahoo.com and implemented a depth-first search for new domains as crawling strategy. To spread the samples, we included only every 25th discovered domain in our analysis. After 350 domains were retrieved in this way, up to 100 pages were processed by MenuMiner for each domain. Since processing a page by MenuMiner requires downloading all neighbors as well, 259,525 pages were downloaded all in all. For each domain, one page was randomly selected and the main menu, the active main menu item, the secondary menu and the active secondary menu item were manually labelled according to the guidelines described in [KEHA13A]. In ambiguous cases, the sample pages were excluded from the data set[27].

The four evaluated subproblems, main menu detection, active main menu item detection, secondary menu detection and active secondary menu detection are evaluated as binary classification task. If a menu or active menu item is returned by our method, we consider it as positive sample, otherwise as negative. For a menu to count as true positive (TP), all links appearing in the mined menu must be contained in the labeled menu as well (the mined menu must not contain any false links). Otherwise, we count it as false positive (FP). On the other hand, we tolerate additional links in the labeled menu because it represents the local perspective (the hierarchy information contained on a single page) and supplementary, "noisy" links on a page are possible. However, 76.4% of the cases were a perfect match between the links in the mined menu and the labeled menu. Moreover, if it was not a perfect match, in average 74.6% of the hyperlinks of the labeled menu were contained in the mined menu. If no menu or active menu item was extracted for a page, we consider it a false negative (FN) if a menu was labeled for this page and a true negative (TN) otherwise. Precision and recall were calculated as described in Section 4.3.3 for all four subtasks. For evaluating the identification of active menu items, we included only samples for which the menu itself was correctly extracted. We evaluated four different configurations: In configurations A and B we exclusively applied the CSS selector-based method and

[27] The evaluation data set can be downloaded from http://dsn.tm.kit.edu/download/icwe2013/data.zip

the URL-based method, respectively, for page assignment. In configuration AB, we applied both methods. Configuration $A_{res}$ is a more restrictive version of A, achieving a higher precision at the expense of a reduced recall by returning only menus if a second level has been detected.

### 6.3.2.2 Evaluation Results

Configurations A, B and AB deliver the first level of the CO hierarchy 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.” [KEHA13A].

**Table 8. Evaluation results [KEHA13A]**

| | Config. **A** | Config. **B** | Config. **AB** | Con. $\mathbf{A}_{res}$ |
|---|---|---|---|---|
| **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 |

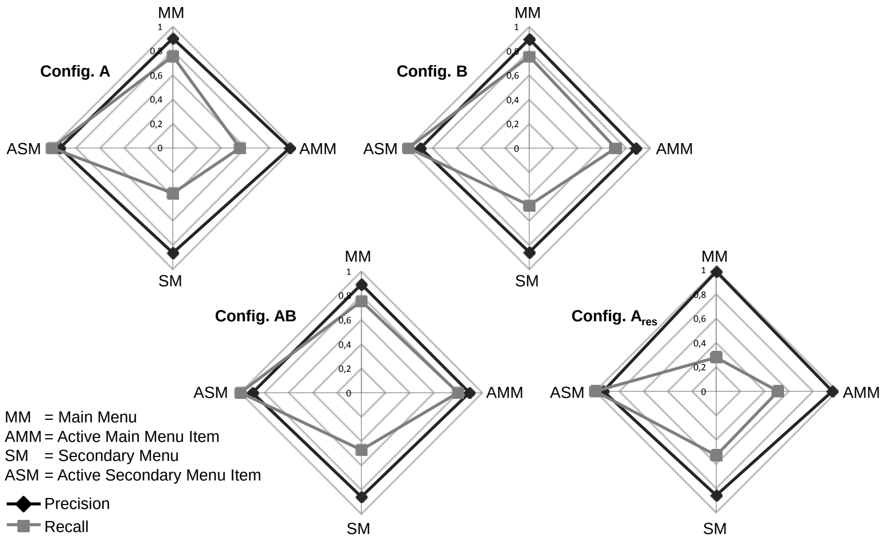


**Figure 64. Precision and recall of the evaluated configurations [KEHA13A].**

## 6.4 Mining Breadcrumbs with GRABEX

In this Section, we describe an application of the GRABEX-approach that allows mining breadcrumb navigations with high accuracy. The breadcrumb navigation design pattern was introduced in Section 2.3.2. Breadcrumb navigations indicate the position of a page in the CO of a site. The hyperlinks in a breadcrumb navigation describe the path from the root node to the active page. Hence, if the breadcrumb navigations from all pages of a site are extracted, the CO can be easily assembled. At the same time, the breadcrumb navigation design pattern is one of the most common patterns, which is present on many hierarchically organized sites. Yet, to the best of our knowledge, an applicable breadcrumb extraction method was not published previous to our GRABEX-based solution [KEHA13B]. However, the search engine Google is able to extract breadcrumbs to enrich the presentation of search results[28]. Although details of Google's solution are not published, we can use it as baseline method for performance evaluation. Since we consider the presentation of search results as one of the main use cases for hierarchical CO-extraction and argue that more hierarchy information should be integrated in the presentation of search results as currently done (cf. Chapter 7), our mining solutions is developed with focus on this use case. As a consequence, we consider the presentation of faulty hierarchy information as more harmful from the user

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

perspective than the absence of hierarchy information. Hence, we prioritize high precision over high recall (cf. Section 4.3.3).

Breadcrumb-Miner implements the four components of the GRABEX framework as follows:

a) **A block graph presentation that reveals characteristic patterns (Section 6.4.1)**
As observed in Section 5.2.2, the breadcrumb navigation design pattern is characterized by tree structures in the partial link graph. We show that small inconsistencies that commonly appear in practice can easily corrupt the tree structure if the graphs are generated this way. Hence, we define another graph representation in which breadcrumb navigations are characterized by tree structures as well but that is more robust against irregularities and random trees.

b) **A classification method for the characteristic patterns (Section 6.4.2)**
To handle inconsistencies (e.g., pages that appear multiple times in the hierarchy) and to avoid misclassification of accidental tree structures, we train a decision tree classifier on a manually labeled training set.

c) **A method for reducing the navigation element candidate set (Section 6.4.3)**
The principle that similar selector token paths indicate a similar visual presentation, which is motivated in Section 5.3.1.4 and also used in MenuMiner, is more systematically exploited. We assume that either all navigational blocks of a navigation element or their parents share at least a class- or id-attribute. Only page block combinations for which this applies are considered as navigation element candidates.

### 6.4.1 Breadcrumb Graph

In Section 5.1, we have introduced the partial link graph $L_j$ for a navigation element $j$, which contains an edge between two pages $a$ and $b$, if a link to $b$ succeeds a link to $a$ in one of the navigational blocks. If the underlying CO is a clean hierarchy without inconsistencies, and $j$ is a breadcrumb navigation, $L_j$ will be a tree. However, we found that exceptions are very common and have to be considered. In particular, individual pages often appear multiple times in the hierarchy despite the fact that they represent individual nodes in the CO. As Figure 65 illustrates, in these cases, the link graph representation does not reflect the hierarchical structure any more. The limitation can be avoided by choosing a more appropriate graph representation in which nodes do not represent individual pages but sequences of pages. We will refer to this graph representation as breadcrumb graph. The breadcrumb graph is generated by iteratively adding new page blocks and proceeding as follows: 1) If the node representing the sequence of links in the page block is not part of the graph, add a new node, 2) if the sequence of links gained by removing the last element is already part of the graph, add a new edge from the existing node to the new node, 3) if it is not part of the graph, add this node as well, connect both new nodes with an edge and go back to step 2. The resulting graph is depicted in Figure 66.

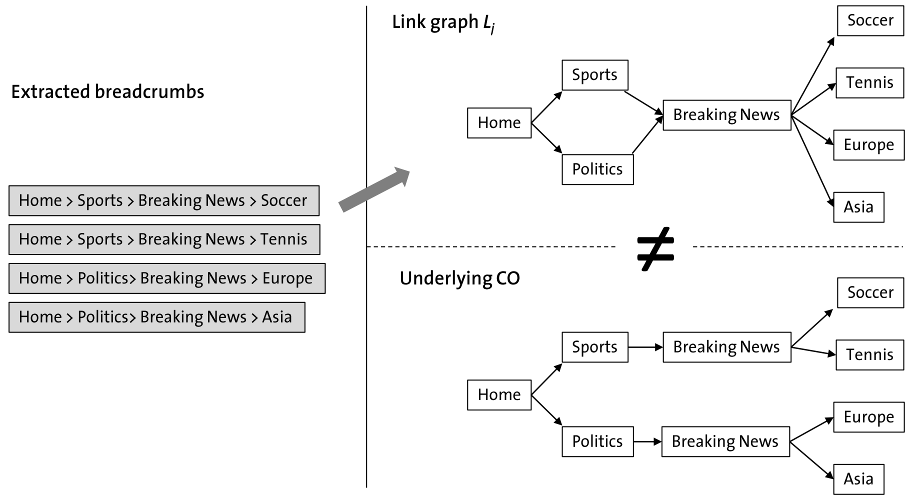


**Figure 65. If a page appears multiple times in the hierarchy as different logical nodes ("Breaking News" in the example), the link graph representation does not reproduce the tree structure any more.**

More formally, we define the breadcrumb graph $C_j$ as follows. According to the notation used in Section 5.1, the $l$-th block of page $i$ is modelled as a sequence of $m_{i,l}$ hyperlinks:

$$B_{i,l} = \left(u_{i,l}^1, u_{i,l}^2, \dots, u_{i,l}^{m_{i,l}}\right) \text{ with } u_{i,l}^1, u_{i,l}^2, \dots, u_{i,l}^{m_{i,l}} \in S \tag{20}$$

If $B_{i,l}^d = \left(u_{i,l}^1, u_{i,l}^2, \dots, u_{i,l}^d\right)$ denotes the subsequence of the first $d$ elements of $B_{i,l}$, with $d \in \mathbb{N}^+, d \leq m_{i,l}$, the set of nodes of the breadcrumb graph $C_j$ is defined by ($N_j$ denotes the set of navigational blocks associated with $j$) :

$$V_{C_j} = \bigcup_{B_{i,l} \in N_j} \left\{B_{i,l}^1, B_{i,l}^2, B_{i,l}^3 \dots, B_{i,l}^{m_{i,l}}\right\} \tag{21}$$

$V_{C_j}$ is the set of link sequences of all page blocks belonging to a navigation element and their subsequences. Each two consecutive prefixes define an edge:

$$E_{C_j} = \bigcup_{B_{i,l} \in N_j} \left\{\left(B_{i,l}^1, B_{i,l}^2\right), \left(B_{i,l}^2, B_{i,l}^3\right), \left(B_{i,l}^3, B_{i,l}^4\right), \dots, \left(B_{i,l}^{m_{i,l}-1}, B_{i,l}^{m_{i,l}}\right)\right\} \tag{22}$$

## 6.4.2 Classification

Simply testing whether the breadcrumb graph is a tree is not sufficient for reliably distinguishing breadcrumb navigations from other navigation design patterns, since tree structures might appear accidentally, in particular small trees. Thus, factors such as tree depth, tree breadth or the number of involved pages must be taken into account. The thresholds to use and the interactions between different factors are not obvious and, hence,

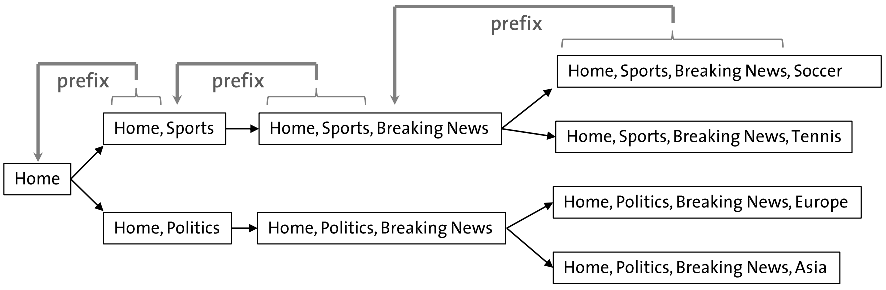

prefix
prefix
prefix
Home
Home, Sports
Home, Sports, Breaking News
Home, Sports, Breaking News, Soccer
Home, Sports, Breaking News, Tennis
Home, Politics
Home, Politics, Breaking News
Home, Politics, Breaking News, Europe
Home, Politics, Breaking News, Asia

**Figure 66. In the breadcrumb graph, the nodes do not represent pages but sequences of pages. Removing the last element of the sequence delivers the parent node.**

a decision tree classifier is trained. We extracted nine different features describing the structure of the breadcrumb graph that we considered to be relevant for the classification:

$F_{NumLinks}$ – the average number of hyperlinks in each block sequence is an obvious structural characteristic. Formally, this feature is defined as:

$$F_{NumLinks} = \frac{\sum_{B_{i,l} \in N_j} m_{i,l}}{|N_j|} \qquad (23)$$

$F_{NumLinksInteger}$ – we assume that some navigation elements will always contain the same number of hyperlinks on each page, while this does not apply in case of the breadcrumb navigation design pattern. To make this characteristic available to the classifier, we compute the average number of hyperlinks, which is an integer if the number of links does not vary. In case of an integer value, $F_{NumLinksInteger}$ is set to 1, otherwise to 0.

$F_{LogicalDepth}$ – we found that the number of hyperlinks does not always reflect the depth of the CO in the same way because different breadcrumb implementations exist. First, the root node may or may not be included in the breadcrumb as a hyperlink (in Figure 66, the root node “Home” is included). Second, the active page may or may not be included in the breadcrumb. In order to reflect the true depth of the CO and to abstract from the different implementations, we added the feature $F_{LogicalDepth}$. The feature also measures the average number of hyperlinks (like $F_{NumLinks}$), but we increased the link count 1) for all page blocks if there are multiple 1-tuple nodes in the breadcrumb tree, since we assume a breadcrumb implementation without homepage link and 2) for each page block whose link sequence appears on multiple pages since we assume that the active page is not part of the bread-crumb. Formally, if we define *r(x)* as

$$r(x) = \begin{cases} 1 & if\ x > 1 \\ 0 & otherwise \end{cases} \qquad (24)$$

and the function $sb_j(B_{i,l})$ that returns all pages blocks of a navigation element that consist of the same link sequence as $B_{i,l}$, $F_{LogicalDepth}$ is given by:

$$F_{LogicalDepth} = r\left(\left|\left\{x \in V_{c_j} \middle| \exists i,l: B_{i,l} \in N_j \wedge B_{i,l}^1 = x\right\}\right|\right) + \frac{\sum_{B_{i,l} \in N_j}\left(m_{i,l} + r\left(\left|sb_j(B_{i,l})\right|\right)\right)}{|N_j|} \qquad (25)$$

**F$_{\text{DepthDiff}}$** – if links to the root node and the active page are part of the breadcrumb, F$_{\text{LogicalDepth}}$ is possibly a more reliable indicator compared to adjusted F$_{\text{LogicalDepth}}$ values. Hence, we included the feature F$_{\text{DepthDiff}}$, which expresses the extent of the correction:

$$F_{DepthDiff} = F_{LogicalDepth} - F_{NumLinks} \tag{26}$$

**F$_{\text{NumBranches}}$** – another obvious structural characteristic of the breadcrumb graph that is available as classification feature is the number of branches:

$$F_{NumBranches} = \sum_{v \in V_{C_j}} r\left(\left|\left\{x \in V_{C_j} \middle| (v,x) \in E_{C_j}\right\}\right|\right) \tag{27}$$

**F$_{\text{Predecessors}}$** – The used graph representation avoids that the tree structure is violated even if pages appear multiple times in the hierarchy (cf. Figure 65 and Figure 66). However, the number of times such inconsistencies appear can be an important classification feature. Hence, F$_{\text{Predecessors}}$ counts the average number of times each page appears in the hierarchy, i.e. the average number of predecessors. With $B_{i,l} = \left(u_{i,l}^1, u_{i,l}^2, \dots, u_{i,l}^{m_{i,l}}\right)$ being the sequence of hyperlinks in block $B_{i,l}$, we define the function $pr_j(u)$ that returns the set of all predecessors of a page $u$ from Site $S$ in the navigational blocks of navigation element *j*:

$$pr_j(u) = \{x \in S | i,l,k \in \mathbb{N} : B_{i,l} \in N_j \wedge u_{i,l}^k = x \wedge u_{i,l}^{k+1} = u\} \tag{28}$$

Then, we calculate F$_{\text{Predecessors}}$ as:

$$F_{Predecessors} = \frac{\sum_{u \in S} pr(u)}{\sum_{u \in S} r(pr(u))} \tag{29}$$

**F$_{\text{LabelRatio}}$** – The previous features are solely URL-based and do not include information about the link labels, i.e. the text embedded in an *a*-element. But we can assume that in case of the breadcrumb navigation design pattern, page names are used as link labels and hence, one and the same URL always has the same label. This does not apply in case of all navigation design pattern: For example, in navigation elements based on the pagination design pattern (cf. Section 5.2.2), a URL can be labeled with “previous” on one page and with “next” on another. Hence, in case of some navigation design patterns other than breadcrumbs, the number of different labels (e.g. “previous” and “next”) is small in comparison to the number of different URLs that appear and this characteristic can be used as additional classification feature. Thus, the ratio of the number of different link labels to the number of different hyperlinks is included as another feature.

**F$_{\text{NumId}}$** – This is an additional feature resulting from the applied method for reducing the set of navigation element candidates (cf. Section 6.4.3). The feature expresses how often the class- or id-attribute that is used in the style identifier of the navigation element candidate appears in average on a page. If the style identifier really represents the unique formatting rules of a navigation element, it should appear only a single time on each page. To allow for some fuzziness, we used the average number of occurrences on a page to express this characteristic instead of a Boolean value.

**$F_{Count}$** – The number of navigational blocks that are associated with the navigation element candidate is another obvious feature that is considered.

## 6.4.3 Reduction of Navigation Element Candidate Set

In this section, we describe how the set of navigation element candidates is reduced in order to allow efficient computation.

### 6.4.3.1 Page Segmentation and Initial Candidate Set

In the context of MenuMiner, we introduced the concept of selector token paths to estimate the visual similarity of two navigational blocks. This was motivated by the assumption that rational designers reuse CSS-rules and selector tokens are used to attach these rules. Since we experienced that this assumption holds for almost all sites, we exploit CSS-selectors more systematically for BreacrumbMiner: We assume that if Web designers implement breadcrumbs, they also implement dedicated CSS-rules that apply only to the breadcrumb navigation and not to other elements of the page. Hence, they need a way to address the breadcrumb with dedicated CSS-selectors. Furthermore, we assume that this is done in one of the following ways:

1. An exclusive id- or class-attribute is assigned to all navigational blocks of the navigation element. This attribute is referred to for defining the CSS-rules for the breadcrumb.
2. An exclusive id- or class-attribute is assigned to the parent element of all navigational blocks. Contextual CSS-selectors such as descendant or child selectors (cf. [KENU10]) are used to address the navigational blocks of the breadcrumb via the parent element.

In theory, CSS-rules that exclusively apply to the navigational blocks of the navigation element can be defined as well if neither the navigational blocks nor their parent elements have an exclusive id- or class-attribute. However, this requires more complex CSS-selectors, more cognitive effort and results in CSS-code that is difficult to understand and to maintain. Hence, CSS-code is written rarely in this way and the above cases cover the vast majority of implementations. With these assumptions, not all combinations of page blocks have to be considered as navigation element candidates any more: Each id- or class-attribute that appears on the site corresponds to a set of page blocks, which is the set containing all the HTML-elements to which this attribute is assigned. These sets have to be considered to cover case 1. To cover case 2 as formulated above, we would have to consider all possible combinations of child nodes of these sets. However, we can also take into account that usually the same page template is used for all pages and that breadcrumbs typically appear on the upper part of a page. The usage of a shared page template implies a shared basic page structure and, hence, we can assume that the breadcrumb is located at the same position in the sequence of its sibling nodes on all pages (e.g. on all pages, the breadcrumb is a second child of the parent element). Furthermore, we cover most cases if we consider only the child nodes at the first child positions because most likely the breadcrumb is placed somewhere

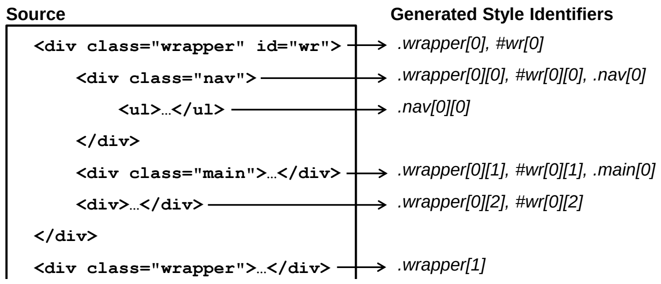

Source
Generated Style Identifiers
<div class="wrapper" id="wr">
<div class="nav">
<ul>…</ul>
</div>
<div class="main">…</div>
<div>…</div>
</div>
<div class="wrapper">…</div>
.wrapper[0], #wr[0]
.wrapper[0][0], #wr[0][0], .nav[0]
.nav[0][0]
.wrapper[0][1], #wr[0][1], .main[0]
.wrapper[0][2], #wr[0][2]
.wrapper[1]

**Figure 67. Computation of style identifers [KEHA13B]**

at the upper part of the page. To generate the initial set of navigation element candidates under these constraints, we compute identifiers for each DOM-element, to which we will refer as style identifiers (Figure 67). The CSS selector syntax is used to distinguish id- and class-attributes: for id-attributes the prefix "." is prepended and for class-attributes the prefix "#"). To improve generality, an additional index is appended to distinguish attributes that appear multiple times on a page (e.g., "#wrapper[1]" denotes the second appearance of the class-attribute "wrapper"). This allows extracting breadcrumbs as well, whose page blocks or parent blocks have class- or id-attributes that are *not* exclusive and appear multiple times on the pages. Style identifiers are transferred to the child nodes by appending an additional index reflecting the position in the sequence of child nodes. All page blocks that share a style identifier represent navigation element candidates. As explained above, we limit the number of child positions that are considered and exclude all style identifiers whose second index exceeds 1 to further reduce the set of candidates.

### 6.4.3.2 Further Reduction of Candidate Set

Prior to feature extraction, the set of navigation element candidates is further reduced by removing candidates that are unlikely to represent breadcrumbs:

1. We assume that hierarchies with more than ten levels are very unlikely and hence, we remove candidates that contain blocks with more than ten hyperlinks.
2. We assume that all links of a breadcrumb navigation are implemented with the same tags and are located at the same level of the DOM tree. Hence, if links in a navigational block differ in their tag paths (e.g., /html/body/div/div/span/a and /html/body/div/div/a), we discard the corresponding navigation element candidate.

After the features are extracted, we use the feature values to remove more candidates that are not likely to represent breadcrumbs:

3. A high $F_{Predecessors}$ indicates many violations of the tree structure and is a clear hint that the candidate does not represent a breadcrumb. Hence, we remove candidates if $F_{Predecessors}$ is above the threshold of 1.2, which was derived from preliminary experiments.

4. We can assume that a site has only a single breadcrumb navigation. It is also obvious that the higher the values of $F_{NumBranches}$, $F_{LogicalDepth}$, $F_{LabelRatio}$ are and the lower the value of $F_{Predecessors}$ is, the more likely the candidate represents a breadcrumb. Hence, we removed all candidates that are not pareto-maximal with respect to $F_{NumBranches}$, $F_{LogicalDepth}$, $F_{LabelRatio}$ and - $F_{Predecessors}$ from the candidate set of a site.

### 6.4.4 Classifier Training and Evaluation

To evaluate the method, 700 randomly selected domains were crawled. We seeded the crawler with the URL http://www.msn.com. The crawler was configured to search for new domains in a depth first manner. To ensure a diverse set of domains, we included only every fifth discovered domain in our evaluation data set. These domains were crawled in breadth-first manner with a page limit of 200.

#### 6.4.4.1 Runtime Performance

Breadcrumb miner ran on a dual core, 3GHz node with 4GBs of RAM. To evaluate how the method scales in the number of pages, we conducted 20 runs with varying number of processed pages per domain. The results plotted in Figure 68 indicate an only sublinear growth of the processing time. This can be explained by the fact that the size of the navigation element candidate set is based on the number of style identifiers. At the same time, the number of style identifiers depends on the number of CSS-rules – which are mainly influenced by the complexity of the visual design and not by the number of pages.

#### 6.4.4.2 Labeled Data Set

If existing, the breadcrumbs of the 700 domains were labeled manually. For this, a random page was selected for each domain and presented in a dedicated Web-frontend that allowed marking the boundaries of page blocks effectively using the mouse pointer. We found that only 4.5% of the breadcrumbs had semantic annotations in the format proposed by schema.org that would allow to retrieve them directly. We removed these breadcrumbs from the data set to ensure a fair comparison with the breadcrumb extraction method of

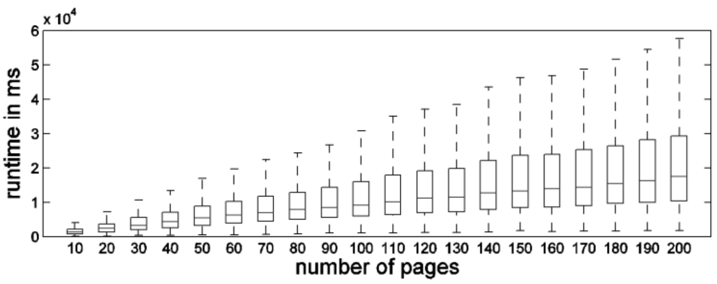


Figure 68. Runtime in dependency of the processed pages [KEHA13B]

**Table 9. Classification Performance [KEHA13B]**

| | Precision | Recall |
|---|---|---|
| Breadcumb Miner C1 | 0.747 | 0.661 |
| Breadcumb Miner C2 | 1 | 0.536 |
| Baseline | 1 | 0.438 |

Google. Since Google is involved in schema.org, it is likely that in case of these domains not the breadcrumb classification method is applied but the semantic annotations are evaluated. The feature vectors for all style identifiers that were found on the labeled page were generated. Style identifiers representing the boundaries of breadcrumbs were labeled as positives, all other style identifiers appearing on the page as negative. Style identifiers appearing on the domain but not present on the page that was labeled were excluded, since we cannot be sure that they do not represent a breadcrumb that is not present on the labeled page but on others (e.g., a breadcrumb of a sub site). [29]

### 6.4.4.3 Cross-Validation

A decision tree learner[30] was trained and evaluated in a 10-fold cross-validation. Since the set of feature vectors contains only a small portion of positive samples and decision tree learners are not capable of dealing with unbalanced data sets well, we oversampled positives and undersampled negatives to achieve a balanced training set of 1000 positives and 1000 negatives in each iteration. Since we are interested in the overall performance of our solution and not only in the performance of the trained classifier, we calculated precision and recall not based on the feature vectors but based on the labeled pages. Otherwise, the conducted preprocessing and candidate set reduction would not be taken into account. For example, a navigation element candidate wrongly discarded at the preprocessing stage would not count as false negative if precision and recall would be computed based on the final feature vectors. We denote this basic configuration as C1 and list the results in Table 9.

Since the focus is on the extraction of hierarchy information to improve the presentation of search results, false positives are more costly than false negatives. In other words, presenting false hierarchy information to users is much more harmful to the business model of the search engine vendor than lacking hierarchy information. We can assume that precision should be close to 1 for a hierarchy extraction approach to be applicable in practice. To make the decision tree learner cost sensitive, we used the MetaCost-method [DOMI99], which can be applied in combination with arbitrary classifiers. The MetaCost-method was configured

[29] The source code of the crawled pages, the labeled breadcrumbs and the generated feature vectors can be downloaded from http://dsn.tm.kit.edu/download/wi2013/data.zip

[30] The C4.5-decision-tree learner of the data mining solution Rapidminer (http://www.rapidminer.com) was used in its default configuration.

with 10 iterations, a false negative penalty of 1 and a false positive penalty of 25. We repeated the 10-fold cross-validation and computed a perfect precision and a recall slightly above 0.5 for this configuration C2 (Table 9).

To the best of our knowledge, details of an applicable breadcrumb mining solution have not been published prior to our own paper [KEHA13B]. However, the search engine Google is able to extract breadcrumbs for some domains and integrate them into the presentation of search results. This allows us to estimate the performance of Google's method without knowing any implementation details: We searched for each of the 700 labeled pages using Google until the page of interest was presented as search result. Then, we could determine whether Google was able to extract the breadcrumb. Since this process is very time-consuming, we checked only pages for which a breadcrumb was labeled and assumed that Google's method does not produce false positives. The results indicate that our own method achieved as higher recall compared to this baseline (Table 9).

## 6.5 Summary

In this chapter, we presented two GRABEX-applications that differ not only in the block graph pattern of interest but also in the way the GRABEX-approach is implemented with respect to classification strategy and reducing the navigation element candidate set. With hierarchical menus and breadcrumb navigations, the presented solutions address the most frequent navigation design patterns that are based on hierarchies and hence, the solutions can be applied to many sites. While reverse engineering breadcrumb navigations is trivial once the navigation element itself is extracted, it is a challenging problem in case of hierarchical menus that was also addressed in this chapter.

The core conclusion of this chapter is that GRABEX-applications allow conducting challenging Web structure mining tasks better than previous solutions with respect to accuracy and computational efficiency in a way that they can be considered as solutions to previously open problems. Evaluations indicate that the first level of hierarchical menus and breadcrumb navigations can be extracted with almost perfect precision and still satisfactory recall. In addition, we found that GRABEX-applications allow detecting site boundaries precisely and more accurately than previous solutions.

The solutions presented in this chapter are applicable in practice and at Web-scale without limitations, since they consume little resources, scale well and neither require page rendering nor downloading other resources than HTML files. Extraction rules as well as classification features are motivated by general Web design rules and not by current Web standards that permanently undergo changes. Hence, they are future-proof and will not be rendered obsolete once new standards emerge.

# 7 Augmentation of Search Results with Site Structure Information

Search engines are a core component of the information source WWW. Without search engines that allow searching billions of indexed pages within seconds, this vast information space would not be accessible. Since search engines are such fundamental tools used by millions of people every day, small improvements have high impact. Current search engines are document-centric in the sense that they consider the WWW as a large collection of HTML-documents. Consequently, search queries are answered by returning a list of ranked HTML-documents. Most research on search engines focusses on how to compose the list of search results, for instance, research on ranking (e.g., [KWCT12]), diversification (e.g., [RABS10]) and personalization (e.g., [SCBW12]) or on how to present search results, for instance, research on clustering results [SFMC12] or improving the visual arrangement [CHKR11].

However, the document-centric approach of current search engines, which results from the lack of appropriate methods for analyzing site structures, contrasts with the way humans experience and interact with the information source WWW. In the eyes of users, the WWW is not a collection of millions of pages but a collection of sites, mostly hierarchically organized. Current search engines aim at satisfying the information need of users by transferring them to the relevant HTML-documents but often, a more appropriate perspective would be trying to transfer users to relevant sites or site sections, i.e. relevant CO-substrees. To the best of our knowledge, except for integrating breadcrumbs of some pages into search result snippets, none of the major search engines is able to analyze and exploit hierarchical COs. In this section, we argue that methods for CO-mining such as the ones presented in this thesis, can provide valuable context information to overcome the limitations of the document centric approach.

## 7.1 Overcoming Limitations of Search Result Presentation by Integration of Hierarchy Information

*Search does not replace navigation.* Research on human searcher behavior reveals that the usage of search engines does not eliminate link-based navigation. In contrast, studies on logged search trails show that a phase of link-based navigation typically follows after users click on search results (e.g., [WHHU10][SIWH10]). We will refer to link-based navigation after keyword search as *post-search navigation*. One could argue that post-search navigation results from limitations of the search engine, which is not always capable of transferring users directly to the desired HTML-page. However, a study indicates that "the perfect search engine is not enough" [TAAK04] and that post-search navigation reflects human searcher behavior rather than search engine limitations. According to Teevan et al. [TAAK04], users prefer to enter less specific search queries in order to reduce cognitive load and approach the target subsequently in a series of navigation steps – a process which is called orienteering by the authors. Besides these findings, there is another obvious argument for the necessity of post-search navigation: in many cases, the information need is too complex to be satisfied by a single page alone [WHHU10]. For example, when a scientist is using a search engine to find information about a conference, he is typically interested in the entire conference Web site and not only in a single page of the site.

*Search engines should consider post-search navigation.* The post-search navigation scenario considered in this section is depicted in Figure 69. "(1.) By entering the search query, the user is transferred to the Search Engine Result Page (SERP). (2.) Then, the user clicks on a promising result and is transferred to the landing page. (3.) The user assesses the information found on the landing page and decides if either to end the search, to return to the SERP or to continue browsing on the target site. Current search engines display short summaries of the target pages on the SERP, called search result snippets. By this they are trying to give hints whether the user can satisfy his information need on the target page." [KEMH13]. We argue that the focus of search result presentation should shift from the

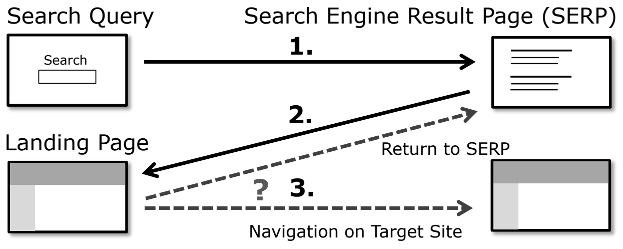
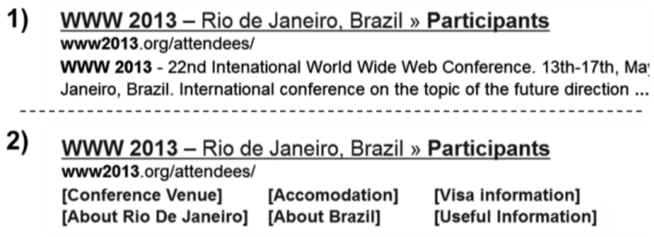


**Figure 69. Scenario: (1) After the search results are displayed, (2) the user is transferred to the landing page of a target site by clicking on a promising result. (3) The user ends his search, returns to the SERP or continues browsing. [KEHA13A]**


1) WWW 2013 – Rio de Janeiro, Brazil » **Participants**
www2013.org/attendees/
**WWW 2013** - 22nd Intenational World Wide Web Conference. 13th-17th, Ma
Janeiro, Brazil. International conference on the topic of the future direction ...

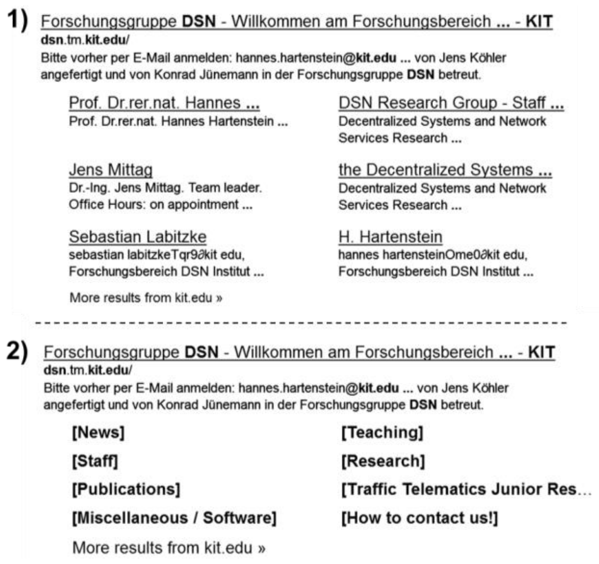
2) WWW 2013 – Rio de Janeiro, Brazil » **Participants**
www2013.org/attendees/
[Conference Venue] [Accomodation] [Visa information]
[About Rio De Janeiro] [About Brazil] [Useful Information]

**Figure 70. Current search results presentation and mockup. (1) Search result snippet for the page "Participants" from the official WWW2013 Web site as presented by google.com (retrieved in). The displayed text is the description provided by the corresponding meta-tag. (2) Mockup in which the description is replaced by links to the child pages. While the original presentation does provide few useful hints about the navigation target, the mockup gives a very good summary. [KEMH13]** (Source page: http://www2013.org/attendees, retrieved on 2013/2/14)

document-centric perspective, which tries to "teleport" (cf. [TAAK04]) users to the final HTML-document, towards a site-oriented perspective, which considers post-search navigation as well. Hence, the presentation of a search result should not only provide information about what a user can find on the target document but also answer the question whether the target document is a good starting point for further exploration. Thus, search result snippets should also include information about where users can go next on the target page.

*Hierarchies determine post-search navigation options*. According to Kalmbach [KALB07] "*Where can I go next?*" is one of the three basic questions Web navigation has to answer (besides "*Where am I?*" and "*What's here?*"). Hence, information gained from reverse-engineering navigation elements provides hints about the possible post-search navigation options if integrated into the presentation of search results. The mockup shown in Figure 70 demonstrates this idea. The upper search result snippet (Figure 70 - 1) shows how the page "Participants" of a conference Web site is presented by google.com. The text used to summarize the page originates from the HTML metadata fields. In the mockup below (Figure 70 - 2), we replaced this text with links to the child nodes of the target page. In contrast to the metadata field, which in this example contains the same site summary on all pages of the site, the child links provide specific and useful information about the contents of the particular site section represented by the "Participants"-page. By this mockup, we do not want to motivate that textual search result summaries should generally be replaced by child node information. Instead, we want to demonstrate that there are cases in which child node information is more useful than traditional page summarizes. However, the question, when to use which kind of information or possibly a combination of both is out of the scope of this thesis.

*Current deep-links do not reflect the entire site content*. Current search engines already integrate additional links in the presentation of some results (Figure 71 - 1). But these, so-called *deep-links* or *site-links*, do not reflect the CO-hierarchy, but the top-ranked pages based on the search-engine's ranking method. Figure 71 – 1 shows how the Web site of our research group is presented by Google. Deep-links to the pages of some individual group


1) Forschungsgruppe **DSN** - Willkommen am Forschungsbereich ... - **KIT**
**dsn**.tm.**kit.edu**/
Bitte vorher per E-Mail anmelden: hannes.hartenstein@**kit.edu** ... von Jens Köhler angefertigt und von Konrad Jünemann in der Forschungsgruppe **DSN** betreut.

| | |
|---|---|
| Prof. Dr.rer.nat. Hannes ... <br> Prof. Dr.rer.nat. Hannes Hartenstein ... | DSN Research Group - Staff ... <br> Decentralized Systems and Network Services Research ... |
| Jens Mittag <br> Dr.-Ing. Jens Mittag. Team leader. Office Hours: on appointment ... | the Decentralized Systems ... <br> Decentralized Systems and Network Services Research ... |
| Sebastian Labitzke <br> sebastian labitzkeTqr9∂kit edu, Forschungsbereich DSN Institut ... | H. Hartenstein <br> hannes hartensteinOme0∂kit edu, Forschungsbereich DSN Institut ... |

More results from kit.edu »

2) Forschungsgruppe **DSN** - Willkommen am Forschungsbereich ... - **KIT**
**dsn**.tm.**kit.edu**/
Bitte vorher per E-Mail anmelden: hannes.hartenstein@**kit.edu** ... von Jens Köhler angefertigt und von Konrad Jünemann in der Forschungsgruppe **DSN** betreut.

| | |
|---|---|
| **[News]** | **[Teaching]** |
| **[Staff]** | **[Research]** |
| **[Publications]** | **[Traffic Telematics Junior Res...** |
| **[Miscellaneous / Software]** | **[How to contact us!]** |

More results from kit.edu »

**Figure 71. Replacing site links with child links. (1) Google supports shortcut links to pages that are considered as the most important pages of the site. The linked pages are computed using Google's ranking algorithms and do not reflect the main topics of the site. (2) The mockup below shows an alternative presentation of the site based on the hierarchy. Users are able to get an overview of the content, even if the text snippet is not understandable. [KEMH13]** (Source site: http://dsn.tm.kit.edu/, retrieved on 2013/2/14)

members are shown in combination with a link to the page listing all employees. While all deep-links point to English pages, they are arranged below the German homepage, because the search engine is not aware of the existence of two separate hierarchies – one for the German contents and one for the English contents. Deep links aim at providing short-cut links to pages of a site that are estimated to be most interesting, but they do not summarize the entire range of site contents well. In contrast, hierarchy links can serve both purposes: providing shortcut links *and* an overview over the entire site content (mockup in Figure 71 – 2). Current deep-links are only useful to those users that are interested in one of the linked pages, while child links are useful to all potential visitors of the site. This demonstrates the limitations of the document-centric approach fundamental to current search engines.

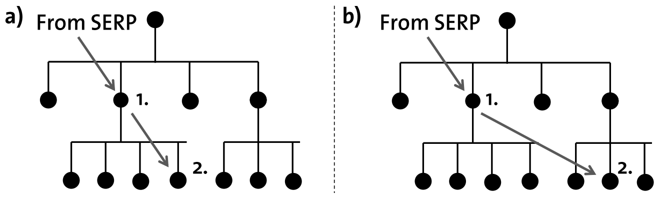


**Figure 72. The user in a) arrives from a search engine and proceeds to a child node, the user in b) proceeds to a non-child node**

## 7.2 Empirical Study

We start by formulating two hypotheses and conduct a study to test whether the hypotheses are supported by the empirical findings.

### 7.2.1 Hypotheses

**Hypothesis 1:** In average, links to child pages are clicked more often than links to non-child pages. In particular, this is true for users arriving from a search engine (Figure 72).

In the previous section, we argued that the CO and in particular the child nodes of a node provide answers to the question “Where can I go next?”. The underlying hypothesis is that users visiting a page prefer child links over non-child links as options for the next navigation step and, hence, the list of child nodes reflects the list of the most-interesting next navigation options well.

**Hypothesis 2:** Current deep links do not generally outperform child links as shortcut links.

We have argued that child links do not only reflect the options for further site exploration but also represent effective shortcut links. While the first aspect is not considered by current search engines because of the document-centric perspective, current search engines do already provide shortcut links. A better overview over the range of contents of the target site or site section should not be bought at the expense of less effective shortcut links. Hence, it should be tested whether child links can serve as shortcut links as well.

### 7.2.2 Experimental setup

To test both hypotheses, we observe how users arriving from a search engine navigate on target sites in relation to the underlying CO. Such kind of analysis requires a large number of post-search clickstreams, which are only available to providers of search engine toolbars or site owners. In addition, the underlying COs must be known. Hence, we studied the usage data of three large-scale Web sites for which we had access to the server log files and could extract the CO from the used content management system (CMS).

"We extracted clickstreams from the server log files of three Web sites for a period of two weeks in May 2012. We considered the sites www.kit.edu (A), www.scc.kit.edu (B) and the site of a municipality responsible for about 25,000 citizens (C). The clickstreams were preprocessed to remove entries from crawlers and to identify individual sessions. At the end 470,827 clickstreams for Site A were analyzed, 89,360 for Site B and 15,953 for Site C. The clickstreams were aligned with a model of the content hierarchy to analyze the navigation behavior in relation to it. At that time the hierarchies consisted of 628 (A), 632 (B) and 154 (C) elements. They had a maximum depth of 10 (A), 9 (B) and 4 (C)." [KEMH13]

## 7.2.3 Testing Hypothesis 1

"For a Web site $W = \{w_1, w_2 \ldots w_n\}$ consisting of $n$ pages $w_1, w_2 \ldots w_n$, the taxonomy is defined by sets $C_{w_l} \subseteq W$ of child pages associated with a parent page $w_i$. A set of j clickstreams $S = \{S_1, S_2 \ldots S_j\}$ is given. The k-th clickstream $S_k = \{s_{k,1}, s_{k,2}, \ldots s_{k,z_k}\}$ describes a user session consisting of $z_k$ consecutive clicks, where $s_{k,l} \in W$ for $l = 1 \ldots z_k$. Regarding the post-search navigation scenario, we are interested in users that arrive from a search engine (which can be identified by the HTTP referer) on a landing page and continue browsing on the site." [KEMH13]. The ratio of clickstreams in which the second visited page is a child page of the first visited page can be calculated as follows, if $1_A(x)$ denotes the indicator function that has the value 1 if $x \in A$ and 0 otherwise:

$$RC_S = \frac{1}{|S|} \sum_{m=1}^{|S|} 1_{C_{s_{m,1}}}(s_{m,2})$$

However, if calculated in this way, $RC_s$ is strongly biased by the ratio of child node links on the landing page. For normalization, we divide by the average number of child nodes on the landing pages and define:

$$AC_S = \frac{\frac{1}{|S|} \sum_{m=1}^{|S|} 1_{C_{s_{m,1}}}(s_{m,2})}{\frac{1}{|S|} \sum_{m=1}^{|S|} |C_{s_{m,1}}|} = \frac{\sum_{m=1}^{|S|} 1_{C_{s_{m,1}}}(s_{m,2})}{\sum_{m=1}^{|S|} |C_{s_{m,1}}|}$$

"In other words $AC_S$ is the ratio of the number of all clicks on child nodes of the landing pages to the number of all child nodes that were shown on the visited landing pages. The same way we computed $AC_S^-$ as the ratio of clicks on non-children to the number of all shown non-child links. By comparing both values we can estimate if it is more likely that a user clicks on a certain link that leads to a child page of the landing page than clicking on a certain link which does not. For both $AC_S$ and $AC_S^-$ we only considered clickstreams with length $> 1$ and landing pages that belong to the Web site taxonomy and had child nodes." [KEMH13].

The results are summarized in Table 10. We found that on average, child-links receive much more clicks than non-child links. The ratio of the average number of clicks on child links to the average number of clicks on non-child links ranges from 4.70 to 11.74 if we consider users arriving from a search engine. This trend is not specific to the search engine scenario, since we observed a similar preference towards child links if all clicks are considered without filtering links incoming from search engines. We used the same method to compare the average number of clicks on parent-links to the average number of clicks on non-parent links (Table 11). For only two of the three Web sites, we observed a preference for parent-links at all, but it is by far not as significant as in the case of child links.

**Conclusion with respect to hypothesis 1:** *Our analysis of three highly frequented Web sites supports hypothesis 1, since we found that users tend to navigate down the hierarchy moving from a page to its child pages. In particular, this is true for users arriving from a search engine.*

## 7.2.4 Testing Hypothesis 2

So-called deep- or site-links of current search engines aim at providing the option for directly jumping to the most relevant pages of a site (Figure 71-1). Site-links are computed by

**Table 10. Child node hits vs. non-child node hits [KEMH13]**

| | Hits from Search Engines | | | | All Hits | | | |
|---|---|---|---|---|---|---|---|---|
| | #Clicks | $AC_S$ | $AC_S^-$ | $\frac{AC_S}{AC_S^-}$ | #Clicks | $AC_S$ | $AC_S^-$ | $\frac{AC_S}{AC_S^-}$ |
| **Site A** | 14918 | 0.0681 | 0.0058 | 11.74 | 91520 | 0.081 | 0.009 | 9 |
| **Site B** | 1672 | 0.0672 | 0.0143 | 4.7 | 19017 | 0.074 | 0.013 | 5.69 |
| **Site C** | 736 | 0.0925 | 0.0085 | 10.88 | 3525 | 0.068 | 0.012 | 5.67 |

**Table 11. Parent node hits vs. non-parent node hits [KEMH13]**

| | Hits from Search Engines | | | | All Hits | | | |
|---|---|---|---|---|---|---|---|---|
| | #Clicks | $AP_S$ | $AP_S^-$ | $\frac{AP_S}{AP_S^-}$ | #Clicks | $AP_S$ | $AP_S^-$ | $\frac{AP_S}{AP_S^-}$ |
| **Site A** | 3711 | 0.0182 | 0.0189 | 0.96 | 93858 | 0.019 | 0.024 | 0.79 |
| **Site B** | 2031 | 0.0429 | 0.0259 | 1.66 | 27930 | 0.04 | 0.021 | 1.9 |
| **Site C** | 294 | 0.068 | 0.021 | 3.24 | 3908 | 0.044 | 0.023 | 1.91 |

search engines using ranking algorithms such as PageRank [PBMW99] and click-through rates. Typically, site-links are presented in combination with the homepage of a site. Despite the fact that site-links do not summarize the content of a site or site section as well as child links do, they could be more useful if they would help users saving more clicks than child links, i.e. if they would be more effective as shortcut links. We want to estimate if this applies for the three sites under examination. Since the ranking-mechanisms of the search engines are not known, we assume that 1) the number of page visits determines the relevance, and 2) the search engine has a perfectly accurate method for determining the most relevant pages of the sites. Hence, we assume that a search engine would display those pages as shortcut links that receive the site-wide most hits. Now, we can consider the following scenario: If we would provide a single shortcut link in combination with each search result, would it be more useful to show the most-visited page of the target site or the first child-link of the target page? To answer this question, we can assume that users that arrived from a search engine and proceeded to another page of the site would have directly accessed the other page in the first place if the corresponding shortcut links would have been present on the SERP. Hence, we can analyze to which kind of pages more people proceeded after arriving from a search engine – to the first child link or the top-ranked page of a site. Figure 73 visualizes the results. The results for considering only a single shortcut link can be found at k=1 for each site. All clickstreams incoming from a search engine with length > 1 were considered. We calculated the fraction of clickstreams in which the second click was on a child page of the landing page on the one hand and the fraction of clickstreams in which the second click was on the page with the site-wide most hits on the other hand. The values at k=2 reflect a scenario in which two shortcut links are displayed in combination with a search result snippet and consequently, the fraction of clicks on the first two child nodes and the fraction of clicks on the links to the two pages with the site-wide most hits are compared.

**Conclusions with respect to hypothesis 2:** *Hypothesis 2 is supported by the results of the study as well. Although we found that site-links would allow users saving more clicks than child nodes in case of site A, the situation is reversed in case of site B, where child links would be more effective as shortcut links. In case of site C, both methods perform with almost identical results.*

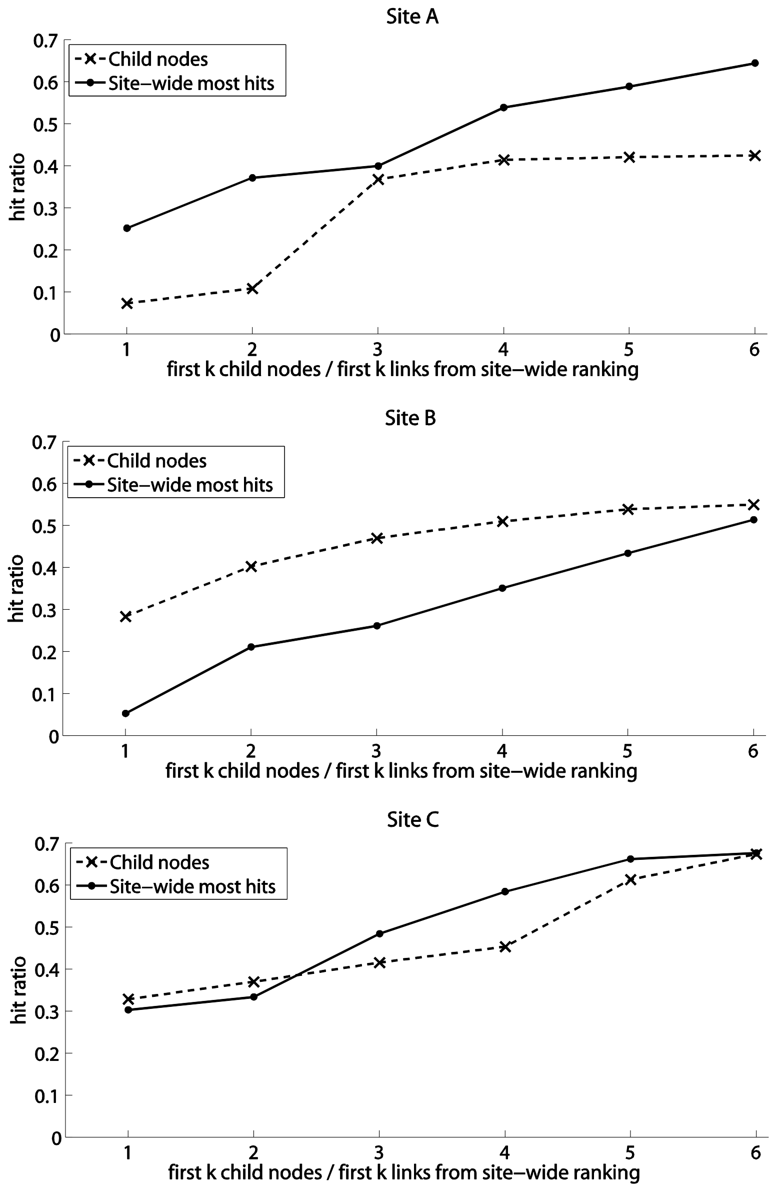


Figure 73. Child links vs. site-wide ranking: For each site, the ratio of hits received by the first k child nodes and the ratio of hits received by the k linked pages with the site-wide most hits are plotted. [KEMH13]

## 7.3 Implications of the Study

To the best of our knowledge, we were the first to analyze browsing behavior in relation to hierarchical COs of real-world Web sites. The findings and implications can be summarized as follows:

- The study indicates that hierarchical COs of Web sites determine the flow of users. Users tend to navigate down the hierarchy from parent pages to child pages. Hence, hierarchical COs allow predicting clickstreams to a certain extent.
- The tendency towards navigating down the hierarchy can be observed for users arriving from a search engine as well. Hence, mining hierarchical COs allows predicting where users navigate next if they explore a target site. By integrating information about the child pages of a target page into the presentation of search results, the next options for further navigation can be anticipated. This would allow users to decide more effectively in advance, whether it is worth visiting the target site or not.
- The study indicates that besides the fact that child links provide more information about the contents of a site or site section, they can be even more effective as shortcut links than the site-links displayed by current search engines.
- The question whether the dominance of parent-child navigation can only be observed for some sites with hierarchical COs or if the results generalize to all such sites was out of the scope of the study. However, automatically determining for a particular site the strength of the influence of the hierarchy on the browsing behavior is an interesting future research topic because it would allow trading off site-links and child-links for different search results individually.

# 8 Conclusions

This thesis was motivated by the observation that content hierarchies as perceived by humans cannot be automatically extracted. Content hierarchies manifest themselves through human observation and are hence difficult to grasp. Nevertheless, we were able to motivate and develop the $O^3$-model that clearly defines and separates different aspects of site structure, including content organization. The $O^3$-model was the foundation to operationalize the fuzzy concept of content hierarchies and to formulate a well-defined problem statement.

The contributions presented in this thesis push forward the boundaries of automatically understanding site structure and provide a basis for future research in this field. Several solutions to previously unsolved problems were presented in this thesis. These solutions are applicable in practice, in particular with respect to accuracy and computational efficiency:

**MenuMiner** (Section 6.2) proved to be a very reliable solution for extracting main menu systems. It can be applied to solve the previously open problem of site boundary detection because a site can be considered as a collection of pages that share the main menu system. Aside from the conducted empirical evaluations, we experienced that MenuMiner is a mature solution. We hardly see any need or room for improvements, except for, maybe, a different trade-off between complexity of the implementation and runtime savings in dependence of the scenario, i.e. implementing a simpler clique extraction method in case of scenarios in which runtime savings are of less importance.

The **rule-based approach of reverse-engineering navigation systems** (Section 6.3) allows identifying the first two hierarchy levels and the active menu items. The approach processes menus delivered by MenuMiner. We tested different configurations that trade off precision and recall in a different way. If a precision around 0.9 can be tolerated, the approach is able to identify the main menu of more than ¾ of the sites. This configuration can be applied, e.g., in combination with ranking algorithms. A more precision-oriented configuration delivered only a single false positive out of 71 extracted main menus at the expense of reduced recall. Though the latter configuration extracts hierarchy information only for the minority of the pages, the results are almost perfectly reliable and can be used to enhance the presentation of search results. Since the first hierarchy level represents the main topics of a Web site, our approach allows to automatically retrieve an accurate list of topics for a significant fraction of sites. In contrast to MenuMiner, there is much room for fine-tuning and further improving menu reverse engineering, e.g., by combining the rule-based approach with machine learning methods. Although we believe that precision and recall can be further increased, the solution is applicable in real-world scenarios and can be regarded as proof of concept.

**BreadcrumbMiner** (Section 6.4) is the first published and applicable solution for extracting breadcrumb navigations. The conducted evaluation indicates that the approach allows identifying breadcrumbs with perfect precision and recall above 0.5. Based on a comparison with the breadcrumbs integrated in the presentation of search results by Google, we conclude that our method is slightly superior. Although not all hierarchically organized sites contain breadcrumb navigations, extracting breadcrumbs is beneficial for understanding site structures because from breadcrumbs the entire hierarchy of a site can be easily retrieved. Our method is solely based on link-analysis and does not yet consider any conventional features such as CSS class names (e.g., CSS classes such as “breadcrumb” or “crumbs” could be used as indicator for breadcrumbs) or separator symbols (“>”). Combining link-based features and conventional features will further improve classification performance.

Based on the findings of this thesis, we can now answer the research questions formulated in the introduction of this thesis as follows:

1. *Which navigation design patterns can be used for CO extraction?*

   We focused on two types of navigation elements that are frequently used and that can be reverse engineered in order to retrieve the hierarchical CO: Hierarchical menus, including menus that implement multiple levels of the hierarchy as well as separate main and local menus on the one hand and breadcrumb navigations on the other hand. Most hierarchically organized sites contain at least one of the two navigation element types. We contributed the proof of concept that both navigation element types can be extracted and reverse engineered in order to gain accurate hierarchy information. If sites contain breadcrumbs, it is more likely that accurate hierarchy information can be extracted with the methods presented in this thesis. While breadcrumbs, if existing, could be extracted for the majority of the sites without any false positives, in case of hierarchical menus, we were only able to retrieve the first level of the hierarchy with a similar precision but lower recall. The second level of the hierarchy is more difficult to extract from hierarchical menus and we only achieved precisions below 0.9. However, if a larger data set of manually labeled Web sites would be made available, e.g., by crowdsourcing menu labelling, machine-learning methods could be combined with the rule-based approach to further improve the method.

2. *How can those navigation design patterns that allow CO extraction be mined? Are the methods accurate enough to produce valuable hierarchy information?*

   First, we took an intuitive approach and derived a basic CO-mining process. The process consists of four tasks: 1) segmenting the pages of a Web site to identify the individual page blocks, 2) classifying the page blocks to identify navigational blocks and their types, e.g., breadcrumb navigations, 3) identifying all navigational blocks that belong to the same navigation element, i.e. aggregating all occurrences of the same breadcrumb from different pages, and 4) reverse-engineering navigation elements in order to retrieve the hierarchical CO. This fundamental process was the basis for surveying related work. Analysis of related work was complicated by inconsistent terminologies in the related fields (cf. Section 4.2) but the $O^3$-modell allowed interpreting and comparing different approaches. We found that for none of the four tasks a general solution exits that could be applied to our problem, though some research has been done on page segmentation and page block classification.

   We made the observation that although it is difficult to automatically derive the type of a navigational block based on a single page, graph representations of the links in navigational blocks that appear on multiple pages show typical patterns that allow deriving the type. Hence, we refined the fundamental CO-mining process and extraction strategy by *first* aggregating navigational blocks that belong together into navigation elements and *then* deriving the type. Although the type can be derived in this way, a prototypical implementation demonstrated that mining navigation elements without prior knowledge about the type is still too challenging if based on the fundamental CO-mining process. We were rethinking the process and proposed the GRABEX-approach, one of the core contributions of this thesis. Instead of trying to extract all navigation elements, GRABEX aims at extracting specific navigation element types, i.e. navigation elements that can be identified by specific patterns of hyperlinking. Basically, GRABEX tries to find these specific patterns among the set of all possible combinations of all possible page blocks. Practically, strategies are needed to reduce the search space, but GRABEX avoids the necessity of an accurate, general navigation element extraction prior to the classification of the navigation element type. Different methods for generating graphs from the links of navigation elements can be applied and suitable methods for reducing the search space can vary in dependence of the GRABEX-application. For discovering characteristic patterns, we implemented a rule-based solution (MenuMiner, Section 6.2) as well as a machine-learning-based approach using decision trees (BreadcrumbMiner, Section 6.4). The conducted evaluations demonstrated that GRABEX-applications allow not only extracting navigation elements accurately but also efficiently. A vast majority of existing page segmentation methods requires computationally expensive visual page rendering and evaluates HTML tag types. The latter results in a strong dependency on the Web standards to which they are tailored to and many solutions can be considered already as deprecated because they were built for outdated standards. GRABEX applications avoid these drawbacks because they are entirely tag-agnostic and do not require page rendering or downloading additional resources such as CSS files or images.

3. *How can we evaluate the correctness of CO mining solutions?*

   Since the solutions presented in this thesis aim extracting hierarchies that reflect the content organization as it is perceived by humans, evaluation can only be conducted against human perception. In our first prototypical implementation (Section 5.3), we had to assess different aspects, such as the quality of page segmentation, the quality of navigational block identification and the quality of navigation element mining for a large number of pages. Hence, we implemented a dedicated visualization tool that allowed inspecting a large number of pages. While we favored efficient evaluation over comparability in this use case, we applied standard measures such as precision and recall in all other evaluations. In contrast to previous CO-mining solutions that have only been evaluated for up to a maximum of 13 selected sites (cf. Section 4.5.4), we applied more rigorous evaluations, including hundreds of randomly gathered sites. This fact illustrates that our solutions are deployable in real-world scenarios.

4. *Which new possibilities arise from the availability of CO-information?*

   We believe that applications in many fields can benefit from CO-mining, e.g., automated sitemap generation, information architecture reverse engineering, search result presentation, search result ranking, web analytics, focused crawling, contextual advertising or Web site transcoding for mobile clients (cf. Section 3.1). In this thesis, we concentrated on the presentation of search results due to this field's relevance. To the best of our knowledge, we presented the first study of post-search navigation with focus on hierarchies. We analyzed three highly-frequented Web sites and gained interesting insights into the influence of hierarchies on users' navigation behavior in particular if they arrive from a search engine. It could be demonstrated that users tend to navigate down towards pages located deeper in the hierarchy, i.e. they gradually move from more general information to more specific information. This has implications for the presentation of search results: We conclude that information about child pages should be integrated into the presentation of search results anticipate more effectively whether it is worth visiting a search result or not.

There are several threads of research that can be picked up in future works. We believe that the rule-based approach of reverse engineering menus should be extended by integrating machine learning methods. We were facing the problem that manually labeling hierarchical menus is very time-consuming and hence, an appropriate data set to train classifiers was not available. However, our rule-based solution in the strict configuration can now be used to produce such a data set and to address the problem with machine learning solutions in the future. BreadcrumbMiner can be applied in the same way and additionally considering conventional features will further improve recall. Another interesting future research field is the development of other GRABEX-instances. They could either focus on hierarchical structures as well or on other organizational schemas such as linear structures (e.g. paginations, cf. Section 2.3). Combining the structural information gained from different GRABEX-instances to achieve a more complete picture of the site organization and a higher coverage is another interesting research topic that could not be addressed in this thesis. Exploiting extracted hierarchy information in the application fields described in Section 3.1 are possible future research directions as well. This includes other ways of integrating hierarchy information into the presentation of search results than the concept of child links proposed in this thesis. For example, if the list of search results contains multiple sibling pages, the parent page could be presented instead to improve the clarity of the presentation. Or, if users search individual sites, the original site hierarchy could be used to arrange the search results instead of returning solely a sorted list.

# A GAPD: Engineering the Functional Organization

Although the purpose of the $O^3$-model is not primarily to support the Web application development process and the focus of this thesis is on Web mining, the $O^3$-model has implications on the design process, too. In this section, we propose Graph Access Pattern Diagrams (GAPD), a method for implementing FO-aspects that results from the $O^3$-model. This excursion into the area of Web application development aims at demonstrating the usefulness of the $O^3$-model. We have published a first draft of the GAPD-notation in [KENU12B] and present an enhanced version in this appendix.

## A.1 Problem Statement

In the design process, navigation design patterns are usually specified implicitly by developing visual prototypes such as wireframes and mockups. There are many different options for designers to realize navigation over hierarchical COs. For example, different hierarchy levels can be represented by distinct navigation elements; child nodes may pop-up when the mouse cursor is moved over the parent item; intermediate levels may be expanded or not, et cetera. In many use cases, all details of menu design and behavior are specified in mandatory style guides. But, in current Web development practice, these patterns of menu behavior cannot be configured but have to be programmed. It is virtually not possible to reuse the menu-generating, server-side code between different applications, because the code heavily depends on the application context, e.g., the used programming language, database system, database connection or database schema. In order to reduce development costs, research with focus on Web engineering has proposed modularization of navigation elements at the granularity of navigation element types [GANM04]. Although this approach is first step towards more efficient implementation of navigation elements, it has limitations because it is difficult to find an adequate set of predefined navigation modules that covers all cases.

## A.2 Approach

The O$^3$-model suggests a solution to this problem. According to the O$^3$-model, navigation design patterns have a CO-, a FO- and a PO-dimension. In the implementation, the CO can be managed by users with administrative rights via a user interface in most application frameworks with CMS functionalities. The FO corresponds to the HTML link structure of the generated navigational blocks and the PO primarily to the CSS-code. In this thesis, we have motivated that navigation design patterns can be considered as patterns that define how the CO is transformed into the FO. Graph Access Pattern Diagrams (GAPD) is a domain-specific language (DSL) for defining such patterns formally. A DSL is a "small, usually declarative, language that offers expressive power focused on a particular problem domain" [DEKV00]. Nussbaumer et al. [NUFG06] argued that DSLs can support a reuse-oriented approach of developing Web applications. GAPD-instances can be directly compiled by a corresponding software module – which will be referred to as *solution building block (SBB)* in accordance with the terminology proposed in [NUFG06]. The GAPD-SBB takes as input the CO, a GAPD-instance and the information, which page is requested, to generate the HTML-source code of the resulting navigational block (Figure 74). GAPD-instances are independent from the used server-side technologies and can be shared between all systems that support the DSL. Nussbaumer et al. [NUFG06] distinguish between the formal language schema – they call it Domain-specific Model (DSM) – and visual notations for manipulation instances,

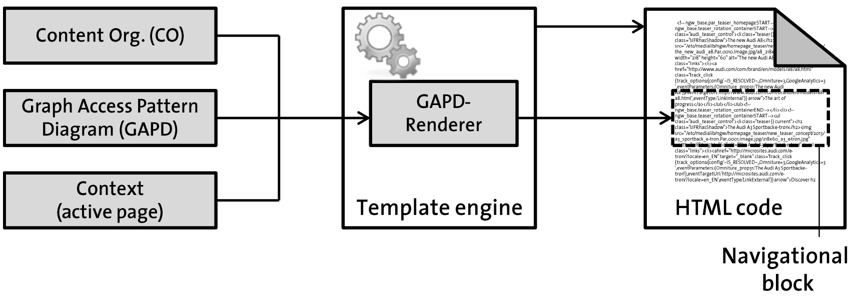


**Figure 74. The GAPD solution building block (SBB) compiles a GAPD-instance in combination with the CO and context information (requested page) to generate the HTML-code of the navigational block. The GAPD-SBB is integrated into a template engine.**

called Domain Interaction Model (DIM). In this section we will introduce GAPD as a visual notation, i.e. DIM. Deriving a serialized representation, i.e. DSM, is trivial and will therefore be omitted. Visual notations are common in software engineering, according to Moody because they "tap into the capabilities of the powerful and highly parallel human visual system" [MOOD10]. We believe that GAPD allows implementing all common variations of navigation elements efficiently without server-side programming, as long as the server provides a GAPD-SBB module. GAPD-instances can be shared between different applications and once the reuse repository of GAPD-instances is large enough, creating an entirely new instance is usually not necessary.

## A.3 The GAPD-Notation

In the GAPD-notation, only position, shape and orientation are used as visual variables (cf. [BERT83]), due to the language's relatively low complexity. Value, color and texture are not used to encode information. A GAPD-instance defines, which hyperlinks are generated on a specific page based on the underlying CO. GAPD assumes that two kinds of relationships appear in the CO: edges that represent parent-child relationships, which are used to model hierarchies and edges that represent predecessor-successor relationships, which are used to model sequences. In GAPD, the arrow symbol is used for both relationships, which can be distinguished by the orientation of the arrow (cf. Figure 75). In case of a hierarchical CO, we assume that the child nodes of each node are ordered and consider the ordered list of children as a sequence (Figure 76(a))[31].

Each GAPD instance has a single starting symbol. A filled disc represents the active node (cf. Figure 75). Starting from the active node, other nodes can be referred to by visualizing their relative location in the CO. For instance, the pattern shown in Figure 76(b) would select the parent node of the active node (single nodes are depicted with the circle symbol). To include a link to a selected node, the square symbol is used. For example, the GAPD-instance in Figure 76(c) would output a link to the parent node of the active node. To select chains of nodes, the asterisk symbol is used (cf. Figure 75). The asterisk symbol selects as many nodes as possible by repeating the relation connecting the symbol (directly or indirectly) with the starting symbol. For example, the pattern showed in Figure 76(d) iteratively selects and outputs the parents of the active node. As a result, all links on the path from the root node to the active node are generated (this pattern can be used to generate a breadcrumb navigation). The maximum number of times that an asterisk symbol is repeated can be specified

[31] According to definition Term **definition 2-5**, the vertices of the CO represent labels. At the same time, each vertex represents a page, too, because the labels are associated with pages. For convenience reasons, we will use the general term "node" in this section.

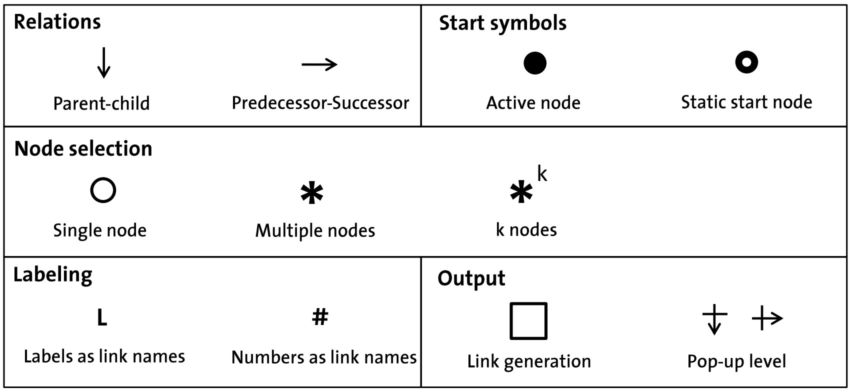


**Figure 75. GAPD visual vocabulary**

explicitly. The GAPD-instance shown in Figure 76(e) generates hyperlinks only to the two direct ancestors of the active node.

Another rule takes precedence over the rule of repeating asterisk nodes as often as possible: the rule of matching as many symbols as possible with nodes of the CO. Figure 76(f) is an example in which this rule is applied. When processing this GAPD-instance, the asterisk symbol is matched with all ancestors but not the root node, which is matched with the single node symbol above instead. If the asterisk symbol would be applied to the root node, the single node symbol would be left unmatched. Figure 76(f) outputs a link to the root node. To achieve this, an alternative start symbol can be used that is not dependent on the active page. This allows generating, e.g., a link to the root node of a CO, even if the active page is not part of the referred CO. It also facilitates simplified notations. The usage of the static start symbol is illustrated in Figure 76(g) and Figure 76(h). This symbol can be considered as a virtual node for selecting the root or the leafs of a hierarchy as well as the first or the last elements in sequences. If this symbol occurs as parent in a parent-child relationship, it represents a virtual (non-existent) parent. Thus, the child of the start symbol is matched with the root node – the only node that does not have a real parent (Figure 76(g)). Similarly, if the static start symbol is part of a predecessor-successor-relationship, the symbol represents a non-existing predecessor, which means that the successor is matched with the first node in case of a sequential CO (Figure 76(h)). The leaf nodes in a hierarchy or the last element in a sequence can be selected in the same way.

Complete GAPD-instances also specify how the link names are generated. The letter "L" means that the CO-labels are used as links names. Alternatively, the "#"-symbol specifies that a consecutive numbering is used to generate link names in case of sequential structures. The link name type can be defined individually for each node or, alternatively, for the entire GAPD-instance, which is surrounded by a bounding box. Figure 76(i) shows an example of a GAPD-instance generating a paging navigation element that provides links to the previous 4 and the next 4 pages of a sequence (cf. Figure 6(d)). "S1" is the identifier of the underlying sequential CO and the sharp symbol specifies that numbers are used as link

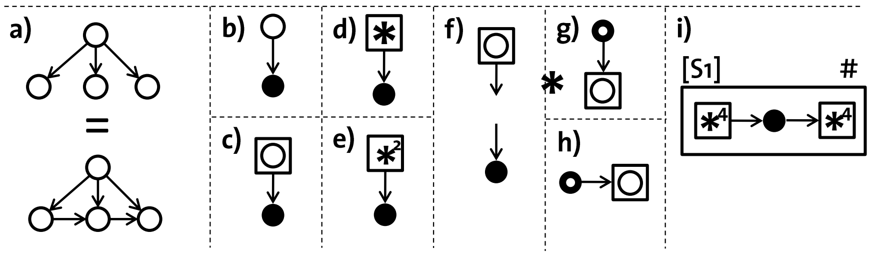


**Figure 76. GAPD examples**

labels. Figure 77 demonstrates the expressiveness of the GAPD vocabulary with a real world example. We manually reverse engineered the navigation elements of a sample page and modeled them as GAPD-instances.

GAPD is designed for hierarchical COs and sequential COs (the latter can be considered as degraded hierarchies). GAPD-instances define which parts of the CO are translated into hyperlinks in dependence of the active page. In other words, GAPD-instances select fragments of the CO. As a result, the original organizational schema is preserved: for instance, the selected nodes of sequential COs maintain sequentially arranged and the selected nodes of a hierarchy remain hierarchically organized as well. These structures can be translated into a hierarchical HTML structure as follows: The resulting hyperlinks are serialized in a depth-first manner. On each descend in the tree structure a new opening tag for an HTML-container[32] is generated. On each ascend a corresponding closing tag is written. In addition, the link to the active page can be tagged, i.e. by application of the "<strong>"-tag. By this, the logical structure is preserved. If the resulting navigational block can be addressed unambiguously by CSS selectors, it is not necessary to add class- or id-attributes to each hyperlink because the HTML-structure alone allows designers to specify different visual properties for each menu level, e.g., by using descendant selectors (cf. [WC11]).

[32] By container elements we mean block level elements [WC99] that can contain other block level elements. To implement navigation elements the unordered list element, i.e. the "<ul>"-tag is usually used. In this case, each hyperlink represents a list entry and is wrapped inside a "<li>"-element.

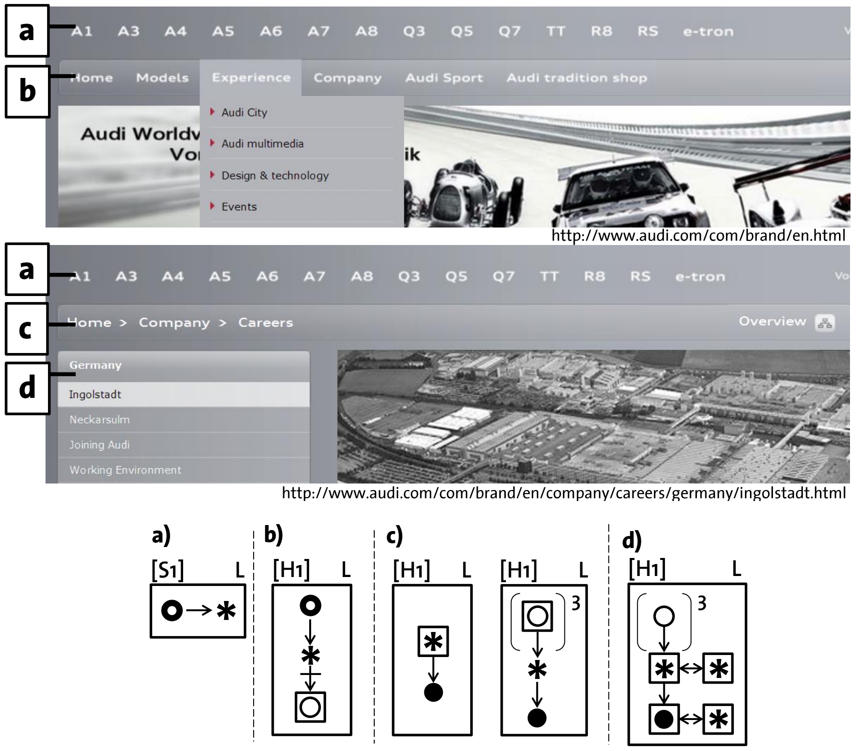

a
A1 A3 A4 A5 A6 A7 A8 Q3 Q5 Q7 TT R8 RS e-tron
b
Home Models Experience Company Audi Sport Audi tradition shop
Audi City
Audi multimedia
Design & technology
Events
http://www.audi.com/com/brand/en.html
a
A1 A3 A4 A5 A6 A7 A8 Q3 Q5 Q7 TT R8 RS e-tron
c
Home > Company > Careers
Overview
d
Germany
Ingolstadt
Neckarsulm
Joining Audi
Working Environment
http://www.audi.com/com/brand/en/company/careers/germany/ingolstadt.html
a)
[S1] L
b)
[H1] L
c)
[H1] L
[H1] L
3
d)
[H1] L
3

Figure 77. Assuming an existing list, i.e. a sequential CO, of the car models of the company (S1) and a hierarchical CO in which the pages of the site are organized (H1), all essential navigation elements of the site can be expressed with GAPD: (a) The complete list of models, (b) a pop-up menu of the first two levels, (c) a simple breadcrumb trail or (left-hand pattern) or a breadcrumb trail showing only the first three levels (right-hand pattern) and (d) a local menu for traversing the lower levels of the hierarchy.

# B Other Related Work

In this appendix, we complete the overview over related fields provided in Section 4 by discussing additional aspects that do not directly impact the methods presented in this thesis. In Appendix B.1, we describe specific page segmentation algorithms in more detail. In Appendix B.2, we analyze block classification methods that do not aim at identifying navigational blocks but at recognizing informative content blocks.

## B.1 Specific Page Segmentation Algorithms

In this Section, we discuss the individual solutions included in the survey in Section 4.3. In dependency of their novelty and impact, selected algorithms are described in more detail, while others are summarized at the end of the section.

**VIPS: Vision-based Page Segmentation Algorithm by Cai et al. [CYWM03]**

The Vision-based Page Segmentation algorithm (VIPS) is a method proposed in 2003 by Cai et al. [CYWM03] that was also applied in a couple of follow-up works and that is frequently used as baseline method to evaluate competing approaches. It is a rule-based method, which incorporates a broad range of features such as tag types, font styles, block sizes and block positions, background colors, et cetera. VIPS generates a hierarchy of blocks for each Web page. The basic VIPS-process is illustrated in Figure 78. In a first step, the *block extraction* phase, the page is segmented into individual blocks by processing the DOM-tree top-down. Thirteen rules are used to decide whether a DOM-node represents a block on its own or whether the algorithm descends further to the child nodes in search of valid blocks. The rules that are applied depend on the node types, e.g., for *table*-elements different rules are used than for *p*-elements. Most of the rules are based on the number and types of the child nodes, only three rules capture visual properties such as the size, background color or font styles. In the next step, the *visual separator detection*, gaps between the found page

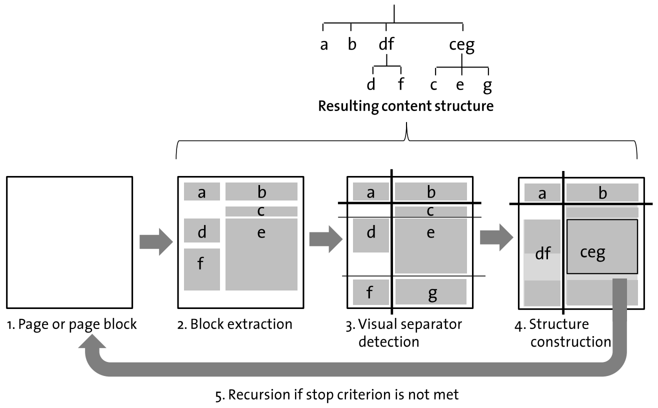

a
b
df
ceg
d
f
c
e
g
Resulting content structure
1. Page or page block
2. Block extraction
3. Visual separator detection
4. Structure construction
5. Recursion if stop criterion is not met

**Figure 78. Overview over the vision-based page segmentation algorithm (VIPS): (1+2) A rule-based approach is used to discover page blocks, (3) visual separators and their weights are extracted, (4) the semantic structure is constructed by iteratively merging blocks in dependence of the separator weights, and (5) a threshold is used to determine whether individual page blocks are further analyzed to extend the tree structure by recursion.**

blocks are analyzed to generate a grid. The grid consists of separators, horizontal and vertical lines, each of which cross the entire width and height respectively of the page. For each separator, a weight is calculated that expresses the strength of the visual separation. For example, distances between the blocks, background colors or differences in the font style are considered to estimate the separator weights. In the *Structure construction* phase, adjacent blocks are iteratively merged, starting from the blocks that are divided by the separators with the lowest weights. This merging process generates the hierarchical semantic page structure. A metric is used to estimate the degree of coherence within each leaf node. If the degree of coherence is lower than a predefined threshold for one of the leaf nodes, a recursion on the leaf is conducted to further refine the hierarchy.

### Web-Page Segmentation based on decision tree learning by Baluja [BALU06]

Baluja presents an interesting approach in [BALU06] that has a couple of unique characteristics. First, the number of cuts or the number of segments respectively is predefined. The use case of the algorithm is to segment a page into 9 tiles that can be selected using the numerical keypad on a mobile phone. Hence, the fixed number of segments is not a limitation but a requirement in this scenario. Second, Baluja's method can produce segments that do not correspond to the DOM-segmentation. This means that the cuts may be conducted in a way that a part of a DOM-element belongs to one block while the rest of the element belongs to another (Figure 80). Other methods usually are able to generate blocks that consist of multiple DOM-elements but they do not cut DOM-elements in halves. A third specific feature of Baluja's method is that color information at pixel-level is analyzed to place the cuts. The method takes a top-down approach that iteratively splits page regions into two parts. Baluja maps this problem onto the problem of training a decision tree classifier. Each DOM-element is considered as a separate class and the area of the DOM-

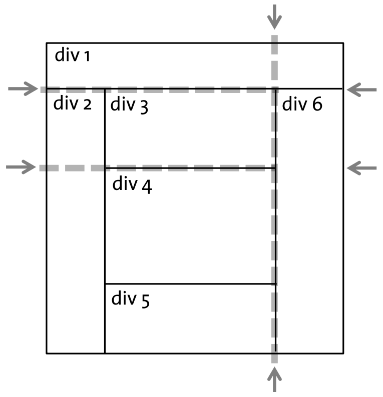
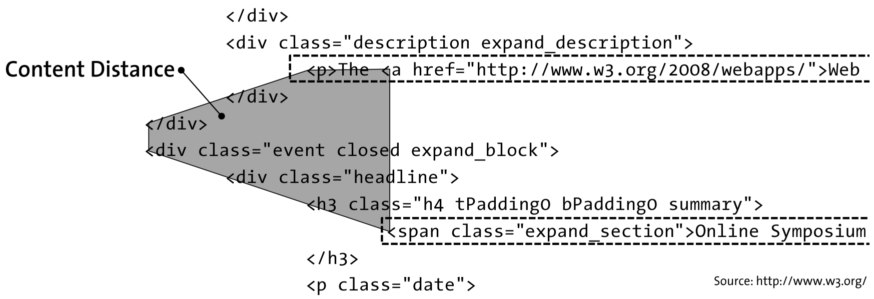


**Figure 80. The segmentation method presented by Baluja [BALU06] can produce cuts that divide DOM elements (e.g., div 1 in this figure).**

element defines the class probability. A standard decision tree learner is employed, which uses the expected information gain as metric to find the best splits along the x- or y-axis. The decision tree learner also considers three other features. One feature penalizes cuts that are not close to 1/3 or 2/3 of the edge lengths of the region to cut. The background is that the method should have a bias toward blocks of uniform size. In addition, the types of the DOM-elements to cut are considered as well as the entropy of the pixels along the cut. In fact, Baluja's method is not a machine learning method in the sense that parameters are learned from a training set. Instead, an algorithm intended to train a classifier is applied in a different context, which is to generate page segmentations instead of classifier instances.

### Block distance metric for page segmentation by Hattori et al. [HHMS07]

Hattori et al. [HHMS07] describe a method that combines a tag-type and block-size based approach for generating the coarse-grained segmentation with an approach for refining the blocks that is based on the structure of the DOM-tree. In the first phase, the DOM tree is processed top-down to discover DIV- and TD-elements that exceed a predefined



```
          </div>
          <div class="description expand_description">
                  <p>The <a href="http://www.w3.org/2008/webapps/">Web
          </div>
  </div>
  <div class="event closed expand_block">
          <div class="headline">
                  <h3 class="h4 tPadding0 bPadding0 summary">
                          <span class="expand_section">Online Symposium
                  </h3>
                  <p class="date">
```

**Figure 79. The gray area illustrates the idea of the content distance metric as defined in [HHMS07]. Not only the number of elements enclosed between two elements influences the metric but also, whether these elements are located at a similar depth of the DOM-tree or not.**

size threshold. Those elements define the preliminary coarse-grained segmentation. However, visual page rendering is not required because the method estimates the element size based on the (deprecated) HTML-attributes width and height or the number of contained characters. For further segmenting the preliminary blocks into the final blocks, the authors propose a metric for measuring the distance between two content elements in the DOM tree. Text data, images, a-elements and scripts are regarded as content elements. The idea of the content distance metric is illustrated in Figure 79. The size of the gray area reflects the value of the content distance metric for the two marked content elements. The design of the metric is inspired by the typical formatting style of HTML code in which each HTML-element creates a new line and the child nodes are indented at each level. Two factors influence the size of the gray area as depicted in Figure 79: First, an increasing number of other HTML elements between two content elements (y-axis in the figure) leads to an increased size of the area and thus to a higher value of the content distance metric. Second, as the enclosed elements are located higher in the DOM-tree, the gray area is becoming larger (x-axis in the figure). The latter aspect captures whether the two elements are located in similar or distant subtrees. If the content distance exceeds a predefined threshold, the segment is split into two separate blocks. The authors also present a way of estimating the optimal threshold value in the paper.

**Site-oriented method by Fernandes et al. [FMSR11]**

Fernandes et al. [FMSR11] present an approach that does not rely on tag semantics (only textual and non-textual nodes are distinguished) or any visual attributes. Instead, only the structure of the DOM-tree influences the segmentation results. In contrast to other approaches, Fernandes et al. conduct the segmentation not on a per-page basis but on a per site basis. The rationale behind this is that through the usage of templates, all pages of the same site have a similar structure. This structure is discovered by merging the DOM trees of all or at least a subset of a site's pages. Then, the segmentation is conducted for the combined DOM-tree (called SOM-tree by the authors). However, Fernandes et al. assume a non-hierarchical segmentation that does not allow nested blocks. The algorithm first pre-processes the DOM-trees of each page individually. In this step, a label for each DOM-element is computed based on the path from the root node to element itself. The element names and all attributes of the ancestor elements are included in the label. Then, the DOM tree is simplified by transforming inner nodes that contain textual content into leaf nodes. In this case, the textual content of the child nodes is appended to the new leaf. Recurring structures such as repeated list items or menu entries are also simplified by merging all child nodes that have a similar structure into a single node. After the pre-processing phase, the SOM-tree is generated. The node labels are used to align the DOM-trees: nodes with the same label from different pages are represented by a single node in the SOM-tree. The SOM-tree is then further processed in order to remove "noisy" nodes. Two heuristics are used: two nodes that are located close to each other in the SOM-tree are merged if both contain textual contents. Also, SOM-tree nodes that were found on only a small number of pages (a threshold of 8 pages is used) are pruned. The remaining leaf nodes of the SOM-tree represent the resulting segmentation.

**Web page segmentation based on text density by Kohlschütter and Nejdl [KONE08]**

Kohlschütter and Nejdl present a method in [KONE08] that is independent of HTML-element semantics in its basic configuration. However, the authors evaluate extensions of the method that *are* element-aware. Kohlschütter and Nejdl propose to adopt segmentation strategies from the fields of linguistics and computer vision to solve the problem of Web page segmentation: region growing [ADBI94] is a common method for image segmentation that iteratively merges adjacent regions into a combined region if they fulfil a similarity requirement (e.g., similar colors). Kohlschütter and Nejdl apply this strategy without considering visual parameters. Instead, their method is based on text density. They consider an HTML-page as a sequence of strings separated by tags. In the basic configuration, no differentiation is made between start and end tags as well as between different tag types. Each text-string is word-wrapped using fixed number of allowed words per line to compute the *text-density*, which is basically the ratio of words to the number of lines. The difference in text-density is used to iteratively merge adjacent strings into larger regions. These regions represent the resulting segmentation. A text-density threshold must be predefined for controlling the segmentation granularity. The authors also implement a version of the algorithm that merges blocks that are “dominated” by the preceding and succeeding block, i.e. groups of three blocks in which the intermediate block with a low density is surrounded by blocks with high densities. The rationale behind this is that single intersecting blocks do not affect the segmentation results. Furthermore, the authors also enhance their method by adding very simple splitting rules based on the HTML element semantics. Interestingly, the element-awareness dramatically boosts segmentation performance.

**Web page segmentation based on graph clustering by Chakrabarti et al. [CHKP08]**

Chakrabarti et al. [CHKP08] transform the problem into an optimization problem on a weighted graph. The basic idea is to consider the DOM-elements of a page as graph nodes and use edge weights to represent the costs for placing two elements in different blocks. However, an element can only be part of a single block. This constraint has to be considered in the formulation of the optimization problem, too. Chakrabarti et al. propose two different approaches. First, they formulate the problem in a way that correlation clustering [BABC04] can be applied and, second, they propose another formulation targeting energy-minimizing graph cuts [BOVZ01]. The cost functions for both approaches must be learned. In their paper, the authors rely on 25 manually segmented pages from randomly selected sites for learning the costs. A broad range of features is considered: the position of a segment, its aspect ratio, the background color, the font sizes, the font types, the average sentence length, the fraction of linked text, the element names, et cetera. GCuts, the energy-minimizing graph cuts approach, also models the distance of two nodes in the DOM-tree. Hence, a parameter must be estimated using additional manually labelled sample-data for balancing the two used cost-functions.

**Others Solutions**

In this Section, we will summarize the specific characteristics of the other solutions that are considered in our review but were not discussed above.

One of the first segmentation approaches is presented by Yang et al. in [YAZH01]. The method is based on the detection of recurring structures (cf. Section 4.3.2). Unlike the later method by Xiang et al. [XIYS06], not only patterns of tag types are mined but visual similarities are also considered. In addition to the tag types, the authors use font sizes, font styles and text lengths to compare DOM-elements. The authors apply a similarity metric to deal with fuzziness. Container elements are compared by comparing the child elements. DBSCAN [EKSX96] is applied to cluster the containers based on the similarities. Then, frequent patterns are detected in the clustered results. If a pattern is found, the element is considered as a so called "list-object" defining a page block.

Chen et al. presented an early rule-based approach [CHMZ03] that includes a broad range of different features. In a first segmentation phase, the DOM-tree is split into predefined high-level blocks such as header, body, sidebar or footer in a top-down manner. For this, the widths of the HTML-elements and their aspect ratios are analyzed. In a second phase, the fined-grained structure is extracted by detecting explicit separators (specific HTML-element types such as hr-elements or table-elements) and implicit separators (gaps between atomic blocks). Chen et al. evaluate their method only indirectly by applying it to transcode Web sites for mobile devices.

Debnath et al. [DMPG05] apply a solely tag-type-based page segmentation method as preprocessing for identifying the main content block. Their method is based on a predefined, ordered list of HTML-element types. The DOM-tree is iteratively split into subsections according to the element types and their order. For example, the first element in the used list was the table-tag. Thus, each table-element is considered as an own block in the first iteration. Discovered blocks are further divided in the successive iterations, e.g., in the second iteration, the tr-element is used as splitting criterion and so on.

Xiang et al. [XIYS06] extended the VIPS-method by preprocessing the DOM-tree and mining recurring block-structures. In the preprocessing phase, specific tags such as line-breaks (br-elements) are removed. Continuous sequences of nodes that are defined as "inline" in the HTML-specification (the authors do not mention, which version of the specification) are merged into a single auxiliary node. After the preprocessing phase, recurring blocks with similar tag structures are identified. The idea is that recurrent patterns determine blocks. For instance, a result page of a search engine contains a list of search results. Each search result is contained within an individual block but all the blocks have a similar DOM-tree structure. Hence, if a repetitive DOM-structure can be discovered, it is very likely that each appearance of this pattern represents a valid page block.

Zou et al. [ZOLT06] adopt the recursive X-Y cut method [HAHP95] to segment Web pages. This method is a general image segmentation algorithm based on bounding boxes. To generate the bounding boxes, called "zones" by the authors, the DOM-tree is processed top-down. Simple segmentation rules are applied that rely mainly on the element types. The authors distinguish inline-elements, tables, insignificant elements and line-breaking elements. The recursive X-Y cut algorithm is applied to induce a tree structure on the atomic zones. According to the authors, the position of the DOM elements and the HTML-attributes *align*, *font-face*, *font-color* and *bgColor*[33] are evaluated to build the tree structure, but details are not reported.

The method by Guo et al. [GMBS07] consists of two phases. In the first phase, the main blocks of a page are discovered by analyzing the alignment of the child elements. If all child elements are consistently aligned either along the x- or along the y-axis, the parent element is considered as block. In the second phase, these blocks are further segmented. This is done by extracting recurring patterns of child element styles. Similar to the idea of Xiang et al. [XIYS06], repetitive structures are considered as separate blocks. However, not the types of the child elements are analyzed but their font-properties (type, style and size).

Burget presents another segmentation method in [BURU09]. He starts with “basic visual areas”, by which he means all text-boxes, images and DOM-elements with non-transparent background or borders. These basic areas are arranged in a grid and adjacent areas are successively merged. First, areas that share the same rows are merged if they have the same background color and no separating border. Second, adjacent areas with the same font-style are merged if they are not visually separated. For detecting visual separators, the VIPS-method [CYWM03] is applied.

Yang et al. claim that their method introduced in [XIYS07] and refined in [YASH09] is based on Gestalt theory. The authors argue that their method uses four laws of Gestalt theory: proximity, similarity, closure and simplicity. However, the authors do not motive why they focus on these laws and not on others (e.g., the ones described in [CHDT02]). Moreover, their implementation rather resembles common segmentation strategies than using novel approaches motivated by Gestalt theory. For instance, the rule of simplicity states that human perception simplifies in a way that the simplest forms are recognized first [CHDT02]. In [YASH09] this rule is implemented by detecting recurring patterns, very similar to the previous method by Xiang et al. [XIYS06].

Vineel describes a method in [VINE09] that conducts the segmentation based on an only 2-dimensional feature space. The first feature is the length of textual content that is arranged under an element (the number of words is used) and the second feature is the Shannon entropy of child element types. Predefined thresholds are set and if both measures exceed the threshold, the corresponding DOM element is considered as block.

Cao et al. [CAML10] present a method that differs from previous approaches because it is solely based on image-processing. This means, rendered Web pages are converted into images before the page segmentation is conducted on the graphic file. A boundary detec-

[33] Note that all evaluated attributes are considered deprecated since HTML 4.0 [WC99] and are typically not used any more.

tion filter is applied to generate a binary image. In a "dividing and shrinking" phase, the image is iteratively split into smaller segments by detecting horizontal and vertical gaps.

Alcic and Conrad [ALCO11] consider DOM-elements as graph nodes and apply graph clustering to generate the page segmentation as done by Chakrabarti et al. [CHKP08] previously. The authors implemented and evaluated three different distance metrics to calculate the edge weights. While a DOM-based distance metric has been proposed before [HHMS07] and geometric distance metrics are commonly used, the application of the third metric, the semantic distance, is an interesting novel contribution in this context. The authors use text processing techniques to estimate the semantic similarity of two text blocks and use the WordNet database [MBFG90] as background knowledge. However, the semantic distance performs worst compared to the other two metrics in the evaluation.

A creative way of segmenting Web pages is proposed by Lin et al. in [LICC11]. They consider Web pages as protein sequences, in which each tag type represents a different amino acid. To find the page segmentation, the authors use an algorithm from the domain of bioinformatics, which separates areas of low complexity (repetitive structures) from areas with high complexity. Hence, the tag types and their order are the only feature exploited.

## B.2 Informative Content Extraction Methods

One of the first methods for extracting informative content blocks was published by Finn et al. [FIKS01]. The authors consider a document as a sequence of two types of tokens: words and tags. They express the aggregated number of tag tokens as a function of the number of all tokens, i.e. the number of tags among the *i*-th first tokens. The authors observed that if the function is plotted, plateaus indicate the main content area (cf. Figure 81). To detect the plateau boundaries, a simple optimization problem is formulated, which is based on the idea of maximizing the number of tags outside the plateau while minimizing the number of tags within the plateau at the same time.

Another early method for extracting informative content blocks is presented by Lin et al. in [LIHO02]. It is assumed that clusters of similar structured pages are known are priori and that the same blocks appear on all pages. The author's idea is to determine the blocks with the lowest average redundancy of the textual content – these blocks are considered as informative blocks. In a first step, Lin et al. filter stop words and apply Porter stemming [PORT80]. The resulting term-document matrix is used to calculated term entropies based on Shannon's formula [SHAN48]. The average entropy of the terms occurring in a block is used to distinguish informative blocks from noisy blocks. The authors assume that blocks with low entropy represent the blocks that contain non-redundant information and keep these blocks. The authors also propose a method to compute the threshold value based on a manually labeled training set.

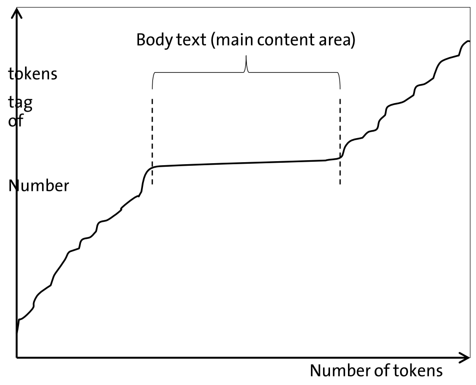


**Figure 81. Finn et al. [FIKS01] analyze the aggregated number of tags as a function of all tokens (tags and words) to discover plateaus.**

A method by Yi et al. [YILL03] is also based on the idea of measuring block entropy and considering high-entropy blocks as redundant and noisy. However, Yi et al. apply a different approach and exploit a larger set of features. In contrast to Lin et al. [LIHO02], the entropy is not computed based on the textual content but on DOM-element attributes. The approach of Yi et al. includes a method to merge DOM-trees for finding recurring blocks – a problem that was not discussed by Lin et al. previously. All pages of a site are merged into a single tree structure, which is called *style tree* by the authors. A style tree consists of two kinds of nodes, element nodes and style nodes (cf. Figure 82) and is generated top-down. For example, the body elements of both DOM trees in Figure 82 contain the same sequence of child nodes (*table, div)*. Hence, they are mapped onto a single style node in the combined style tree. For each style node, the number of occurrences is also saved (2 in this case). In the style tree, the child nodes of a style node – called element nodes – are given by the elements in the sequence that the style node represents. A single element node in the style tree can represent multiple nodes from different source trees. The child sequences of these source nodes can now again be compared to generate the next level of the style tree. If the sequences of child nodes differ, multiple style nodes are created. For example, the *div*-element in Figure 82 appears in conjunction with varying sequences of child nodes and thus, multiple style nodes are inserted below this node in the style tree. All leaf nodes in the style tree are element nodes. After the style tree is generated, the importance of each leaf node is estimated based on Shannon's entropy definition [SHAN48]. For this, the value distributions of the attributes (e.g., link references, background colors, widths or heights) of the source DOM elements are taken into account. More variations among these values increase the value of the used entropy-based importance metric. Entropy is also used to assess the distribution of the style nodes below each element node in the style tree. Both metrics are combined and propagated bottom-up. A predefined threshold is used to distinguish main content nodes from noisy nodes.

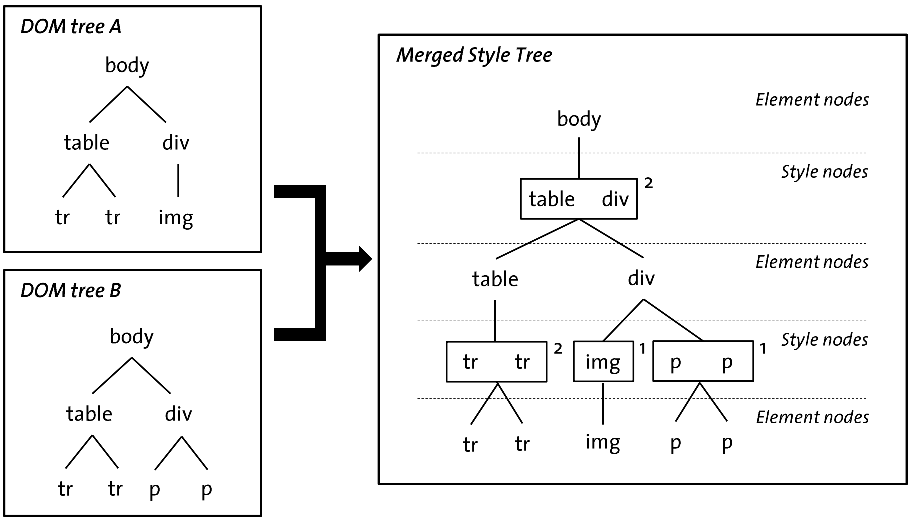


**Figure 82. Merging two DOM trees into a single style tree according to [YILL03]**

Different entropy measures are the foundation of a method by Kao et al. [KAHC05] as well. The method returns up to *k* informative blocks for each page, with *k* being a predefined parameter. In contrast to most other methods, Kao et al. consider not only the blocks containing informative text passages as informative blocks but also blocks that contain tables of contents (TOCs). The main routine of their method selects informative blocks of a preprocessed page in a greedy way. In each iteration, the block with the highest aggregated *anchor precision index (API)* is selected (the authors also introduce other measures but the API performs best in their evaluation). The API is computed for each hyperlink: the inverse values of the term entropies (calculated as in [LIHO02]) for all terms that appear at the same time in the link text and on the linked page are summed up. Before a block is selected, it is checked whether it does not violate one of the *type constraints*. The first type constraint requires that the average term entropy of the textual content within the block is below a predefined threshold (cf. [LIHO02], low entropy indicates informative sections). The second type constraint requires that the average API of the links within the block exceeds a threshold. If the type constraints are not violated, the block is added to the candidate block set and the algorithm proceeds with the next iteration until *k* candidates are found. The candidate set is further refined by merging additional elements into the candidate blocks. This is done by checking whether direct sibling nodes or siblings of the first ancestor that actually has siblings pass the type constraints. If this is true, the siblings are added to the candidate block as well. In a last step, parts of the merged subtrees are pruned again if they violate the type constraints and the remaining blocks are considered as informative blocks.

Song et al. [SLWM04] present a systematic machine-learning approach in conjunction with an interesting user study. In the user study, they distinguish four levels of block importance with redundant blocks at the one end of the scale and the blocks containing the main information at the other end. The study included 600 pages from 405 different sites, resulting in a total number of 4539 blocks, which are obtained with the VIPS algorithm [CYWM03]. Five human assessors assigned one of the four importance values to each of the blocks. It was found that in 92.9% of the cases at least three assessors agree on the importance of the blocks. If the importance levels 3 and 4 are combined, this value increases to 99.5%. Hence, the three level approach is applied to train the classifiers. A dozen of classical features are used, including image number, image sizes, link text length, text length, et cetera. For classification, a linear Support Vector Machine (SVM), a non-linear SVM with RBF kernel and a RBF neural network are trained.

Debnath et al. [DMPG05] present three different rule-based algorithms for extracting main content blocks, targeting sites from the news domain. The first algorithm, *ContentExtractor,* is a site-oriented method that detects non-redundant blocks, i.e. blocks that are not repeated on multiple pages of the site. The feature vectors of two blocks are compared using cosine similarity and a threshold determines whether two blocks are considered as being identical on not. The feature vectors consist of simple quantities: the number of words within the block, the number of tables, the number of links, et cetera. The second algorithm, *FeatureExtractor,* allows specifying target features, e.g., the word number. Then, each feature of a block is normalized by divided by the maximum observed value of this feature. All blocks of a page for which the sum of the normalized target features exceeds the sum of the normalized non-target features, form a new “winner” set (the rationale behind this approach is not documented in the paper). The normalization is repeated, now, based on the maximal observed values *within* the winner set. The block with the highest value of the target feature is selected as informative content block. The third algorithm, *K-FeatureExtractor* is similar to FeatureExtractor but it is able to output multiple informative blocks for each page. This is achieved by applying k-means clustering to the winner set and selecting the highest ranked cluster.

Grotton presents another method that considers a document as token sequence in [GOTT08]. Grotton experiments with two token granularities: word tokens and character tokens. In each case, tokens that correspond to tags are distinguished from tokens that correspond to textual content. Content tokens are initialized with 1 and tag tokens are initialized with 0. The token sequence is iteratively blurred using weighted averages until the values start to settle. Afterwards, a threshold is used to decide for each token whether it is regarded as informative content or not. Grotton also proposes a modified version of his solution which ignores hyperlinks in order to improve classification performance on wiki-like sites.

A straightforward solution by Burget and Rudolfova [BURU09] (cf. Appendix A for the page segmentation method) distinguishes 9 different classes of main content blocks, including h1 (headline 1), h2 (headline 2), subtitle, author, date, et cetera. Nine font-based, spatial, textual and color-based features are used to train a decision tree classifier (based on Qinlan’s C4.5 algorithm [QUIN93]).

Another machine-learning based solution that operates on token sequences is proposed by Pasternack and Roth in [PARO09]. Tokenization of documents is conducted by filtering uncommon tags, deleting *script*- and *style*-elements, applying porter stemming [PORT80] and converting all numbers to 1. The authors use a naïve Bayes classifier to assign a value in the range [-0.5, 0.5] to each token, with a higher value expressing a higher probability that the token belongs to the main content block. The classifier is trained on only two features: token trigrams and the most-recent-unclosed tag. To identify a coherent subsequence corresponding to the main content block, the maximum subsequence, i.e. the coherent subsequence with the maximimal sum of the contained tokens, is extracted. Besides the basic supervised approach, the authors also propose a semi-supervised extension in order to boost classification performance. Before the semi-supervised approach is applied on a site, the classifier is iteratively refined with the use of unlabeled pages from the same site. This is done by applying the classifier and selecting a predefined number of pages for which the predictions are estimated to be most-likely correct. The weight of trigrams that are located close to the main content borders in this subset is increased in order to guarantee a higher influence in subsequent iteration rounds. The maximum subsequence extraction is repeated with the token values being corrected by the learned weights. Ten such iterations are conducted before the main content extraction is considered as finalized.

**Evaluation of Informative Content Extraction**

The method for informative content extraction by Lin et al. [LIHO02] is evaluated empirically for selected sites. However, the results are difficult to interpret because although precision and recall values are provided, these metrics are not applied intuitively to measure the block classification performance. Instead, it is measured whether the same terms appear in the manually labelled and automatically detected informative blocks.

Yi et al. [YILL03] show that their main content extraction solution can improve Web page classification and clustering quality but they do not compare their method to previous block classification solutions.

Song et al. [SLWM04] conducted an elaborate and convincing evaluation of their method, which assigns one of three importance classes to each page block. Five human assessors labeled 4539 blocks extracted from 600 pages originating from 405 different sites. Only blocks for which at least three human assessors assigned the same class label were included in the evaluation. Hence, 4517 individual blocks remained. Three classifiers were evaluated in a 5-fold cross-validation. Precision and recall are provided for each of the three class labels separately as well as the overall F1-performance. With an F1-value of 0.790, the SVM classifier outperforms the other methods. To assess the classification quality, the authors compare this value to the performance of the worst human assessor who reached an only slightly higher F1-value of 0.792. Thus, the gap between the human and machine-based importance assessment is only small.

Kao et al. [KAHC05], which distinguish informative blocks and redundant blocks, evaluate the classification performance for a couple of selected sites from the news domain. The data set is manually labeled to compute precision and recall. The bar plots, which are included in the paper, indicate good performance of the method for most sites. Average precision and recall values are not provided and the method is not compared to previous solutions.

Debnath et al. [DMPG05], which propose three different main content extraction algorithms in their paper, evaluate their methods on 15 selected sites, mostly from the news domain. The number of crawled pages ranges from 100 for some sites to a maximum number of 415 for another site. The blocks are manually labelled. Precision and recall are calculated. All three methods perform well, with F1-measures mostly above 0.9 and often close to 1. These are surprisingly good results, achieved with rather simple algorithms. However, it is unclear if these methods would work this well if applied on today's more complex Web sites.

Grotton [GOTT08] evaluates his own method and three baseline methods, including the solution by Finn et al. [FIKS01][34]. Since the considered methods are all token-based (documents are considered as sequences of tokens) rather than block based, precision and recall are calculated on a token basis and not on a block basis. To compute these metrics, the number of tokens that appear at the same time in the retrieved and relevant texts is considered. The evaluation data consists of a set of 65 manually labeled pages (the sources are not mentioned) and 13 larger sets from selected sites, labeled semi-automatically. The authors provide F1-measures for all data set / algorithm combinations individually but no average values that would allow to assess the overall performance. The results are mixed: the performance of all tested algorithms (including the novel ones proposed by Grotton) strongly depend on the data set. Grotton's methods do also not significantly outperform previous solutions, although the evaluation indicates a slight improvement.

The method by Burget and Rudolfová [BURU09], which aims at distinguishing different classes of main content areas, is evaluated empirically as well. On unseen sites, the performance of the method is disappointing, with the block type "date" reaching the highest F1-measure of 0.769 among the block types.

Pasternack and Roth [PARO09] evaluate their supervised and semi-supervised algorithms on two different data sets. One data set includes samples from five news sites. For each site, 50 samples were manually labeled. The second data set originates from 40 different sites and contains five manually labeled samples from each site. A large training set was used: from each of the 12 training sites, 2000 samples were gathered. To label this amount of training data, a dedicated wrapper was written for each of the twelve sites. The supervised method performs well on both data sets with F1-scores around 0.95. The average recall is almost perfect, but precision values are below 0.95. The semi-supervised approach improves the performance to very good F1 scores above 0.97.

[34] The other two methods are not discussed in this section because (1) the method by Gupta et al. [GKNG03] is not considered as a block classification method and (2) the method used by Pinto et al. [PBCC02] is very similar to the solution present in [FIKS01].

# C BlockCliqueFinder-Algorithm

***The description of the BlockCliqueFinder below is an excerpt from our paper "MenuMiner: Revealing the Information Architecture of Large Web Sites by Analyzing Maximal Cliques" [KENU12A]. However, we adapted some terms to be in line with the terminology used in this thesis and adjusted the references.***

The BlockCliqueFinder-algorithm returns the local *SC** (cf. Section 6.2.2), which is the set containing all block graph cliques to which a block 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 Section 6.2.2, the algorithm successively computes the largest clique of blocks from different pages and removes all links that are part of that block graph clique. It terminates if no block graph clique of a minimal size of 3 is found. If multiple block graph cliques with maximal size are possible, the algorithm returns the one whose blocks are most uniform concerning their placement in the DOM tree.

One, multiple or no block graph clique can be embedded in a web graph clique. For each page of a web graph clique *WC* that contains a block graph clique *SC*, a block 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 83 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

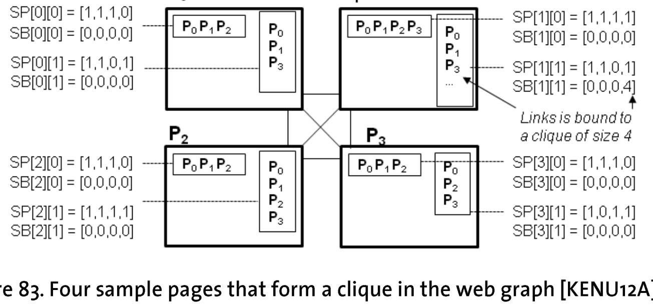


**Figure 83. Four sample pages that form a clique in the web graph [KENU12A]**

that is currently processed, so only blocks containing links to $p_0$ have to be considered on the other pages. The blocks are encoded as a three dimensional array *SP[i][j][k]* that has the value 1 if the block *j* on page *i* contains a hyperlink to page *k* and the value 0 otherwise. The blocks on pages $p_1$-$p_3$ can contain links that are already assigned to a block graph clique when previous pages were processed. Since according to Section 6.2.2, larger block graph cliques are preferred, these links have to be removed from the original clique and assigned to a new block graph clique if the new clique is larger. For this the array *SB[i][j][k]* contains the minimum sizes that a new block graph clique must have to include a hyperlink or target page respectively. In the example shown in Figure 83 only the link to $p_3$ in block 1 of $p_1$ is bound to another clique, which has the size of 4. The other links can be assigned to block graph cliques of any size so SB has the value 0.

Each web graph clique is compared to all blocks of $p_0$. If the web graph clique and the block *j* share at least three pages, the block 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 *BlockCliqueFinder* algorithm. The example includes only a single web graph clique and both blocks 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 (Algorithm 1, line 04, function *GetStateWithMaxScore*). From the states with the maximal number of pages the state associated with the smallest level is selected.

In Figure 84 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 block graph clique that does not contain the page of the next level (Algorithm 1, lines 10-12). New states are also created for each block of the page representing the next level (line 13). If such a block 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 (line 17). Before a new state is added to the state list it is tested to see if links are already bound to larger block graph cliques. If that is the case these links are removed (lines 21-23). If the block graph 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 blocks that are included in a state. If an equal state (regarding *SP* and *SB*) associated with the same level already exists, only the state with the higher block uniformity is kept. To compute the uniformity we align the DOM paths of all blocks. Node names, class- or id-attributes that differ are replaced by wildcards. A lower number of wildcards indicates that the blocks 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 84(3) the state that is joined with *SP[2][0]* is kept because its blocks are more uniform.

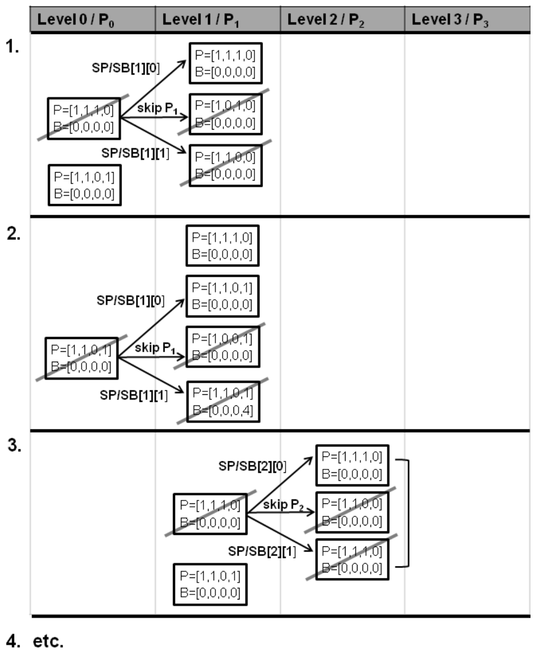


**Figure 84. Illustration of the BlockCliqueFinder-algorithm [KENU12A]**

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 block graph cliques of *SC** of 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: BlockCliqueFinder**

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

Output: Maximal block graph clique

```
EndStates ← new List; MaxEndScore ← 0;
WHILE (States.count > 0)
  NewStates ← new List;
  S ← GetStateWithMaxScore(States);
  IF ($\Sigma_j$ S.pages[j] < MaxEndScore)
    RETURN BestState(EndStates);
  NextLevel = S.level+1;
  States.remove(S);
  IF (S.level < M - 1)
    SN ← S.copy();
    SN.pages[NextLevel] ← 0;
    NewStates.add(SN)
  FOR(all blocks k of page $P_{NextLevel}$)
    SN ← new State;
    FOR(all $P_j$)
      SN.pages[j] ← min(S.pages[j], SP[NextLevel][k][j]);
      SN.binding[j]  ← max(S.binding [j], SB[NextLevel][k][j]);
    SN.level  ← NextLevel
    NewStates.add(SN);
  FOR(all States SN in NewStates)
    FOR(i = 0…M)
      IF(SN.binding[i] > $\Sigma_j$ SN.pages[j])
        SN.pages[i] ← 0; i ← 0;
      IF ($\Sigma_j$ SN.pages[j] > 2)
      IF (SN.level < M-1) AddToStateList(States, SN);
      ELSE IF ($\Sigma_j$ SN.pages[j] = MaxEndScore)
        EndStates.add(SN);
      ELSE IF ($\Sigma_j$ SN.pages[j] > MaxEndScore)
        MaxEndScore ← $\Sigma_j$ SN.pages[j];
        EndStates ←  new List; EndStates.add(SN);
```

# REFERENCES

[ADBI94] R. Adams and L. Bischof, “Seeded region growing,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 6, pp. 641–647, 1994.

[AKYE13] M. E. Akpınar and Y. Yeşilada, “Heuristic Role Detection of Visual Elements of Web Pages,” in *Web Engineering*, vol. 7977, F. Daniel, P. Dolog, and Q. Li, Eds. Springer Berlin Heidelberg, 2013, pp. 123–131.

[ALCO11] S. Alcic and S. Conrad, “Page segmentation by web content clustering,” in *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, Sogndal, Norway, 2011, pp. 1–9.

[ALCZ10] A. Alshukri, F. Coenen, and M. Zito, “Web-Site Boundary Detection,” in *Advances in Data Mining. Applications and Theoretical Aspects*, vol. 6171, P. Perner, Ed. Springer Berlin Heidelberg, 2010, pp. 529–543.

[ALCZ11] A. Alshukri, F. Coenen, and M. Zito, “Incremental web-site boundary detection using random walks,” in *Machine Learning and Data Mining in Pattern Recognition*, Springer, 2011, pp. 414–427.

[BABC04] N. Bansal, A. Blum, and S. Chawla, “Correlation Clustering,” *Machine Learning*, vol. 56, no. 1–3, pp. 89–113, Jul. 2004.

[BALU06] S. Baluja, “Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework,” in *Proceedings of the 15th international conference on World Wide Web*, Edinburgh, Scotland, 2006, pp. 33–42.

[BERT83] J. Bertin, *Semiology of graphics*. University of Wisconsin Press, 1983.

[BLMI98] A. Blum and T. Mitchell, “Combining labeled and unlabeled data with co-training,” in *Proceedings of the eleventh annual conference on Computational learning theory*, Madison, Wisconsin, USA, 1998, pp. 92–100.

[BOIK05] B. Boiko, *Content management bible*, 2nd ed. Indianapolis, IN: Wiley Pub, 2005.

[BORS92] R. A. Botafogo, E. Rivlin, and B. Shneiderman, “Structural analysis of hypertexts: identifying hierarchies and useful metrics,” *ACM Transactions on Information Systems (TOIS)*, vol. 10, no. 2, pp. 142–180, 1992.

[BOVZ01] Y. Boykov, O. Veksler, and R. Zabih, “Fast approximate energy minimization via graph cuts,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 11, pp. 1222–1239, 2001.

[BRKE73] C. Bron and J. Kerbosch, “Algorithm 457: finding all cliques of an undirected graph,” *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[BURU09] R. Burget and I. Rudolfova, “Web Page Element Classification Based on Visual Features,” *Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on*, pp. 67–72, Apr. 2009.

[CAML10] J. Cao, B. Mao, and J. Luo, “A segmentation method for web page analysis using shrinking and dividing,” *Int. J. Parallel Emerg. Distrib. Syst.*, vol. 25, no. 2, pp. 93–104, 2010.

[CDMM12] S. Choudhary et al., “Crawling rich internet applications: the state of the art,” in *Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research*, Toronto, Ontario, Canada, 2012, pp. 146–160.

[CHBD99] S. Chakrabarti, M. van den Berg, and B. Dom, “Focused crawling: a new approach to topic-specific Web resource discovery,” *Computer Networks*, vol. 31, no. 11–16, pp. 1623–1640, May. 1999.

[CHDT02] D. Chang, L. Dooley, and J. E. Tuovinen, “Gestalt theory in visual screen design: a new look at an old subject,” in *Proceedings of the Seventh world conference on computers in education conference on Computers in education: Australian topics - Volume 8*, Copenhagen, Denmark, 2002, pp. 5–12.

[CHEN76] P. P.-S. Chen, “The entity-relationship model - toward a unified view of data,” *ACM Trans. Database Syst.*, vol. 1, no. 1, pp. 9–36, 1976.

[CHKP08] D. Chakrabarti, R. Kumar, and K. Punera, “A graph-theoretic approach to webpage segmentation,” in *Proceedings of the 17th international conference on World Wide Web*, Beijing, China, 2008, pp. 377–386.

[CHKR11] F. Chierichetti, R. Kumar, and P. Raghavan, “Optimizing two-dimensional search results presentation,” in *Proceedings of the fourth ACM international conference on Web search and data mining*, New York, NY, USA, 2011, pp. 257–266.

[CHMZ03] Y. Chen, W.-Y. Ma, and H.-J. Zhang, “Detecting web page structure for adaptive viewing on small form factor devices,” in *Proceedings of the 12th international conference on World Wide Web*, Budapest, Hungary, 2003, pp. 225–233.

[CKGS06] C. H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan, “A Survey of Web Information Extraction Systems,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 18, no. 10, pp. 1411–1428, Oct. 2006.

[CLWP03] Z. Chen, S. Liu, L. Wenyin, G. Pu, and W.-Y. Ma, “Building a web thesaurus from web link structure,” in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, 2003, pp. 48–55.

[CLZC05] X. Chen, M. Li, W. Zhao, and D.-Y. Chen, “Discovering conceptual page hierarchy of a web site from user traversal history,” in *Advanced Data Mining and Applications*, Springer, 2005, pp. 536–543.

[CONK87] J. Conklin, “Hypertext: An Introduction and Survey,” *Computer*, vol. 20, no. 9, pp. 17–41, 1987.

[CYWM03] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma, *VIPS: a vision based page segmentation algorithm*. Microsoft technical report, MSR-TR-2003-79, 2003.

[CZSZ01] 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*, New York, NY, USA, 2001, pp. 587–596.

[DANI03] D. R. Danielson, “Transitional volatility in web navigation,” *Information Technology and Society, 1(3),Special Issue on Web Navigation*, pp. 131–158, 2003.

[DEKV00] A. van Deursen, P. Klint, and J. Visser, “Domain-specific languages: an annotated bibliography,” *SIGPLAN Not.*, vol. 35, no. 6, pp. 26–36, Jun. 2000.

[DLCT00] Y. Diao, H. Lu, S. Chen, and Z. Tian, "Toward Learning Based Web Query Processing," in *Proceedings of the 26th International Conference on Very Large Data Bases*, 2000, pp. 317–328.

[DMPG05] S. Debnath, P. Mitra, N. Pal, and C. L. Giles, "Automatic identification of informative sections of Web pages," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 9, pp. 1233–1246, Sep. 2005.

[DOMI99] P. Domingos, "MetaCost: A General Method for Making Classifiers Cost-Sensitive," in *In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, 1999, pp. 155–164.

[DUCH00] S. Dumais and H. Chen, "Hierarchical classification of Web content," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, 2000, pp. 256–263.

[EKSX96] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *In Second International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

[ENMG06] Eduarda Mendes Rodrigues, Natasa Milic-Frayling, Martin Hicks, and Gavin Smyth, *Link Structure Graphs for Representing and Analyzing Web Sites*. Microsoft Research, 2006, p. 11.

[FAHB09] F. Fauzi, J.-L. Hong, and M. Belkhatir, "Webpage segmentation for extracting images and their surrounding contextual information," in *Proceedings of the 17th ACM international conference on Multimedia*, Beijing, China, 2009, pp. 649–652.

[FGKS12] T. Furche, G. Grasso, A. Kravchenko, and C. Schallhart, "Turn the Page: Automated Traversal of Paginated Websites," in *Web Engineering*, vol. 7387, M. Brambilla, T. Tokuda, and R. Tolksdorf, Eds. Springer Berlin Heidelberg, 2012, pp. 332–346.

[FIKS01] A. Finn, N. Kushmerick, and B. Smyth, "Fact or fiction: Content classification for digital libraries," in *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.

[FMSR11] D. Fernandes, E. S. de Moura, A. S. da Silva, B. Ribeiro-Neto, and E. Braga, "A site oriented method for segmenting web pages," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, New York, NY, USA, 2011, pp. 215–224.

[GANM04] M. Gaedke, M. Nussbaumer, and J. Meinecke, "WSLS: A Service-Based System for Reuse-Oriented Web Engineering," *Engineering Advanced Web Applications, Rinton Press, NJ*, pp. 26–37, 2004.

[GARR11] J. J. Garrett, *The elements of user experience: user-centered design for the Web and beyond*. Berkeley, CA: New Riders, 2011.

[GECO00] D. M. Germán and D. D. Cowan, "Towards a Unified Catalog of Hypermedia Design Patterns," in *Proc. of the 33rd Hawaii International Conference on System Sciences (HICSS), Maui*, 2000.

[GEST83] D. Gentner and A. L. Stevens, Eds., *Mental models*. Hillsdale, N.J: Erlbaum, 1983.

[GKNG03] 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*, New York, NY, USA, 2003, pp. 207–214.

[GLPG11] P. K. GM, K. P. Leela, M. Parsana, and S. Garg, "Learning website hierarchies for keyword enrichment in contextual advertising," in *Proceedings of the fourth ACM international conference on Web search and data mining*, Hong Kong, China, 2011, pp. 425–434.

[GMBS07] H. Guo, J. Mahmud, Y. Borodin, A. Stent, and I. V. Ramakrishnan, “A General Approach for Partitioning Web Page Content Based on Geometric and Style Information,” *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, vol. 2, pp. 929–933, Sep. 2007.

[GOTT08] T. Gottron, “Content Code Blurring: A New Approach to Content Extraction,” in *Proceedings of the 2008 19th International Conference on Database and Expert Systems Application*, Washington, DC, USA, 2008, pp. 29–33.

[GOZI06] G. Goos and W. Zimmermann, *Grundlagen und funktionales Programmieren: mit 30 Tabellen*. Berlin [u.a.]: Springer, 2006.

[HAHP95] J. Ha, R. M. Haralick, and I. T. Phillips, “Recursive X-Y cut using bounding boxes of connected components,” *Proceedings of the Third International Conference on Document Analysis and Recognition*, vol. 2, pp. 952–955 vol.2, Aug. 1995.

[HHMS07] 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*, New York, NY, USA, 2007, pp. 361–370.

[HOEX12] Q. Ho, J. Eisenstein, and E. P. Xing, “Document hierarchies from text and links,” in *Proceedings of the 21st international conference on World Wide Web*, Lyon, France, 2012, pp. 739–748.

[HUAR85] L. Hubert and P. Arabie, “Comparing partitions,” *Journal of Classification*, vol. 2, no. 1, pp. 193–218, Dec. 1985.

[HYCD09] W.-C. Hu, H.-J. Yang, L. Chen, and R. Deshmukh, “Web Content Adaptation for Internet-Enabled Mobile Handheld Devices,” in *Proc. of the Midwest Instruction and Computing Symposium (MICS 2009)*, 2009.

[JACC12] P. Jaccard, “The distribution of the flora in the alpine zone. 1,” *New phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[JIJO00] Jinbeom Kang and Joongmin Choi, “Block Classification of a Web Page by Using a Combination of Multiple Classifiers,” *Fourth International Conference on Networked Computing and Advanced Information Management, NCM ’08*, vol. 2, pp. 290–295, 2.

[JOFU87] W. P. Jones and G. W. Furnas, “Pictures of relevance: a geometric analysis of similarity measures,” *J. Am. Soc. Inf. Sci.*, vol. 38, no. 6, pp. 420–442, Nov. 1987.

[KAHC05] H.-Y. Kao, J.-M. Ho, and M.-S. Chen, “WISDOM: Web intrapage informative structure mining based on document object model,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 5, pp. 614–627, May. 2005.

[KALB07] J. Kalbach, *Designing Web navigation*. Sebastopol, (Calif.): O’Reilly, 2007.

[KEHA13A] M. Keller and H. Hartenstein, “Mining Taxonomies from Web Menus: Rule-Based Concepts and Algorithms,” in *Proceedings of the 13th International Conference on Web Engineering*, Aalborg, Denmark, 2013.

[KEHA13B] M. Keller and H. Hartenstein, “GRABEX: A Graph-Based Method for Web Site Block Classification and its Application on Mining Breadcrumb Trails,” in *Proceedings of the 2013 IEEE/WIC/ACM International Conference on Web Intelligence*, Atlanta, USA, 2013.

[KEMH13] M. Keller, P. Mühlschlegel, and H. Hartenstein, “Search result presentation: supporting post-search navigation by integration of taxonomy data,” in *Proceedings of the 22nd international conference on World Wide Web companion*, Rio de Janeiro, Brazil, 2013, pp. 1269–1274.

[KENU10] M. Keller and M. Nussbaumer, “CSS Code Quality: A Metric for Abstractness; Or Why Humans Beat Machines in CSS Coding,” *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the*, pp. 116–121, Sep. 2010.

[KENU11] M. Keller and M. Nussbaumer, “Beyond the Web Graph: Mining the Information Architecture of the WWW with Navigation Structure Graphs,” in *Proceedings of the 2011 International Conference on Emerging Intelligent Data and Web Technologies*, Tirana, Albania, 2011, pp. 99–106.

[KENU12A] M. Keller and M. Nussbaumer, “MenuMiner: revealing the information architecture of large web sites by analyzing maximal cliques,” in *Proceedings of the 21st International Conference Companion on World Wide Web*, Lyon, France, 2012, pp. 1025–1034.

[KENU12B] M. Keller and M. Nussbaumer, “Graph Access Pattern Diagrams (GAP-D): Towards a Unified Approach for Modeling Navigation over Hierarchical, Linear and Networked Structures,” in *Current Trends in Web Engineering*, vol. 7059, A. Harth and N. Koch, Eds. Springer Berlin Heidelberg, 2012, pp. 155–158.

[KLEI99] J. M. Kleinberg, “Authoritative sources in a hyperlinked environment,” *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[KONE08] C. Kohlschütter and W. Nejdl, “A densitometric approach to web page segmentation,” in *Proceedings of the 17th ACM conference on Information and knowledge management*, New York, NY, USA, 2008, pp. 1173–1182.

[KUPT06] 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.

[KWCT12] C. Kang, X. Wang, Y. Chang, and B. Tseng, “Learning to rank with multi-aspect relevance for vertical search,” in *Proceedings of the fifth ACM international conference on Web search and data mining*, New York, NY, USA, 2012, pp. 453–462.

[LEKL04] C. H. Lee, M.-Y. Kan, and S. Lai, “Stylistic and lexical co-training for web block classification,” in *Proceedings of the 6th annual ACM international workshop on Web information and data management*, 2004, pp. 136–143.

[LICC11] S.-H. Lin, K.-P. Chu, and C.-M. Chiu, “Automatic sitemaps generation: Exploring website structures using block extraction and hyperlink analysis,” *Expert Systems with Applications*, vol. 38, no. 4, pp. 3944–3958, Apr. 2011.

[LIHO02] S.-H. Lin and J.-M. Ho, “Discovering informative content blocks from Web documents,” in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 588–593.

[LINL04] Z. Liu, W. K. Ng, and E.-P. Lim, “An Automated Algorithm for Extracting Website Skeleton,” in *Proceedings of the 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004), Jeju Island, Korea*, 2004, pp. 799–811.

[LYHL10] C. X. 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*, Springer, 2010, pp. 222–229.

[MAUN04] K. Makino and T. Uno, “New algorithms for enumerating all maximal cliques,” in *Algorithm Theory-SWAT 2004*, Springer, 2004, pp. 260–272.

[MBFG90] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, “WordNet: An on-line lexical database,” *International Journal of Lexicography*, vol. 3, pp. 235–244, 1990.

[MEBD08] A. Mesbah, E. Bozdag, and A. van Deursen, “Crawling AJAX by Inferring User Interface State Changes,” *Web Engineering, 2008. ICWE ’08. Eighth International Conference on*, pp. 122–134, Jul. 2008.

[MEMF07] E. Mendes Rodrigues, N. Milic-Frayling, and B. Fortuna, “Detection of Web Subsites: Concepts, Algorithms, and Evaluation Issues,” in *Proceedings of the*

*IEEE/WIC/ACM International Conference on Web Intelligence*, Washington, DC, USA, 2007, pp. 66–73.

[MFRS10] E. S. de Moura, D. Fernandes, B. Ribeiro-Neto, A. S. da Silva, and M. A. Gonçalves, “Using structural information to improve search in Web collections,” *J. Am. Soc. Inf. Sci. Technol.*, vol. 61, no. 12, pp. 2503–2513, 2010.

[MKKG08] J. Madhavan, D. Ko, Ł. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy, “Google’s Deep Web crawl,” *Proc. VLDB Endow.*, vol. 1, no. 2, pp. 1241–1252, 2008.

[MLGR98] D. Mladeni’c and M. Grobelnik, “Feature selection for classification based on text hierarchy,” in *Text and the Web, Conference on Automated Learning and Discovery CONALD-98*, 1998.

[MOOD10] D. L. Moody, “The ‘physics’ of notations: a scientific approach to designing visual notations in software engineering,” in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, New York, NY, USA, 2010, pp. 485–486.

[MORO06] P. Morville and L. Rosenfeld, *Information architecture for the World Wide Web*. Sebastopol, (Calif.): O’Reilly, 2006.

[NELS75] T. H. Nelson, *Computer lib: you can and must understand computers now*. South Bend: Nelson, 1975.

[NEMN11] M. Nebeling, F. Matulic, and M. C. Norrie, “Metrics for the evaluation of news site content layout in large-screen contexts,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Vancouver, BC, Canada, 2011, pp. 1511–1520.

[NUFG06] M. Nussbaumer, P. Freudenstein, and M. Gaedke, “The impact of domain-specific languages for assembling web applications,” *The Journal Engineering Letters*, vol. 13, pp. 387–396, 2006.

[PALM92] S. E. Palmer, “Common region: A new principle of perceptual grouping,” *Cognitive Psychology*, vol. 24, no. 3, pp. 436–447, Jul. 1992.

[PARO09] J. Pasternack and D. Roth, “Extracting article text from the web with maximum subsequence segmentation,” in *Proceedings of the 18th international conference on World wide web*, New York, NY, USA, 2009, pp. 971–980.

[PBCC02] D. Pinto et al., “QuASM: a system for question answering using semi-structured data,” in *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, New York, NY, USA, 2002, pp. 46–55.

[PBMW99] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank Citation Ranking: Bringing Order to the Web*. 1999.

[PICD08] N. Pinto, D. D. Cox, and J. J. DiCarlo, “Why is Real-World Visual Object Recognition Hard?,” *PLoS Computational Biology*, vol. 4, no. 1, p. e27, 2008.

[PORT80] M. F. Porter, “An algorithm for suffix stripping,” *Program: electronic library and information systems*, vol. 14, no. 3, pp. 130–137, 1980.

[QUIN93] J. R. Quinlan, *C4. 5: programs for machine learning*, vol. 1. Morgan kaufmann, 1993.

[RABS10] D. Rafiei, K. Bharat, and A. Shukla, “Diversifying web search results,” in *Proceedings of the 19th international conference on World wide web*, New York, NY, USA, 2010, pp. 781–790.

[RAND71] W. M. Rand, “Objective criteria for the evaluation of clustering methods,” *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[ROSG97] G. Rossi, D. Schwabe, and A. Garrido, “Design reuse in hypermedia applications development,” in *Proceedings of the eighth ACM conference on Hypertext*, Southampton, United Kingdom, 1997, pp. 57–66.

[SAWY75] G. Salton, A. Wong, and C. S. Yang, “A vector space model for automatic indexing,” *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[SCBW12] D. Sontag, K. Collins-Thompson, P. N. Bennett, R. W. White, S. Dumais, and B. Billerbeck, “Probabilistic models for personalizing web search,” in *Proceedings of the fifth ACM international conference on Web search and data mining*, New York, NY, USA, 2012, pp. 433–442.

[SCZS06] H. A. Schildt, S. A. Zahra, and A. Sillanpää, “Scholarly Communities in Entrepreneurship Research: A Co-Citation Analysis,” *Entrepreneurship Theory and Practice*, vol. 30, no. 3, pp. 399–415, 2006.

[SENE05] P. Senellart, “Identifying websites with flow simulation,” in *Web Engineering*, Springer, 2005, pp. 124–129.

[SFMC12] U. Scaiella, P. Ferragina, A. Marino, and M. Ciaramita, “Topical clustering of search results,” in *Proceedings of the fifth ACM international conference on Web search and data mining*, New York, NY, USA, 2012, pp. 223–232.

[SHAN48] C. E. Shannon, “A Mathematical Theory of Communication,” *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, Jul. 1948.

[SHMA00] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[SIWH10] A. Singla, R. White, and J. Huang, “Studying trailfinding algorithms for enhanced web search,” in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2010, pp. 443–450.

[SLCO12] H. Sleiman and R. Corchuelo, “A Survey on Region Extractors From Web Documents,” *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, no. 99, pp. 1–1, 2012.

[SLWM04] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma, “Learning block importance models for web pages,” in *Proceedings of the 13th international conference on World Wide Web*, New York, NY, USA, 2004, pp. 203–211.

[STGH03] A. Strehl and J. Ghosh, “Cluster ensembles---a knowledge reuse framework for combining multiple partitions,” *The Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2003.

[TAAK04] J. Teevan, C. Alvarado, M. S. Ackerman, and D. R. Karger, “The perfect search engine is not enough: a study of orienteering behavior in directed search,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2004, pp. 415–422.

[TBLM94] T. Berners-Lee, L. Masinter, and M. McCahill, “Uniform Resource Locators (URL), RFC 1738,” 1994. [Online]. Available: http://www.w3.org/Addressing/rfc1738.txt. [Accessed: 20-Nov-2013].

[TEDN99] Ted Nelson, “Ted Nelson’s Computer Paradigm, Expressed as One-Liners,” 1999. [Online]. Available: http://xanadu.com.au/ted/TN/WRITINGS/TCOMPARADIGM/tedCompOneLiners.html. [Accessed: 29-Aug-2013].

[TOXB13] A. Toxboe, “User Interface Design Pattern Library,” 2013. [Online]. Available: http://ui-patterns.com/patterns. [Accessed: 28-Aug-2013].

[VINE09] G. Vineel, “Web page DOM node characterization and its application to page segmentation,” in *Proceedings of the 3rd IEEE international conference on*

*Internet multimedia services architecture and applications*, Bangalore, India, 2009, pp. 325–330.

[WALZ07] C. Wang, J. Lu, and G. Zhang, “Mining key information of web pages: A method and its application,” *Expert Syst. Appl.*, vol. 33, no. 2, pp. 425–433, Aug. 2007.

[WAWA07] S. Wagner and D. Wagner, *Comparing clusterings: an overview*. Universität Karlsruhe, Fakultät für Informatik, 2007.

[WC00] W3C, “G140: Separating information and structure from presentation to enable different presentations | Techniques for WCAG 2.0.” [Online]. Available: http://www.w3.org/TR/2014/NOTE-WCAG20-TECHS-20140311/G140. [Accessed: 16-Apr-2014].

[WC11] W3C, “Selectors Level 3 - W3C Recommendation 29,” 2011. [Online]. Available: http://www.w3.org/TR/css3-selectors/. [Accessed: 10-Dec-2013].

[WC99] W3C, “HTML 4.01 Specification - W3C Recommendation,” 1999. [Online]. Available: http://www.w3.org/TR/1999/REC-html401-19991224/. [Accessed: 10-Dec-2013].

[WCRE99] W3C Recommendation 5-May-1999, “Web Content Accessibility Guidelines 1.0,” 1999. [Online]. Available: http://www.w3.org/TR/WCAG10/. [Accessed: 27-Aug-2013].

[WERT38] M. Wertheimer, “Laws of organization in perceptual forms,” *A source book of Gestalt psychology, First published as Untersuchungen zur Lehre von der Gestalt II, in Psycologische Forschung, 4, 301-350.*, pp. 71–88, 1938.

[WHHU10] R. W. White and J. Huang, “Assessing the scenic route: measuring the value of search trails in web logs,” in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2010, pp. 587–594.

[XIYS06] P. Xiang, X. Yang, and Y. Shi, “Effective Page Segmentation Combining Pattern Analysis and Visual Separators for Browsing on Small Screens,” in *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, 2006, pp. 831–840.

[XIYS07] P. Xiang, X. Yang, and Y. Shi, “Web Page Segmentation Based on Gestalt Theory,” *Multimedia and Expo, 2007 IEEE International Conference on*, pp. 2253–2256, Jul. 2007.

[YALI09] C. C. Yang and N. Liu, “Web site topic-hierarchy generation based on link structure,” *J. Am. Soc. Inf. Sci. Technol.*, vol. 60, no. 3, pp. 495–508, Mar. 2009.

[YASH09] X. Yang and Y. Shi, “Enhanced gestalt theory guided web page segmentation for mobile browsing,” in *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies, WI-IAT’09*, 2009, vol. 3, pp. 46–49.

[YAZH01] Y. Yang and H. Zhang, “HTML Page Analysis Based on Visual Cues,” in *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, 2001, p. 859.

[YILL03] L. Yi, B. Liu, and X. Li, “Eliminating noisy information in Web pages for data mining,” in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, 2003, pp. 296–305.

[YJZN10] Q. Yang, P. Jiang, C. Zhang, and Z. Niu, “Reconstruct Logical Hierarchical Sitemap for Related Entity Finding,” in *The Nineteenth Text REtrieval Conf. (TREC 2010)*, 2010.

[YZYH12] L. Ying, X. Zhou, J. Yuan, and Y. Huang, “A novel focused crawler based on breadcrumb navigation,” in *Advances in Swarm Intelligence*, Springer, 2012, pp. 264–271.

[ZOLT06] J. Zou, D. Le, and G. R. Thoma, “Combining DOM tree and geometric layout analysis for online medical journal article segmentation,” in *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, Chapel Hill, NC, USA, 2006, pp. 119–128.