

# Automatic Extraction of Generic Web Page Components

Sephorah Graves Eriksson  
sephorah@stp.ling.uu.se

Master's Thesis in Computational Linguistics  
Språkteknologiprogrammet  
(Language Engineering Programme)  
Uppsala University · Department of Linguistics

19th April 2004

**Supervisor:**  
Mats Dahllöf, Uppsala University

## **Abstract**

Information on the World Wide Web is accessed not just visually, but also automatically by systems, such as search engines and alternative browsers (e.g. screen readers and voice browsers), which extract and present relevant data automatically from Web pages. In most cases extraction cannot be performed directly, since HTML documents of today lack adequate semantic markup. This thesis proposes a method that converts an HTML document to a semantically enhanced document representation, from which generic document components can be extracted for further knowledge exploration or alternative presentation. The document is parsed and iteratively smaller nodes are mapped to a classification ontology, which then are aggregated into larger segments, thereby creating a semantically enhanced parse tree. Segment boundaries are detected based on visual and document segments, such as images and headings. Experimental results of the implementation show that document components, such as headings and menus, can be extracted directly from the semantic parse tree. The heading extraction experiment achieved recall and precision rates of 88% and 91%. The recall and precision rates for the menu extraction experiment were 90%.

# Acknowledgements

I would like to thank Fredrik Larsson and Niclas Bergström at Phoneticom, Uppsala, for getting me started on this subject, that is, the extraction of generic document components from Web pages. I would also like to thank Roussanka Loukanova, Department of Linguistics at the Uppsala Univesity, who encouraged me to participate in the course in Information Retrieval in Tübingen, which gave me the strength and motivation to continue with this thesis. My supervisor Mats Dahllöf I would like to thank for seeing me through to the end. I also want to thank my husband Markus for his immense support.

# Contents

<b>Acknowledgements</b>	<b>i</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Purpose	1
1.2 Outline	2
<b>2 Background</b>	<b>3</b>
2.1 Information Extraction	3
2.2 Web Document Analysis	4
2.2.1 Document Analysis	5
2.2.2 Web Document Structure	6
2.3 Document Segmentation	7
2.3.1 Structural Tags	8
2.3.2 Visual Separators	9
2.3.3 Coherent Components	10
2.3.4 Contextual Cues	10
2.4 Object Classification	11
2.4.1 Ontology-Based Classification	11
2.5 Areas of Application	12
<b>3 Implementation</b>	<b>13</b>
3.1 The General Approach	13
3.2 Related Work	14
3.3 Training data	14
3.4 Preprocessing	15
3.4.1 Retrieving the Web Page	15
3.4.2 Parsing the HTML	15
3.4.3 Extracting Frame Content	16
3.4.4 Detecting Visual Boundaries	16
3.4.5 Preprocessing Algorithm	18
3.5 Segmenting the Document	19
3.5.1 Boundary Markers	19
3.5.2 Segmentation Algorithm	21
3.6 Classifying the Objects	22
3.6.1 Basic Objects	22
3.6.2 Composite Objects	24
3.7 Extracting Generic Objects	25
3.8 Notes on the Implementation	26

<b>4</b>	<b>Experimental Results</b>	<b>27</b>
4.1	Evaluation Measures . . . . .	27
4.2	Test data . . . . .	27
4.3	Extraction Experiment I . . . . .	28
4.3.1	Extraction Target: "Headings" . . . . .	29
4.3.2	Results Experiment I . . . . .	29
4.4	Extraction Experiment II . . . . .	32
4.4.1	Extraction Target: "Site Navigation Menus" . . . . .	32
4.4.2	Results Experiment II . . . . .	33
4.4.3	Discussion Experiment II . . . . .	34
4.5	Results compared to Related Work . . . . .	35
<b>5</b>	<b>Conclusions</b>	<b>36</b>
	<b>References</b>	<b>38</b>

# List of Figures

1.1	Example of a Web Page from the Test Corpus, <a href="http://www.uppsala.se">http://www.uppsala.se</a> . . . . .	2
2.1	A Section of an HTML Document. . . . .	7
2.2	Example of a DOM Tree from Part of the HTML Code in Figure 2.1. . . . .	8
2.3	Part of the Car-Ad Extraction Ontology from Embley et al. (1999a), page 8. . . . .	11
3.1	Example of an HTML::Element, in B), for the HTML Code in A). . . . .	16
3.2	Example of a Visual Boundary Generated by an Image. . . . .	17
3.3	Example of a Boundary Generated by an "Empty Element" in a "Select Form". . . . .	17
3.4	Example of a BO with an Embedded Element. . . . .	20
3.5	Example of a BO with Multiple Embedded Elements. . . . .	20
3.6	Example of a BO that really is a CO, it has Embedded BOs. . . . .	20
3.7	Example of the Semantic Representation for the Link in Figure 3.1. . . . .	22
3.8	Example of the Classification Ontology for Basic Object. . . . .	23
3.9	Example of the Classification Ontology for Composite Objects. . . . .	25
4.1	Example of a Test Page, <a href="http://www.foreningsbanken.se">http://www.foreningsbanken.se</a> . . . . .	28
4.2	Example of a False Positive Generated in Experiment I and Experiment II. A) Displays Part of a Column from the Page <a href="http://svt.se/nyheter/">http://svt.se/nyheter/</a> . B) the Conceptual Model of the Content in A). C) Shows How the Impementaion Analyzes A). . . . .	31
4.3	Example an Ambiguous Component that generates a False Positive in Experiment II, the Logo from <a href="http://svt.se/">http://svt.se/</a> . . . . .	34

# List of Tables

4.1	The Results of the "Heading" Extraction, Presented in Average Recall and Precision for Two Test Pages from each Test Site (February 2004). . . . .	30
4.2	The Results of the "Site Navigation Extraction", Presented in Average Recall and Precision for Two Test Pages from each Test Site (February 2004). . . . .	33

# 1 Introduction

The World Wide Web is used today generally as a forum for visually publishing and distributing information. It is comparable to an electronic library with static data, accessed through browsers such as Internet Explorer or Netscape. But browsing the Web and finding the information that we seek is not always as straightforward as we could wish. Successful interaction is largely affected by the information architecture of a page/site. Rosenfeld and Morville (1998) state "users don't notice the information architecture of a site unless it isn't working". But, the interpretation of information on the Web, and the interaction with the Web is not just affected by "poor" information architecture. It is also affected by accessing or using information on the Web in a way the "author" had not intended, or foreseen it to be.

Information on the Web is in general created for visual access, and visual browsing. But far from all interactions with the Web are performed visually. Information on the Web can also be accessed automatically by computer applications such as a search engines, "intelligent" agents, or a phone monitor acting as an alternative browser, which all locate, filter, and collect information, more or less intelligently, for a specific users need. These automatic approaches of interacting with the Web are increasing rapidly. But they are rarely considered in the design and construction of a Web site/page.

In order to facilitate automatic Web information exploitation, Web pages need to incorporate more semantic knowledge on its document structure and content. Researchers are developing methods to enhance the structures of the Web with semantic information. Alternative markup formats, such as XML and RDF, are being launched. But the ongoing use of HTML, and the enormous amount of available HTML documents, calls for methods that can automatically exploit HTML in order to deduce information from the content of a page.

## 1.1 Purpose

The aim of this thesis is to develop and implement an algorithm that automatically identifies blocks of information in a Web page. The information blocks that are to be identified are the "building blocks" of a Web page. The "building blocks" are generic document components found in all types of Web pages, see for example the Web page in figure 1.1. Generic document components are, for example, "headings", "paragraphs", "menus", "forms", "frames", and "pictures". These objects are defined by their general function or purpose, regardless of their potential domain specific role. For example, the general function of an "interactive form" can be "text search", and its domain specific role could be "search publications", or "find person", depending on what type of data that is searched.



Figure 1.1: Example of a Web Page from the Test Corpus, <http://www.uppsala.se>.

In order to locate information of interest in a Web page, the objects must first be identified, that is, the document must be segmented and segments must be classified. The general idea governing the segmentation is to identify boundaries that indicate the beginning and end of a document component. Boundaries are detected directly from the HTML markup or via visual, contextual and linguistic constraints derived from the HTML markup and the content of a page. An ontology, specified for the domain of "generic Web page components", is used to annotate segments. The annotations are in turn used to identify the objects of interest.

The performance of the generic Web document analysis algorithm, proposed by this thesis, is tested by embedding the algorithm in to a "wrapper", that is, an application that automatically extracts information from documents given a specific user request.

The general notion governing this thesis is that identifying the generic document structure of a Web page facilitates the interpretation of Web page content for further information exploration, for example, for Web page summarization, page adaptation to alternative browsers, or even as a help in the creation of parallel alignments.

## 1.2 Outline

In Chapter 2 a brief introduction to "Information Extraction" is given, and methods relevant for extracting information from a Web page are presented more in detail. The implementation, and Web document analysis approach taken by this thesis, is presented thoroughly in Chapter 3. The performance of the implementation is tested in Chapter 4, by two extraction experiments. In chapter 5 conclusions derived from the performance results are presented, as well as suggestions for future work and possible improvements in the current version of the implementation.

# 2 Background

This chapter gives a brief introduction to the field of Information Extraction and Web Document Analysis. The Document Analysis methods that are relevant to this thesis are described more in detail.

## 2.1 Information Extraction

Information extraction (IE) refers to the activity of automatically extracting information from a set of documents. Kushmerick (1997) uses the term "wrapper" to refer to the procedure of extracting information from a particular resource. The wrapper takes as input a request, identifies the information of interest in a set of documents, and maps the data to a suitable output format (Leander, Ribeiro-Nato, da Silva and Texeira 2002).

There are two general approaches to the development of IE systems. In the first approach, the "Knowledge Engineering" (Appelt and Israel 1999) or "Hand-coded" (Kushmerick 1997) approach, the wrapper is developed through an iterative process. Rules are written by an expert in the domain of the application (knowledge engineer), which are then tested against a training corpus, and modified appropriately. The second approach, "Automatic Training Approach" (Appelt and Israel 1999) or "Automatic Inductive Wrapper" (Kushmerick 1997), requires a training corpus where the information that is to be extracted is already adequately annotated. Then a training, or learning, algorithm is used to generate extraction information, that is, a hypothesis that generalizes the training examples, which a system can use when analyzing unknown texts (Appelt and Israel 1999).

The main advantage of the first approach is that good performance is not conceptually hard to develop, if good domain knowledge is available. Its disadvantage is that it is time consuming (Appelt and Israel 1999). The main advantage of the second approach is that domain portability is relatively straightforward, making it more flexible compared to the first approach. It is also less time consuming to develop, if relevant training corpora are available. The disadvantage is that it can require a large amount of training data, and if the annotated corpus is not available it can be expensive to obtain (Appelt and Israel 1999).

Leander et al. (2002) describe a variety of techniques used to generate wrappers for IE on the Web.

**Languages for Wrapper Development** are programming languages especially designed for assisting users in wrapper generation.

**HTML-aware Wrappers** are wrappers that rely on inherent structural features of HTML documents for extracting data. The features are derived by parsing the HTML into a tree, and by applying extraction rules onto the parse tree, which is the approach taken by this thesis.

**Modeling-based Wrappers** are wrappers that map portions of data in a document to a target structure of interest. The target structure is based on a set of modeling primitives describing the underlying data model of interest.

**Wrapper Induction Wrappers** learn extraction rules from training examples. The training examples are based on formatting features that implicitly define the data of interest.

**NLP-based Wrappers** identify information by rules that apply syntactic and semantic constraints onto text segments, thereby relying on techniques such as filtering, part-of-speech tagging, and lexical semantic tagging.

**Ontology-based Wrappers** rely on a domain specific ontology that locates data, from which "extraction objects" are constructed.

The most challenging aspect of IE is, according to Leander et al. (2002), the task of identifying information of interest within a potential large amount of irrelevant information. The complexity of identifying relevant information depends greatly on the information structure in a given document. Depending on what techniques a wrapper relies on different types of structures are analyzed. In NLP-based wrappers text structure is analyzed. Wrappers based, for example, on induction or HTML-awareness derive delimiters from the visual presentation or layout structure of a document.

In the IE literature, e.g. Soderland (1999) and Kushmerick (1997), three different document layout structures are mentioned. Layout is considered "structured" if the information in the document is presented in a rigid structure, for example, a table. In which case extraction rules can rely explicitly on layout information. Document layout is referred to as "semi-structured" if there are elements of varying presentation within the same document, some of which must be ignored by extraction rules. Semi-structured documents are for example a newspaper, a newspaper article, a book that contains chapters and possibly sub-chapters. Extraction rules for semi-structured layout can rely partially on the presentation of information, for example, font and/color, but requires further data analysis in order to locate data that is to be extracted. If a document has little or no layout structure, for example, text within a paragraph, the data must be analyzed in order to identify relevant information by using, for instance, NLP techniques such as part-of-speech tagging and syntactic analysis. Hence, the decrease of layout in a document requires an increase of data analysis in order to locate data of interest prior to extraction (Soderland 1999).

## 2.2 Web Document Analysis

Web Document Analysis (WDA) is a rather young field of research. Gu, Chen, Ma and Chen (2002) compare the detection of data structure in a Web document to the reverse process of Web authoring. The detection process is based on the division and repeated subdivision of the content of a Web page into increasingly smaller parts, that is, document segmentation. Depending on the purpose of the analysis, the document segments are mapped or classified according to certain conceptualizations.

The content of a Web page is typically heterogeneous. Data is multi-modal since different types of data can be displayed in the same page, such as images, text, hypertext, interactive forms, as well as dynamically created data embedded in scripts. The content is typically multi-topic containing, for example, varying articles, contact information, ads. Data is in general semi-structured, that is, the logical structure of data has a hierarchical nature, since many logical segments contain sub-segments. For example, a "news article" contains the sub-segments "heading" followed by a "paragraph".

The heterogeneous nature of the content in Web documents is challenging for different Language Technology methods. Web Document Analysis can be used to segment the content of a Web page in order to detect homogeneous content. For example, Yu, Cai, Wen and Ma (2003) use WDA in order to assist the selection of query expansion terms, terms which are used to improve performance of Web Information Retrieval. The selection of expansion terms is affected by noise and multiple topics in a document. If the selected query terms are unrelated to the topic of the original query, retrieval performance may likely decrease. Yu et al. (2003) filter noisy content (navigation menus, ads) from a page, and detect segments of multiple topics based on visual cues derived through WDA, thereby increasing the quality of selecting query expansion terms.

A variety of WDA approaches have been suggested and implemented by researchers. But, regardless of the approach, the general notion of WDA can be compared to that of "traditional" Document Analysis.

### 2.2.1 Document Analysis

Document Analysis (DA) analyzes the structure of a digitalized document. The document is instantiated either in a markup-based format, such as SGML or HTML, or in a layout-based format, that is, an "image like" version of a document instantiated by scanning and Optical Character Recognition (OCR) or by a Page Definition Language (PDL) such as PostScript (Summers 1998).

In DA two types of "structure" are primarily studied. Structure based on the presentation of the document is referred to as *layout* or *geometrical* structure. The second document structure is based on the human-perceptible meaning of a document's content, which is referred to as *logical* structure (Tang, Cheriet, Liu, Said and Suen 1999). The key notion of DA is to discover layout structure and map it to a logical structure (Tang et al. 1999), and the precise relationship between layout and logical structure is not known prior to the analysis.

DA is generally performed in two stages, which may overlap (Tang et al. 1999). Firstly, the layout structure of a document is extracted by repeatedly dividing the content into smaller and smaller segments. These segments are referred to by Tang et al. (1999) as *objects*. Objects that cannot be divided further are *basic objects* (BO). Objects that contain other objects are *composite objects* (CO) (Tang et al. 1999). A layout object must be visually distinguishable from surrounding objects, and the content of the object must be a semantic component, e.g. it must have meaning as a separate piece within the content of the document. Hence, if the document were to be transformed into a different presentation style the object must still be identifiable as a segment, that is, separable from its surroundings (Summers 1998). Segmentation is based not on the *content* of the current, or surrounding objects, but on *non-content-based* cues based on visual, typographic, markup, linguistic, and/or contextual information (Summers 1998).

The segmentation methods are traditionally hierarchical, suitable for documents in a markup format. But Tang et al. (1999) also describe a non-hierarchical method, which Summers (1998) refers to as "flat segmentation", used to analyze images of documents. The hierarchical methods include top-down or bottom-up parsing, and an adaptive split-and-merge approach. In the adaptive split-and-merge approach heterogeneous regions are separated, and homogeneous regions are merged simultaneously as the document is parsed. But the hierarchical approaches are not effective enough for documents represented as an image with high geometric complexity, because they are too time consuming. Therefore Tang et al. (1999) present a non-hierarchical method based on "modified fractal signature". The approach maps an image of a document onto a gray-level surface, from which geometrical structure can be approximated (Tang et al. 1999). Dark areas on the gray-level surface represent information segments, and light

spaces information boundaries. The hierarchical approach to segmentation is of relevance to this thesis. Therefore, no further description of non-hierarchical segmentation will follow.

Secondly, the layout structure is mapped into a logical structural representation. In other words, the layout object is classified considering the logical relationship between objects in the domain of the document (Tang et al. 1999). Since logical structure consists of a hierarchy of logical objects, the ancestry in the hierarchy reflects containment among document components, object clusters. The categories of different logical objects in a document are application-dependent. For example, a BO can be a word, an image, and CO's can be an article, a list of items, a comic strip etc. (Tang et al. 1999). There are a variety of mapping methods based for example on, tree transformation, formatting knowledge, and description languages. Tree transformation rules are created with information on layout design, and on how humans read. If formatting knowledge is available for a specific document its logical structure can be deduced from it with promising results (Tang et al. 1999).

Doermann, Rivlin and Rosenfeld (1997) propose an additional level of structure, functional structure. Functional structure is at an intermediate level between the layout and the semantic structure of a document. A functional interpretation of data can, according to Doermann et al. (1997), greatly facilitate tasks associated with object classification and use, and reflects the efficiency with which the document transfers its information. Class-independent identification of an object by, for example, attributes such as position, reveals the objects function. The semantic identification of an object is class-dependent. For example, the functional object "header" may be defined as the semantic object "title" depending on the class domain relevant for the analysis (Doermann et al. 1997).

## 2.2.2 Web Document Structure

In "traditional" DA textual documents with "semi-structured" layout are studied, such as newspapers. In WDA Web pages are of interest, which are also considered to be "semi-structured". The main difference between a "traditional" document and a Web page, besides the addition of media types such as sound, video, and interactive elements, is the documents format. HTML is the most commonly used Web page format for publishing hypertext on the Web.

HTML is a simple markup language used for publishing hypertext on the Web. It consists of a small set of predefined tags, which describe the format, the structure, the semantics, and other attributes of data displayed in a Web page. HTML is typically used to reflect layout structure rather than content structure. Even though certain tags can reveal semantic structure. For example, the tag `<address>` is used to provide contact information for the author of a document. "Meaningful" tags are frequently used for presentational purposes, e.g. `<table>`, which makes it nearly impossible to derive logical structure based just on tags. There are specific fields where metadata, that is, data about data, can be included in the HTML structure. But metadata is used in general to define the topic, and related keywords of the page content. Metadata cannot be used to reflect the generic function of each element in a page.

A variety of methods have been proposed in order to analyze the structure of a Web document. The purpose of the analysis may vary, but the overall notion is to identify specific logical segments of information in a Web page. Certain researchers suggest methods that are independent of the HTML markup, relying on "Image Analysis" and "Pattern Detection" techniques. Gu et al. (2002) locate structural boundaries based on "visual projection" of data. Yang and Zhang (2001) use "Suffix Trees" to detect frequent and similar visual patterns among objects in a page. A majority of approaches rely heavily on HTML markup

to derive logical structure, which is also the approach taken by this thesis. Methods based on HTML structure are presented more in detail below.

## 2.3 Document Segmentation

The general idea of DA methods that explore HTML markup is to detect boundaries between elements that can be used to delimit logical structure. HTML start-and-end tags are often nested within each other, see the HTML code in figure 2.1<sup>1</sup>.

```
...
<td class="bgCatStart">
<a href="/omsr/" title="Om Sveriges Radio"></a>
<div class="cat">
<b class="textBlue">SENASTE NYTT OM SR</b><br>
<table border="0" cellspacing="0" cellpadding="0" width=""> <tr><td
class="nsFix"> <a href="/omsr/press/press2004/0210.stm" class=""
title="Grammis till P3s Kaj Kindvall">Grammis till P3s Kaj Kindvall</a><br>
</td></tr>
<tr><td class="nsFix">
<a href="/omsr/tabla.stm" class="" title="Programtablå för 2004">Programtablå
för 2004</a><br>
</td></tr>
</table><br>
<b class="textBlue">VÄLKOMMEN TILL SR</b><br> <a href="/omsr/om.foretaget/"
title="Företaget">Om företaget</a><br>
<a href="/omsr/om.public.service/" title="Public Service-uppdraget">Public
service-uppdraget</a><br>
<a href="/omsr/publik/" title="Publik & Lyssnande">Publik & lyssnande</a><br>
<a href="/digitalradion/" title="Digitalradion">Digitalradion</a><br>
...
</td>
...
```

Figure 2.1: A Section of an HTML Document.

The Document Object Model (DOM) provides a standardised set of objects that define the structure of HTML and XML documents (Robie 1998). The DOM enables programmers to create, navigate, and alter the structure of a document regardless of the programming language. The logical structure of a DOM has a hierarchical nature, it resembles a tree. But, as Robie (1998) states, the DOM does not require documents to be implemented as a tree. The DOM representation of part of the section of HTML code in figure 2.1 is given in figure 2.2.

In a DOM tree most tags are represented as non-terminal nodes, see `<a>` and `<td>` in figure 2.2, "empty tags" tags are represented as terminal nodes, see `<br>` and `<img>` in figure 2.2. Empty tags, as well as text segments are the leaf nodes of the tree. Chen, Zhou, Shi, Zhang and Wu (2001) define a BO (basic object or leaf node) in WDA as a non-breakable HTML element, an object that does not contain any embedded tags tags. In figure 2.2 there are three leaf nodes, `<img>`, `Senaste Nytt om SR`, and `<br>`.

But the content of a leaf node can be quite fragmental. Depending on the purpose of the analysis, identifying larger logical segments may be required, that is, composite objects. COs can be a whole subtree or

<sup>1</sup>The code is from the page <http://http://www.sr.se/> (February 2004)

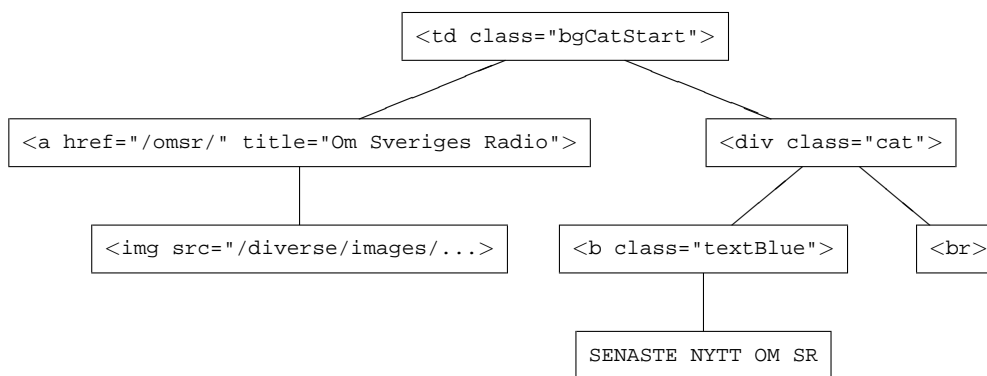


Figure 2.2: Example of a DOM Tree from Part of the HTML Code in Figure 2.1.

parts of subtrees, and identifying the boundaries of a CO in the parse tree is therefore not as straightforward as locating leaf nodes.

There are a variety of methods that attempt to locate the beginning and end of logical segments in a Web page. Certain approaches rely heavily on the HTML syntax, see section 2.3.1. These approaches are knowledge driven, and anticipate that the HTML syntax segments the page into relevant content blocks. The tree is traversed top-down until the desired segment is identified. Most of these algorithms use a combination of specific HTML tags and attributes to detect boundaries between logical segments. Other methods, see section 2.3.2, locate boundaries by first identifying visual separators, which are derived from the syntax of the tree. Approaches that rely less on the HTML syntax and more on the actual content of the objects can be considered as data driven methods, see sections 2.3.3 and 2.3.4. In "data-driven" methods the tree is traversed bottom-up, objects are classified, anticipating that categorization of objects can reveal contextual dependencies and structure.

### 2.3.1 Structural Tags

Certain proposed segmentation approaches locate information by identifying specific HTML tags. Fong, Hui and Vu (2002) extract publication information from certain types of Web pages. In their approach they regard certain tags as "item" tags (BO tags), and others are seen as "block" tags (CO tags), which in turn are used to structure basic objects. Fong et al. (2002) differ between structured and unstructured block segments. In unstructured blocks items (BOs) and blocks (COs) can have the same tags, for example, nested paragraph tags `<p>` in a paragraph. In structured blocks, the block tag and item tag differ, reflecting a logical structure. The block "definition list" is defined with the tag `<dt>`, and each list item with a `<li>` tag.

Embley, Tao and Liddle (1999b) present a method which extracts tabular information from car ads. They identify "tables" by the `<table>` tag, and treat its content as if it has a logical structure, which, as they acknowledge, is not always the case. In many pages `<table>` is used for layout purposes only. In order to distinguish between the ambiguous use of `<table>`, Penn, Hu, Luo and McDonald (2001) propose heuristics for identifying genuinely tabular information in a Web page. Their rules require that a genuine table contains multiple rows, and that the content of the cells are "leaf nodes" containing text with a rather short length.

Methods that just rely on the hierarchical structure of HTML and specific "semantically rich" tags can run

into problems. For example, Gupta, Kaiser, Neistadt and Grimm (2003) present a method which extracts the main text content of a page. In their approach, chunks of information such as ads and "lists of links" are directly filtered from the tag tree. Their approach lacks further structural analysis of the layout, which causes trouble when attempting to filter "list of links" or menus from link rich pages, because the syntax of the document alone is not always a good enough indicator for where a "list of links" starts, and where it ends.

Lin and Ho (2002) also propose a method for extracting content from Web pages. They, on the other hand, focus only on analyzing pages that rely just on `<table>` for the page layout, which according to their research 70% of the pages do. This approach reduces the complexity of the parse tree, and increases the chance that tabular tags act as delimiters between content rich composite objects (COs). In other words, they treat `<table>` as a layout block tag, where the content of the block is a logical segment of the page, but doesn't require that the table contains coherent information. In case a table contains a lot of textual information the table is broken down into several sub-blocks.

### 2.3.2 Visual Separators

A number of researchers attempt to locate the beginning and end of a block of information through visual cues. For example, the "Vision-based Page Segmentation Algorithm" (VISP) described by Yu et al. (2003) extracts "Visual Blocks" from a DOM tree. The VISP algorithm parses the HTML into a DOM tree. But Yu et al. (2003) find that certain nodes in the DOM tree need further segmentation. The additional segmentation in the VISP algorithm is based on certain cues within the content of a node. Yu et al. (2003) suggest that tags such as `<hr>`, which is used to create a horizontal line, as well as attributes that indicate a change in background color can indicate the beginning or end of a segment. These segmentation cues are seen as "visual separators" and are classified by Yu et al. (2003) as horizontal or vertical lines. After the VISP algorithm has parsed the HTML, visual separators are detected in the parse tree. The separators receive weights which are adjusted depending on constraints based on separator dimension, adjacent tags such as `<hr>` (a line), and surrounding font sizes. The separators divide subtrees further, depending on if they cross a "visual" block of information or not. Finally, the content structure of the page is created, by merging "visual" blocks that are not divided by separators.

Embley, Jiang and Ng (1999a) describe a set of heuristics in order to discover content record boundaries. They first locate subtrees in the tag tree where boundary markers could appear. They consider separators to appear most likely in subtrees with high fan-outs, that is, a subtree with many daughters. Then the amount of times each tag appears in the subtree is calculated. Tags that appear frequently are regarded as probable record separators. Unless just one candidate tag is found in the tag tree, five heuristics are applied to rank candidates in order to determine which candidate tag is most likely a separator. The heuristics are based on:

**Frequency** values of a tag in the page, assuming that separators are likely to appear repeatedly; they use a "predefined" list of likely separator tags.

**Standard deviation** values for tags based on the amount of text between each candidate tag, assuming that the amount of text between separators is larger than between non separators.

**Repeating-tag patterns** assuming that adjacent tags reappear consistently with a separator.

**Ontology matching** is used to locate record-identifying fields, from which the number of records in the page is estimated, which in turn is used to rank the candidate separators based on their similarity in frequency.

Each heuristic returns one or more candidate separator tags with a measure of certainty/uncertainty attached to each candidate. Finally, the individual heuristics are combined to define which separator tag is considered as the record boundary indicator of the page.

### 2.3.3 Coherent Components

Mukherjee, Yang and Ramakrishnan (2003) transform the DOM tree of a HTML document into a "semantic partition" tree, by first traversing the DOM tree top-down until the leaf nodes are reached, and then by restructuring the tree bottom-up according to certain heuristics. Smaller logical segments are aggregated into larger logical segments, and larger logical segments are verified and potentially divided further. This segmentation approach is similar to the segmentation proposed by Cai, Yu, Wen and Ma (2003), a split-and-merge approach. The difference in the two approaches is that Mukherjee et al. (2003) do not integrate boundary cues in their structural analysis, as is suggested by Cai et al. (2003). They segment the HTML structure with visual and lexical consistency cues among siblings.

To verify the structure of a node that contains multiple daughters, Mukherjee et al. (2003) consider "lexical association" and "maximal repeating substrings" among daughter nodes. The former refers to the occurrence of a word in several daughters, indicating semantic coherence between daughters. The latter refers to, that semantically related items exhibit consistency in presentation style and spatial locality. If daughters reveal lexical or visual coherence, the daughter nodes are either split, or merged with high confidence (Mukherjee et al. 2003).

Chen et al. (2001) find, similarly to Mukherjee et al. (2003), that semantically related segments have a consistent visual style. Coherent segments are separated from objects in other categories by visual boundaries. Chen et al. (2001) detect segment boundaries by clustering neighboring objects based on a visual similarity pattern detection algorithm. The algorithm measures the visual distance between objects. Clustering relationships such as complementary, parallel, or tabular based on visual density are applied to segments in the DOM tree in order to determine the visual dependencies among siblings, which indicate where visual boundaries appear.

### 2.3.4 Contextual Cues

Document segmentation based on visual presentation, and HTML parse tree structure, may reveal logical segments at a fine level of granularity (BOs), rather than at a coarser level (COs). The boundaries of a composite object can be found by combining visual information with contextual information. But the context of objects can only be derived if the logical structure is known for each object. As stated above, certain HTML tags can reveal the logical structure of a segment, e.g. <a> is used to tag hypertext (links). But the content of a link can have many different functions, e.g. it can be a heading, a menu item, a picture, it can point to a section within the same page, within the same web site, or even point to a different site. Therefore, in order to "understand" the meaning of an object and consequently its context, each object must first be classified according to a certain domain knowledge.

According to Summers (1998) contextual cues are either *local* or *global*. Local contextual constraints use information from a limited amount of neighboring nodes. Global contextual cues use information from a document as a whole.

## 2.4 Object Classification

The analysis of layout and logical structure may overlap (Tang et al. 1999), a tactic which can increase the accuracy of segmentation since logical structure is derived more or less simultaneously. This "intertwined" approach is used by, for example, Summers (1998), Embley et al. (1999b), and Chen et al. (2001). The semantic structure of an object can be identified and categorized in a variety of ways. The classification method applied in the implementation presented in this thesis is based on a domain ontology.

### 2.4.1 Ontology-Based Classification

Mukherjee et al. (2003) propose a method which automatically annotates logical segments in a Web page. They derived semantic labels for certain nodes directly from the HTML markup, though which are not stated by Mukherjee et al. (2003). Assumably certain "semantic" tags such as <h1>-<h6> used to tags "headings". When annotations cannot be derived from the HTML syntax, i.e. when BOs are aggregated into a CO, Mukherjee et al. (2003) employ ontology encoded domain knowledge to infer concept associations that are used to classify the content of a segment. For example, an object is identified and classified as a "major news headline item" by a rule in the ontology based on the key features associated with the item such as the HTML tag <a> (link), certain news keywords (Reuters, TT), and features such as constraints on the length of text a heading can have (Mukherjee et al. 2003).

Embley et al. (1999b) also depend on an ontology in order to identify and classify objects. They extract tabular structured car ads from Web pages. Embley et al. (1999b) define an ontology as a domain specific conceptual-model. Their ontology consists of two components. Firstly, an object/relationship model that describes the models sets of objects, sets of relationships among objects, and constraints over objects, and relationship sets that can exist in the conceptual model (line 1-8 in figure 2.3). Secondly, data frames for the different object sets that define the potential contents of the object set (line 9-17 in figure 2.3 ).

```
1. Car [-> object];
2. Car [0:1] has Year [1:*];
3. Car [0:1] has Make [1:*];
4. Car [0:1] has Model [1:*];
5. Car [0:1] has Mileage [1:*];
6. Car [0:*] has Feature [1:*];
7. Car [0:1] has Price [1:*];
8. PhoneNr [1:*] is for Car [0:1];
9. Year matches [4]
11.   constant {   extract "\d{2}'';
12.               context "\b'[4-9]\d\ b'';
13.               substitute "^^'-> "19'';};
14.   ...
15. Mileage matches [8]
16.   ...
17.   keyword "\bmiles'', "\bmi\.'', "\bmileage'',
...;
18. ...
```

Figure 2.3: Part of the Car-Ad Extraction Ontology from Embley et al. (1999a), page 8.

## 2.5 Areas of Application

There are a variety of approaches, and areas of use for Document Analysis. Depending on the purpose with the analysis documents are parsed based on their visual layout, markup, and/or text structure.

Embley et al. (1999a) parse Web documents based on the HTML tag structure in order to identify tabular data, that is, visually rigid structured document components, from which information is mapped to an extraction ontology in order to extract information on car adds.

Yu et al. (2003) use knowledge on document structure to enhance the quality of query expansion terms in Information Retrieval. Their VISP document segmentation algorithm detects segment boundaries based on visual cues. The hierarchical segmentation groups semantically related content into a single segment. The estimation of term correlations, used to detect query expansions, is therefore improved, by focusing on correlations only within the same content segments and not within the whole page. Yu et al. (2003) improve the performance of the pseudo-relevance feedback from a baseline of ca. 16% to ca. 20% by applying document analysis.

Boguraev and Neff (2000) suggest that knowledge on the logical structure of documents could enhance the performance of automatic summarization systems. Boguraev and Neff (2000) use a document structure representation that provides means of identifying content and layout "metadata". The document structure is derived by parsing document based on the markup tag structure providing information on document segments such as section, sub-section, heading, paragraphs, tables, figures, captions, sidebars and information extraneous to the main document narrative. This structural information is used to increase the values of silent items (i.e. data that appears more than once in the same document which are seen as summarization candidate items) depending on the segments position. They find that their segmentation-based summarization has a higher accuracy in short summarizations than in longer summarizations compared to their base summarization approach.

Resnik and Smith (2003) propose an algorithm which attempts to use the Web as a Parallel Corpora, and state an observation, which implies that authors tend to use the same document structure when presenting the same content in two different languages. They test a structural alignment algorithm that matches the document structure in one page to the other based on the HTML markup. In their concluding remarks Resnik and Smith (2003) state that the alignment of markup based document structure "produces reasonable first-pass alignments of the inverting text".

Table interpretation, query expansion, and summarization, as described above, are areas where document analysis is applied. What they all have in common is that they try to detect generic document components. Embley et al. (1999a) detect "tables", Yu et al. (2003) "content sections", and Boguraev and Neff (2000) locate a variety of document components.

# 3 Implementation

The purpose of this thesis, as defined in the Introduction, is to develop a program that extracts generic document components from a Web page, that is, a wrapper. The wrapper consists of four major components: Preprocessing, Document Segmentation, Object Classification, and Extraction. In the following sections the general approach to the document analysis is described first, followed by a more detailed description of the different components of the implementation.

## 3.1 The General Approach

The wrapper generation approach taken by this thesis is based on "knowledge engineering", that is, the extraction rules are constructed manually with domain specific knowledge and not learned automatically, as suggested by for example Kushmerick (1997). Document analysis methods, as described in section 2.2.2, are used to compute the layout structure, and thereof the semantic structure of a document.

The document analysis presented here uses the hierarchical nature of HTML in order to create a parse tree of the content of a Web page. The parse tree represents the layout structure of the document. The aim of the document analysis is to convert the layout structure to a logical structure, representing the generic document structure. The parse tree is converted by traversing the tree in a top-down, bottom-up, and split-and-merge fashion, locating smaller logical segments, which in turn are aggregated into larger logical segments. The parse tree is segmented based on visual boundaries such as images, and document segment markers such as "headings". The document segments are classified based on certain HTML tags and attributes, typographic cues, and contextual constraints, described in an ontology for "generic Web document components".

The document analysis, proposed by this thesis, tries to identify the semantic structure of an object *regardless* of the documents topic. The identification is therefore considered "class-independent". From that I conclude that the functional level of an object proposed by Doermann et al. (1997), see section 2.2.1, is in this thesis equivalent to the semantic structure of an object. But depending on the aim of the document analysis, "class-dependent" information might be necessary in order to identify the semantic role/topic of an object. In that case I agree with Doermann et al.'s (1997) approach, by first identifying the function of an object, and then its class-dependent semantic meaning. In such cases a general, class-independent document analysis software module could be reused for a variety of domains, and a class-dependent module could be built directly on the general module, presuming that the general analysis is relevant to the class-dependent analysis. The logical segments of interest to this thesis

are, for example, a "heading", a "paragraph", a "navigation menu", a "picture" etc. I refer to these elements as "functional objects" or "generic objects", following the reasoning of (Doermann et al. 1997) on class-dependent vs. class-independent structural level of an object.

## 3.2 Related Work

The semantic structure derived by the present implementation is similar to that proposed by Chen et al. (2001). But differences can be found in both the segmentation and the semantic mapping. The semantic mapping in Chen et al. (2001) creates function-based object models from a HTML parse tree, for Web page adaptation. The semantic mapping in this thesis transforms the HTML parse tree into a tree with semantic objects with relevant annotations, relevant for object extraction. The segmentation in Chen et al. (2001) relies on density clusters<sup>1</sup> based on visual similarity patterns, in order to find the boundaries of a CO. The segmentation proposed here focus on objects that function as boundary markers, similarly to Yu et al. (2003). The detection of visual boundaries differs from Yu et al. (2003), in that a "primitive" form of image analysis is performed in this implementation in order to detect potential image boundary markers. Also the positions of "heading" objects are considered as potential boundary markers, depending on their context, not just headings tagged with `<h1>`-`<h6>`, but all potential headings are considered. Contextual cues are also regarded, similarly to Summers (1998), as well as coherence among siblings, as suggested by Mukherjee et al. (2003), both in the segmentation and classification of objects.

## 3.3 Training data

The misuse of HTML markup is widely spread. Browsers compensate for bad code, so poor code can be visualized correctly. But when analyzing Web page structure, inconsistent use of markup can generate problems. Lin and Ho (2002) reduce the complexity of their analysis by focusing on pages with a presentation based on tabular layout. The approach taken by this thesis, to reduce parse tree complexity, is to insure the use of high quality code. There is no safe way to declare whether or not a page is coded well. But, if a pages creator defines a "document type definition" (DTD) in the beginning of the code, the creator assures that the code is constructed following a specific W3C standardized markup syntax. W3C (World Wide Web Consortium) develops standard technologies for the Web. Example of a DTD:

```
< !DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd" >
```

Two pages from 10 different web sites were included in the training corpus. I chose mainly Swedish governmental sites, both for training and testing, because, according to the Swedish decree 2001:526, governmental authorities must provide information equally accessible to all, they must strive especially to adapt information to the functionally impaired. I hoped that this decree would increase the code quality of public Swedish sites, since functionally impaired often rely on alternative browsers and keyboard navigation when accessing the web, which requires high quality code. Three sites where included in the training set although they did not state a DTD standard. The only non-Swedish site chosen was `http://www.w3c.org`, since they create the standards I assumed that their code would be exemplary for this task.

---

<sup>1</sup>A density cluster is defined as a maximal set of density-connected points

W3C provides a validation service, an SGML parser, which can be used to validate HTML or XHTML documents according to W3C standards and recommendations. Out of curiosity I tested the front pages of all the sites that were included in the training set, which had declared a DTD (7 sites). Just two out of seven front pages contained 100% valid code, one of which was W3C's own front page. This shows that even though Web page creators attempt to follow W3C standards few actually manage to.

## 3.4 Preprocessing

The preprocessing component of the implementation fetches and prepares the document of interest for the document analysis.

### 3.4.1 Retrieving the Web Page

The first task of the implementation is to retrieve the desired page from the Web. The user of the application gives as input to the system the URL (Uniform Resource Locator) of the page to be analyzed. Then, by using the Perl module LWP, a HTTP (Hypertext Transfer Protocol) request is created and the document the URL points to is fetched from the web. A URL is a type of URI (Uniform Resource Identifier), for example, `http://www.dn.se/`.

A URI provides a way for identifying an object on the Web, using a specific syntax for encoding the components of an identifier. The primary access mechanism of a URL is "http", so a http URI is a URL (Burke 2002). HTTP is a server/client protocol used to fetch most documents on the Web. The client, for example, a Web browser or the current implementation, uses the URL of a document to identify and contact the server that holds the document, who in turn returns the document to the client (Burke 2002).

### 3.4.2 Parsing the HTML

In order to identify the smallest logical segments of a page, the HTML markup is parsed into a tag tree similarly to Embley et al. (1999b), Yu et al. (2003), and Mukherjee et al. (2003). The tag tree is created by the HTML::TreeBuilder Perl module. The HTML::TreeBuilder and its sister modules do not try to attempt to parse the HTML code by strictly following one of the W3C HTML standards. They attempt to parse the code as closely as possible to the way a Web Browser would (Aas 2003), thereby compensating for inconsistent code by, for example, "closing" open tags.

The parse tree generated by HTML::TreeBuilder consists of two types of nodes, HTML::Elements and text nodes. In figure 3.1, B) represents an HTML::Element of an HTML::TreeBuilder parse tree for the HTML code in A). An HTML::Element stores all attributes that a node can have, for example, "href" in line 2, and "class" in line 3 of B) in figure 3.1. The HTML::Element keeps track of the elements ancestry, line 4, and the elements content, line 7. The text nodes, that is, a string of text that contains no embedded tags, are the leaf nodes of the HTML::TreeBuilder parse tree. The leaf node in B) is found in line 7. An HTML::Element can also appear as a leaf node. In which case they are "empty elements", that is, elements that lack content. They lack content either because they are used for layout purposes, or because they are elements that never contain textual content, such as breaks `<br>` or images `<img>`.

A)

```
<a href="/DNet/jsp/polopoly.jsp?d=678&a=244670&previousRenderType=6"
class="rubriklank" target="_top">Industriavtal
&ouml;verl&auml;mnat till parterna</a>
```

B)

```
1.  bless( {
2.      'href' => '/DNet/jsp/polopoly.jsp?d=678&a=244670&previousRenderType=6',
3.      'class' => 'rubriklank',
4.      '_parent' => $VAR1->{'_content'}[0]['_content'}[3],
5.      'target' => '_top',
6.      '_tag' => 'a',
7.      '_content' => ['Industriavtal överlämnat till parterna']
8.  }, 'HTML::Element' ),
```

Figure 3.1: Example of an HTML::Element, in B), for the HTML Code in A).

### 3.4.3 Extracting Frame Content

Before the analysis can continue any further, the parse tree has to be checked for framesets. Otherwise, potential information may be lost since the content of a frameset is stored elsewhere. Framesets are used to control the layout of the Web page, by creating rectangular spaces. A frame within the frameset contains a URL to the source document, the content, of the frame. In order to retrieve the content of a frameset, all URL's of frames in the frameset are gathered. The URL's source documents are then retrieved as described in section 3.4.1. The HTML code of the documents is parsed, see section 3.4.2, and all the parse trees of the framesets in the document are gathered, in extraction order, and together they then represent the content of the page to be analysed.

### 3.4.4 Detecting Visual Boundaries

The final step of the preprocessing searches for potential visual boundaries in the documents parse tree, and stores them for later use in the segmentation of the document (see section 3.5). Visual boundaries can indicate the boundaries of a segment of data.

Certain boundary positions can be derived explicitly from the elements of the parse tree, for example, the tags `<hr>` (horizontal rule/line) and `<br>` (break), which are often used as boundary indicators. Relying just on those two boundary tags as visual boundary indicators is not efficient enough for the segmentation. The `<hr>` tag is used very rarely. The use of `<br>` is much more frequent, but differs greatly between pages and also within pages. The meaning of a single `<br>` is generally that of "new line". But a "new line" can appear in very different contexts. It can be used to break lines in a paragraph, or between a heading and the following sentence/paragraph. In the first case the break does not indicate a segment boundary, but it could in the second case. Therefore, `<br>` is by this implementation not regarded as a visual boundary indicator. But the implementation does use `<br>` positions as contextual information in the classification of objects.

The primary visual boundary cues, used by this implementation, are derived from images and certain empty nodes. Images are analyzed in a "primitive" way in order to determine if they are a visual boundary

marker, or not. The analysis classifies the image based on its format, how frequent its source is used in the page, the presents of "alt-text", and the images dimensions. The dimensions are derived from its "width" and "height" attributes, and are used to determine the images "linearity" and "size". For example, an image is not considered as a boundary indicator if it square, it must be linear; neither are images with a JPEG<sup>2</sup> format.



Figure 3.2: Example of a Visual Boundary Generated by an Image.

Figure 3.2 displays a sequence of data from <http://www.sr.se> (February 2004). There are two lines in the data sequence, a horizontal and a vertical dotted line, which both function as boundaries. The vertical dotted line is not recognized by this implementation, because it is part of the image "Senaste Ekot", to its immediate right. The horizontal line, coded as ``, is recognized as a boundary because on a scale between 0 to 1, where 0 represents a square image, the horizontal line has a linearity of 0.95.

Empty elements can also be considered boundary indicators if they are an empty "basic node", not empty "block nodes" (see section 3.5). For example, an empty `<option>` tag in a "select form" may indicate a topic shift between select values, see the "select form" from "Svergies Radio" (<http://www.sr.se>, February 2004) in figure 3.3. The select options are divided into two segments by an "empty select option". The first segment holds information on "radio channels", and the second segment contains select options on "company information".



Figure 3.3: Example of a Boundary Generated by an "Empty Element" in a "Select Form".

<sup>2</sup>JPEG is an image format. It is used to compress digital, "photographic" images, most often in color. GIF is an image format used to compress grayscale, "grafic like" images.

### 3.4.5 Preprocessing Algorithm

1. The user of the application inputs the URL of the page to be analyzed.
2. Then, using the Perl module LWP::UserAgent, a HTTP request is created and the body of the URL is fetched from the web.
3. Create parse tree, with the Perl module HTML::TreeBuilder.
4. Check tree for frames.
  - 4.1. If the parse tree contains a frameset the URL for each frame in a frameset is extracted.
  - 4.2. The URL is checked to see if it is an absolute URL, <http://www.w3c.org/>, or a relative URL, `.../TR/html4/`.
    - 4.2.1 If the URL in a frame is relative it has to be converted to an absolute URL, this is implemented by use of the model "URI".
  - 4.3. Then the HTML code for each frame is retrieved from its host server, and a pseudo HTML file is created with the content from each frameset, and each frame in a frameset.
  - 4.5. The new parse tree is created from the pseudo HTML file, and the original parse tree is discharged.
5. Check presence of boundary markers
  - 5.1. Traverse the HTML parse tree and extract all "empty" elements. Certain HTML elements are ignored such as empty "block" nodes and "new lines".
  - 5.2. If the "empty" element E has a width and height attribute, then X is the shortest value and Y the longest value
    - 5.2.1 calculate the linearity L for E.

$$L(E) = 1 - \frac{X}{Y}$$

The linearity value,  $L(E)$ , is a positive number between 0 and 1, where 0 indicates that an element is non-linear (square).

- 5.3 classify the element E
  - 5.3.1. If X for E is less than 5 pixels the potential boundary is not classified as a boundary, because visually it is hardly perceived as a boundary.
  - 5.3.2. Elements with a linearity value above a certain threshold are classified as boundary markers, and receive therefore a high boundary probability weight.
  - 5.3.3. Else, if a node lacks dimensional attributes it is still classified as a boundary but receives a lower boundary probability weight.

## 3.5 Segmenting the Document

The HTML parse tree is traversed top-down, depth-first. Each daughter of a node in a subtree is either considered to be a "block node", or a "basic node".

A block node is usually not a leaf node, because they can contain an unlimited amount of daughters. Tags that are considered as block nodes are, for example, `<table>`, `<tr>`, `<td>`, `<form>`, `<div>`, `<p>`, and `<ul>`. Tags seen as basic nodes are, for example, `<a>`, `<span>`, `<img>`, `<h1>`-`<h6>`. The block node is traversed further until a basic node is reached.

A basic node is considered as the leaf node of the tree, e.g. a basic object (BO). Chen et al. (2001) define a BO as a node with no embedded tags. The definition of a BO in this thesis differs slightly from that proposed by Chen et al. (2001) (s. 509). Here, a leaf node may contain embedded elements, such as a "bold" text segment in figure 3.4 below. The tag `<b>` (bold) is seen as an attribute to the text, and not as the text's main function. But in some cases leaf nodes may contain multiple daughters, that is, if the daughters are perceived as a single coherent unit such as an "image text link" in figure 3.5 below. Hence, in this implementation a whole subtree can be recognized as a BO. In some cases the basic node `<span>` appears as a block node, e.g. if it contains an embedded `<span>` element as in figure 3.6 below.

Once a leaf node is located it is classified and the node receives a "semantic representation", see section 3.6.1. When all the daughters of a "block node" are annotated, they are aggregated iteratively into a larger logical segment, that is, a composite object (CO). But before the CO is classified the daughters are searched for boundary markers. The presence of a boundary marker indicates that the granularity of a block node is too coarse, and that the daughters must be segmented further. The delimiters of a new segment are identified by the position of a boundary marker, see section 3.5.1.

If no boundary markers are present amongst the daughters of a block node, the content is considered cohesive, and the daughters are aggregated into a single CO. The CO is then classified, see section 3.6.2. If the content of a "block node" is a single object, CO or BO, the parse tree is pruned and the object is moved up one level of the parse tree, i.e. the granularity of the tree was considered too fine.

In this iterative manner the parse tree is traversed recursively top-down, depth-first, in order to locate and classify BOs. Then segmentation (aggregating, pruning and merging), and classification of COs is performed iteratively, bottom-up until the root node of the tree is reached. The HTML parse tree is thereby transformed from a tree containing HTML::Elements, into a tree of semantically annotated, and semantically structured objects that can be searched, analyzed, and indexed depending on the purpose of the analysis.

### 3.5.1 Boundary Markers

There are two types of boundary markers used in the segmentation, visual separators (VS) such as images see section 3.4.4, and headings (H). If a segment of data contains both types of boundary markers, constraints are applied to the potential delimiters in order to determine which of the markers have the highest precedence. A rough description of the precedence rules is given below.

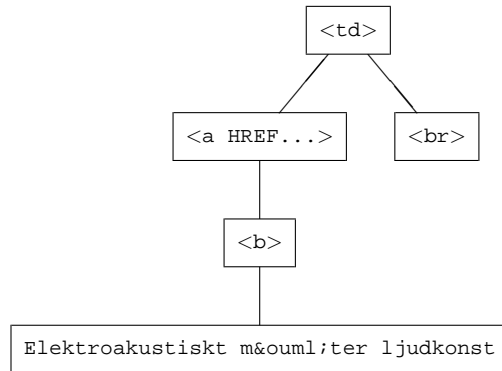


Figure 3.4: Example of a BO with an Embedded Element.

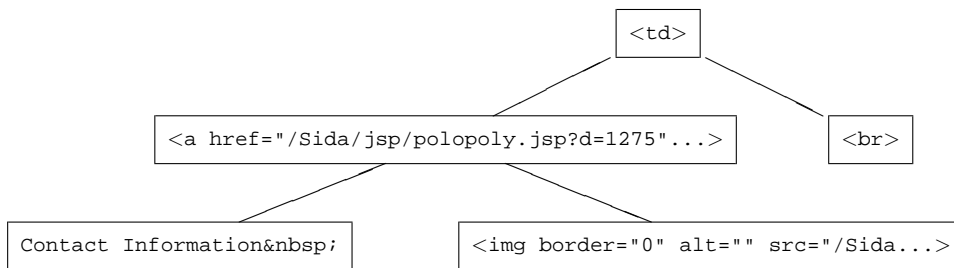


Figure 3.5: Example of a BO with Multiple Embedded Elements.

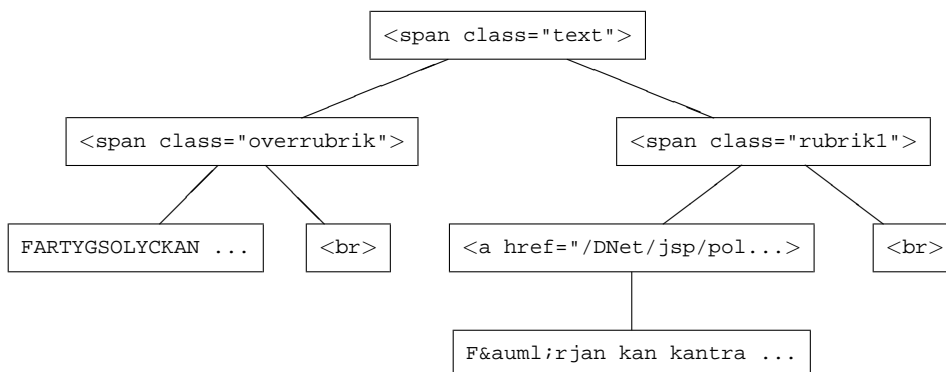


Figure 3.6: Example of a BO that really is a CO, it has Embedded BOs.

1. If two or more headings are visually similar, then they have the highest precedence as segment delimiters, regardless of the amount of visual boundary markers amongst the siblings. Visual similarity is determined by identical class attributes, heading tags, or typographic text tags such as "bold".
2. If there are more "Visual Separators" (VS), such as empty elements or image boundaries, then headings, or exactly the same amount, then the positions of the VS are regarded as the most likely delimiters.
3. Else, both heading and VS positions are used to determine segment boundaries.

For example, the sequence of data in figure 3.2 can be represented by the of objects in example A) below. H1 is the heading "Ekots fem senaste" (the five most recent news from Ekot). CO1 is a composite object, a table with two columns (time and news), and five rows containing information on the latest news and the time were posted. VS1 is a visual seperator, the horisontal dotted line. CO2 and CO3 are composite objects that hold the two segments of data below the dotted line. The segment has two boundary delimiters, H1 and VS1. Following the precedence rule 2, VS1 is defined as the boundary marker. The result of the segmentation is given in example B) below.

A) [H1 CO1 VS1 CO2 CO3]  
 B) [[H1 CO1][CO2 CO3]]

### 3.5.2 Segmentation Algorithm

1. The current node N is a block node; gather all the daughters of N (Start at the root node (the root node has to have daughters, else the page is empty))
  - 1.1. Foreach daughter node DN of N
    - 1.1.1. If the node DN is a block node, continue with step 2.1.
    - 1.1.2. Else, the node DN is a basic node (BO), classify DN see section 3.6.1
2. When all daughters of N are classified, examine the daughters in their context, adjust type definitions (annotations) if required, and search the daughters for the presents of boundary markers.
  - 2.1. Adjust certain type definitions according to their context, see section 3.6.2
  - 2.2. For each new segment indicator found amongst the daughter nodes of N
    - 2.2.1. Expand the sub-tree of N with a new node N+1, the daughters of N in the split are pruned from N and are made the daughters of N+1, e.g. a new level is added to the sub-tree of N
    - 2.2.2. Adjust all the numbers of all the descendants of N+1 to the "expansion"
    - 2.2.3. Before the next split is performed, continue to 2. with N+1 and check if there are segment boundaries in N+1
  - 2.3. When no further splits can be made in N, the CO block node N-CO is created and classified.
    - 2.3.1. If N just contains one daughter N+1, CO or BO, N is not a true CO, discard N and move the daughter node N+1 up one level (the sub-tree is pruned/merged)
      - 2.3.1.1. Adjust all the numbers of descendants in N+1 to the merge
    - 2.3.2. Else, classify N-CO, see section 3.6.2

## 3.6 Classifying the Objects

The logical structure of an object can be determined to a certain extent from the HTML markup. Certain tags reflect the semantics of its content, such as `<h1>`, a tag that usually contains a heading of level 1. But tags that reflect semantic structure are often misused and can't be seen as a reliable source of information. But tags are a reliable source of information when determining the modality, the general category, of a BO. Tags reveal if a BO contains an image, a text segment, hypertext, an interactive form element etc. But tags say little of the precise function of a BO, e.g. if a link is a heading, an image, or a whole sentence. The function of a BO, and especially a CO, is generally determined by analyzing its content, and its attributes. In rare cases the layout of a COs content can reveal its general function.

The classification of BOs and COs is based on heuristic knowledge of which functional objects a Web page is composed of, how objects are identified, and what relationships exist between objects. This knowledge is conceptualized in an ontology, defined for this implementation. Figure 3.8 and figure 3.9 display sections of the ontology the implementation relies on.

The implementation maps the objects to the domain ontology, through decision trees and regular expressions. Once a node of a decision tree, or the final state of an automaton is reached, the object receives a semantic representation. The semantic representation contains an annotation revealing the "class type" of the object, e.g. "link text heading", and other class specific attributes, e.g. if the link is site internal or if its source is site external, see figure 3.7.

```
{
  'link_type' => 'site internal',
  'NR' => \'1.5.1.1.5.1',
  'class' => \'rubriklank',
  'href' => bless( do\my $o =
    'http://www.dn.se/DNet/jsp/polopoly.jsp?d=678&a=244670&previousRenderType=6' ),
    'URI::http' ),
  'type' => \'link text heading',
  'BO' => '<a class="rubriklank"
    href="/DNet/jsp/polopoly.jsp?d=678&a=244670&previousRenderType=6"
    target="_top">Industriavtal &ouml;verl&auml;mnat till parterna</a>'
}
```

Figure 3.7: Example of the Semantic Representation for the Link in Figure 3.1.

### 3.6.1 Basic Objects

Once a basic node is detected, the modality of the BO is determined first. If the BO is a segment of words, it is considered an object of type "text". If the BO starts with a tag, then the tag determines the BOs modality. There are 7 general modalities for BOs, "link", "image", "text", "form", "map area", "empty element", and "html element" (unknown objects). Most categories contain subclasses, e.g. a text segment can be classified as a "heading", or a "sentence". The subclass of a node is defined by analyzing the content of a node, and/or by the presents of certain attributes in the node. The content analysis that this implementation relies on is very shallow. It would be desirable to include more advanced analysis methods. But incorporating, for example, advanced NLP techniques in the classification of text segments is beyond the scope of this thesis.

```

1. Link [-> BasicObject];
2. Link [1:1] has-a Annotation [1:1];
3. Annotation [1:1] is-a LinkType [1:1];
4. Link [1:1] has-a Href [1:1];
5. Link [1:1] has-a LinkNavigation [1:1];
6. Link [0:1] has-a Class [1:1];
7. Link [0:1] has-a AltText [1:1];
8. LinkType [1:1] is-a Text [1:1];
9. LinkType [1:1] is-a IMG [1:1];
10. LinkTag [1:1] is-a Tag [1:1];
11. Text [-> BasicObject];
12. Text [1:1] has-a Annotation [1:1];
13. Text [0:1] has-a Class [1:1];
14. Text [0:1] has-a TextTag [1:1];
15. TextType [1:1] is-a Annotation [1:1];
16. TextHeading [1:1] is-a TextType [1:1];
17. TextSentence [1:1] is-a TextType [1:1];
18. IMG [-> BasicObject];
19. IMG [1:1] has IMGType [1:1];
20. IMG [0:1] has Linearity [1:1];
21. IMGType [1:1] is-a Photo [1:1];
22. IMGType [1:1] is-a Image [1:1];
23. Image [0:1] is-a ImageHeading [1:1];
24. Image [0:1] is-a ImageBoundary [1:1];
25. ...
26. TextHeading matches: /^[ A-Z ].*(:\w).*$/ ...;
27. TextTag matches: <b> <i> <strong> <font> <span> <acronym> <abbr>
<cite> <nobr> <sup>;
28. Photo matches: /src=''.*.jpeg''/;
29. Image matches: /src=''.*.gif''/;
30. ImageBoundary matches: if Linearity > 0,9, Size < 10;
31. Class matches: /class=''.*''/;
32. AltText matches: /alt=''.*\w.*''/;
33. ...

```

Figure 3.8: Example of the Classification Ontology for Basic Object.

Textual content is classified by rules relying on orthographic cues, and text length. For example, a text segment is considered a "heading" if it starts with an upper-case letter, ends with a lower-case letter, and has a text length below a certain threshold. Text segments can have attributes, such as "bold" or "emphasized", which are defined by text tags. For example, if the node starts with the text tag "b", the node has the attribute "bold".

An image is classified based on attributes that define the images dimensions, see section 3.4.4, that is, its format, the presents of alt-text, and the amount of times the image appears in the page. For example, if the image is compressed as a jpeg file it is classified as a photo, otherwise it is an image. An image can be classified as a "heading", a "boundary", or just as an image. If the image has attributes that specify its dimensions, its linearity and size can be calculated (see section 3.4.4). An image is considered a "heading" if it has a size and linearity above a certain threshold, and occurs just once in the page.

Links are classified by their content, and their attributes. They generally contain either a segment of text, or an image. Regardless of the content, image or text segment, the content is classified and the link inherits the type definition, and attributes of the content. A link also receives information on its type of navigation, for example, whether the source of the link is page or site internal, or site external, see figure 3.7.

### 3.6.2 Composite Objects

Classifying a composite object requires several more steps than the classification of a basic object. Before a CO can be classified its content must be mapped against certain constraints. Contextual constraints reveal faulty type definitions, and coherence constraints detect "granularity" violations amongst daughters and granddaughters.

The contextual constraints amongst sisters in this implementation alter just the types of basic objects, not the types of composite objects. For example, if a daughter has the type definition "heading" the following contextual constraints, among others, are applied to the node.

- A BO is not a "heading" if it appears in far right position of a CO, because a heading must be followed by at least one BO or CO.
- A BO is not a "heading" if its mother block node has the tag `<p>`.
- A BO is not a "heading" if it is a link and it is pointing to a section within the same page. Page internal links are primarily used as navigation links.

If a constraint above is matched, the current nodes annotation "text heading" is changed to "text phrase".

Coherence amongst daughters can be discovered in different ways. This implementation checks coherence based on certain attributes, and the patterns of type definitions amongst daughters and granddaughters. Checking the coherence amongst sisters can reveal faulty type definitions, and granularity violations. For example, coherent sisters can appear on different hierarchical levels, do to poor or inconsistent coding. This type of violation is detected by matching sequences of type definition patterns amongst adjacent BO daughters that are followed by a CO and the daughters of the CO, that is, by estimating the Longest Common Subsequence (LCS) of the types of objects between adjacent daughters and granddaughters.

For example, in A) below the LCS is calculated between the types contained in the sequences [BO1 BO2 BO3 BO4] (A1) and [BO5 BO6 BO7 BO8] (A2). If the LCS for A1 and A2 is above a certain threshold, A1 and A2 are seen as coherent blocks of information, that is, they create the same type of functional object.

```
A) [BO1 BO2 BO3 BO4 CO[BO5 BO6 BO7 BO8]]  
B) [CO1[BO1 BO2 BO3 BO4] CO2[BO5 BO6 BO7 BO8]]
```

Coherence can also be revealed by certain markup attributes. The use of Cascading Style Sheets (CSS), which describe the presentation style of content in a page, is increasing in Web page design. The use of CSS enables the presentation style of a document to be separated from its content, simplifying thereby Web authoring and site maintenance. Class attributes in HTML nodes are used to refer to specific styles defined in the CSS. Therefore, daughters with the same class attributes, that is, the same visual presentation, should also have the same type definitions. If this is not the case, type definitions are adjusted.

Once the content is verified, the content is checked for visual separators. If separators are present the content is segmented further, see section 3.5. Then, at last, the CO can be classified.

The classification of a CO is based on its content, and in few cases the layout structure of the content. The content analysis is shallow, and is based on the presents or absents of certain types of objects, and the order in which objects appear. For example, the content of a block node is classified as a "menu" if it contains at least two link objects. A menu can contain one type of "non-link object", i.e. a "heading". But the "heading" must occupy the far left position, either on the same level as the links, or one level above the links, in which case it precedes the "body" of the menu.

```

1.  Menu [-> CompositeObject];
2.  Menu [2:*] has-a Link [2:*];
3.  Menu [0:1] has-a MenuHeading [1:1];
4.  Menu [0:1] has-a NavigationType [1:1];
5.  MenuHeading matches: left most daughter is-a BasicObject and has-a
Annotation that matches TextType or LinkType which match eiter TextHeading
or ImageHeading;
6.  PageInternal [1:1] is-a NavigationType [1:1];
7.  SiteInternal [1:1] is-a NavigationType [1:1];
8.  SiteExternal [1:1] is-a NavigationType [1:1];
9.  SiteInternal matches: if all Href's in Link daughters point to a "site
internal location/server"
;

```

Figure 3.9: Example of the Classification Ontology for Composite Objects.

In certain cases the layout of segments can reveal a logical segment. For example, a rigid grid like structure can reveal the presence of a "genuine" table, as proposed by Penn et al. (2001). Specific HTML tags are used to create a grid like structure, e.g. <table>. But the <table> tag is frequently used to structure the layout of segments rather than to reflect logical structure. A grid like structure is detected, when classifying a CO, by a strong parallel coherence amongst daughters. The coherence is estimated based on the LCS of objects within daughters. For example:

- If all the daughters are COs and have the same type definition (rows), and
- If all the daughters have the exact same content, i.e. two or more BOs/simple COs (columns), then the Object is a Genuine Table.

In figure 3.2, the heading "Ekots fem senaste" (the five most recent news from Ekot) precedes a genuine table that has two columns (time, links) and five rows.

### 3.7 Extracting Generic Objects

Extracting functional objects from the semantically transformed parse tree is, in this implementation, straightforward and explicit. The extraction is a "direct extraction", that is, no further analysis is needed in order to determine which generic Web page components to extract.

When the analysis is completed the users is asked to define which objects are to be extracted. The implementation requires the user to use query terms that are part of the ontology defined for this specific analysis. The functional objects that can be extracted are, for example, links, sentences, paragraphs, headings, forms, menus, and "news content".

Once the query is chosen the tree is traversed top-down, mapping the query term to the annotations in the objects semantic representation. If the query and annotation match the object is extracted, and printed to an output file.

### 3.8 Notes on the Implementation

The algorithms described in the previous sections are all implemented in Perl 5.6.1. Various predefined Perl modules, which are freely distributed on the Web by the Comprehensive Perl Archive Network (CPAN) at <http://www.cpan.org>, are integrated in the program.

**LWP** is a module that is used to retrieve the documents from the web (section 3.4.1).

**URI** is a module that converts a relative URL into an absolute URL, for example,  
`www.cpan.org -> http://www.cpan.org.`

The URI module is used in several occasions to verify that an URL is absolute, for example, before a HTTP request is posed, or when creating the semantic representation of a link object.

**HTML::TreeBuilder** is used in section 3.4.2 to parse the HTML and to create a parse tree.

**HTML::Element** is used to traverse the parse tree, for example, in order to locate basic and block nodes in section 3.5, and to derive features and sub-nodes from the nodes of the tree.

**String::Trigram** calculates string similarity based on the trigram method, that is, by creating trigrams of a string and by comparing them to trigrams of another string and by calculating the number of matching trigrams. String similarity is used in specific contexts to determine if the URL of a link BO is associated with the URLs of link BOs in the following CO, revealing homogeneous information.

**Algorithm::Diff** calculates the Longest Common Subsequence (LCS) between two or more arrays. The LCS is used to estimate the coherence between adjacent COs, or between a sequence of BOs followed by a CO.

# 4 Experimental Results

The performance of the implementation is evaluated based on the results of two extraction experiments. In the first experiment a generic Web page component, "headings", is extracted from a set of test pages. In the second experiment a more specific Web page component is extracted, "site navigation menus". The two extraction experiments are evaluated based on the recall and precision for each page and site.

## 4.1 Evaluation Measures

The performance of the implementation is evaluated by extracting specific functional objects from pages in a test corpus, and by determining the relevance of the extracted data to the query.

There are two types of successful, and two types of unsuccessful actions generated by the extraction. The implementation is successful when extracting objects of interest, which are *true positives (tp)*, and when objects are rightfully not selected, which are *true negatives (tn)*. The system is unsuccessful if objects are mistakenly selected, *false positives (fp)*, and if relevant objects are not extracted, *false negatives (fn)*.

The relevance of the extraction results can be estimated based on recall and precision. Recall is the measure of relevant items selected by the system. Precision is the measure of the amount of items that the system has correctly selected.

$$Recall = \frac{tp}{tp + fn} \qquad Precision = \frac{tp}{tp + fp}$$

In order to calculate recall the total amount of target objects must be known. If the extraction space is composed of a very large amount of documents, for example the whole Web, calculating recall is impossible. This implementation extracts information from a very small amount of documents therefore recall can be calculated.

## 4.2 Test data

The test data is composed, similarly to the training data, of public Swedish sites, largely from government authorities. The test corpus is composed of 20 pages, from 10 different Web sites. The front page from

all sites is included in the corpus. The second page of the sites was chosen depending on if it contained relevant information to the extraction, or not.

In order to insure high quality HTML code, test site where chosen just if they specified which DTD standard they followed. All pages in the test set define which DTD markup standard the Web pages code is generated from. Most use HTML 4.01, just one site uses XHTML 1.0.

The code validity from the front page of all test sites was checked against W3C's SGML parser. Just one page proved to contain valid code according to the specified DTD standards. In some pages the inaccuracies were minor compared to others. But the validity test shows that even though the pages in the test corpus aim at following certain standards, they fail to do so to a 100 percent.

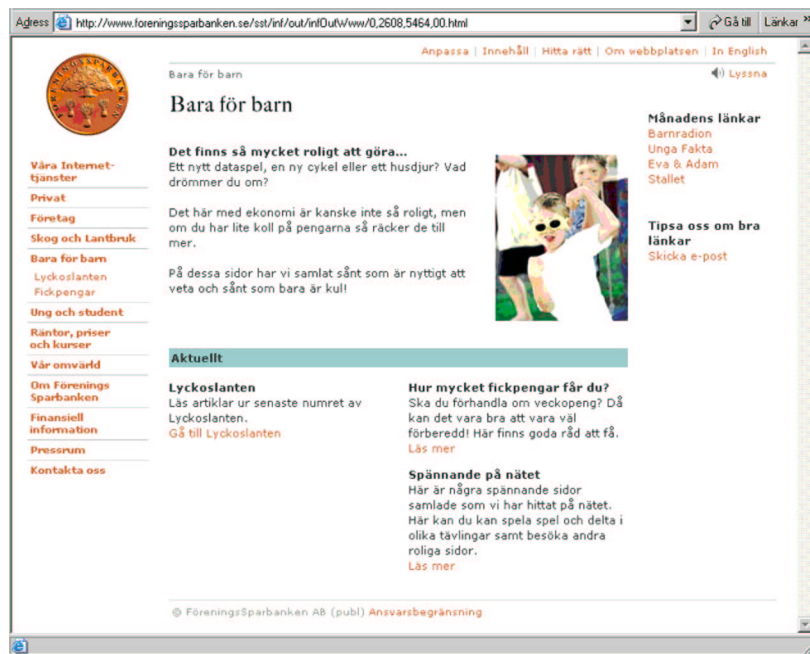


Figure 4.1: Example of a Test Page, <http://www.foreningsbanken.se>.

### 4.3 Extraction Experiment I

The aim of the first evaluation experiment is to extract headings from the test pages. I chose to test "heading extraction", since successfully identifying headings is vital to the document segmentation method presented in this thesis, headings function as segment boundary markers. Successfully identifying headings is relevant for application areas such as topic classification of documents, and document summarization, since headings are seen as important and content rich document segments. Before the results of the "heading" extraction are presented and discussed, the target object "heading" is described more in detail.

### 4.3.1 Extraction Target: "Headings"

A "heading" is a single word or a phrase that defines the topic of subsequent components, distinguishable through typographic variations, such as font size and color, as well as syntactic conventions, for example, short phrases and limited use of sentence final punctuations (no full stop). In HTML headings are presented as text or images (with alt-text), and they can also be links. Their typographic features are realized either explicitly in the code, by the use of certain tags (`<h1-h6>`, `<font>`) and attributes (`color="..."`), or implicitly, by the use of class attributes defined by site specific stylesheets, or a combination of both. The Web page displayed in figure 4.1 contains a total of 9 headings.

1. Bara för barn (small)
2. Bara för barn (big)
3. Det finns så mycket roligt att göra...
4. Aktuellt
5. Lyckoslantent
6. Hyr mycket fickpengar får du?
7. Spännande på nätet
8. Månadens länkar
9. Tipsa oss om bra länkar

### 4.3.2 Results Experiment I

The results of the *heading extraction* gave for all 20 test pages an average recall of 88% and precision of 91%, see table 4.1. The test set contained altogether 304 headings, an average of 15,2 headings a page. Out of the 304 headings, 37 relevant objects were not recognized, that is, 37 false negatives, and 25 false positive (irrelevant objects) were recognized by the implementation.

#### False Negatives in Experiment I

The 37 false negatives created by the *heading extraction* can be divided into three groups of each 12 mappings. The first group of false negatives is caused by *unconventional heading syntax*. For example, a small letter rather than a capital letter in the beginning of a heading, or the use of certain punctuation within the heading which is not recognized by the implementation, e.g. certain HTML ASCII features and full stops in URL's in a heading.

The second group of mismatches is caused by *ambiguous image dimensions*, for example, images that function as headings, which either lack dimensions (width and height), or use dimensions that are not recognized as heading dimensions (based on linearity and height). The ambiguity of image components is problematic, since the use of images and the image dimensions vary from site to site.

The third group of mistakes are caused by *irretrievable data*, for example, headings which are embedded in scripts or pictures (see figure 4.1, the large heading "Bara för barn" (Just for Children)), or image headings which lack alt-text (see figure 4.2).

<i>Test sites</i>	<i>TP</i>	<i>FP</i>	<i>FN</i>	<i>Recall</i>	<i>Precision</i>
http://svt.se	31	2	7	82%	94%
http://svt.se/svt/jsp/Crosslink.jsp?d=1803	19	1	13	59%	95%
http://www.lif.se	9	2	4	69%	82%
http://www.lif.se/Media/Media.asp	9	2	3	75%	82%
http://www.foreningssparbanken.se	5	1	2	71%	83%
http://www.foreningssparbanken.se/(bara för barn)	8	0	1	89%	100%
http://www.kth.se	5	1	0	100%	83%
http://www.kth.se/utbildning	15	4	3	83%	79%
http://www.arkitekturmuseet.se	10	3	1	91%	77%
http://www.arkitekturmuseet.se/om_museet/	17	1	0	100%	94%
http://skatteverket.se	7	1	0	100%	88%
http://skatteverket.se/servicetjanster/elda/elda.html	10	0	0	100%	100%
http://www.diabetes.se	1	0	0	100%	100%
http://www.diabetes.se/start.asp?sida=1834	52	6	0	100%	90%
http://www.uppsala.se	11	0	0	100%	100%
http://www.uppsala.se/templates/UKLevel2Page____3005.asp	8	0	0	100%	100%
http://www.funkanu.se	10	0	0	100%	100%
http://www.funkanu.se/start.asp?sida=59	25	1	3	89%	96%
http://www.ho.se	6	0	0	100%	100%
http://www.ho.se/start.asp?lang=sv&sida=418	10	0	0	100%	100%
<i>Total</i>	267	25	37	88%	91%

Table 4.1: The Results of the "Heading" Extraction, Presented in Average Recall and Precision for Two Test Pages from each Test Site (February 2004).

Figure 4.2 displays a section of the Web page <http://svt.se/nyheter/>. The elements are "image headings" followed by "links" (see B) in figure 4.2), and the section together form part of a column that looks like a menu. The implementation does not manage to recognize the "image headings" because they lack alt-text, they are perceived as boundaries. Consequently, the images are eliminated and the links become adjacent objects creating a rather incoherent menu (see C) in figure 4.2).

### False Positives in Experiment I

The false positives generated by the *heading extraction*, total of 25 cases, can be divided into two groups. A total of 16 false positives were caused by the use of *ambiguous syntax*, 12 of which were caused by the lack of sentence final punctuation, and 2 by the use of final punctuations that can appear in both sentences and headings, for example, "... " and "?". Finally, in 2 cases certain "date constructions" are recognized as headings instead of dates.

The second group of false positives, 9 in total, is caused by *contextual ambiguity*, that is, the implementation misinterpreted an object based on its contextual position. For example, a link with heading syntax, positioned immediately before a form lacking a heading, is recognized as the forms heading, but in truth they are not cohesive. Another example of contextual ambiguity is when components, displayed in a tabular layout, are recognized as headings although they are part of a row, this problem occurs in 3 cases.

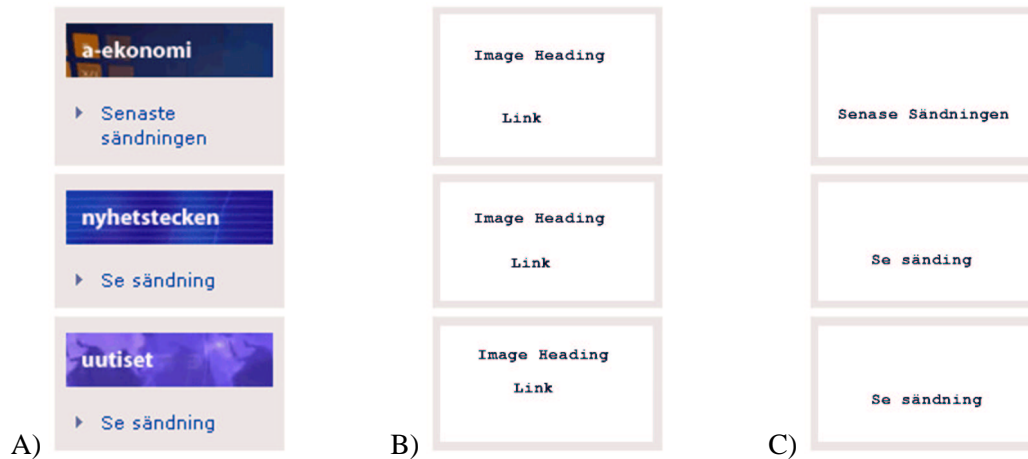


Figure 4.2: Example of a False Positive Generated in Experiment I and Experiment II. A) Displays Part of a Column from the Page <http://svt.se/nyheter/>. B) the Conceptual Model of the Content in A). C) Shows How the Impementaion Analyzes A).

## Discussion Experiment I

The recognition of headings with unconventional syntax can be improved, to a certain extent, by extending the classification ontology of headings, and by improving the conversion of HTML ASCII features to their textual counterparts. But recognizing headings that start with small letters would require a more extensive contextual analysis. The recognition of image headings, which either lack dimensions, have ambiguous dimensions (square rather than linear), or have alt-text which start with small letters could probably be improved if the analysis would incorporate a more extensive contextual analysis based on the heading probability of questionable images.

The false negatives caused by irretrievable data, either because images lack alt-text, or because headings are embedded in scripts or pictures, are more difficult to solve. Headings embedded in scripts could be recognized by parsing the content of scripts. But just one site of the 20 sites, in both the training and test set, uses headings embedded in scripts. Therefore, parsing script content in order to determine if they contain headings could decrease the performance, and especially the affectivity of the implementation in contrast to the gain of performance.

Headings that are embedded in images and pictures, which lack alt-text, would require extensive image analysis in order to extract the text of the headings, which isn't seen as a very realistic solution. A better solution would probably be to generate an image description by, for example, analyzing adjacent text segments, or, in case the image is a link, by extracting the title of the page that the link is referring to.

False positives are generated by ambiguous use of syntax conventions, for example, by the lack of final punctuation, or the presents of heading final punctuations. It is questionable though, whether sentence final punctuations should be regarded as an orthographic heading criterion or not, since they appear rather rarely in headings. Compensating for spelling mistakes, e.g. the lack of sentence final punctuations, should maybe not be regarded as well. These mistakes are easiest taken care of by the Web page creator. But, the implementation must be checked if it in such cases can detect typographic heterogeneity between the false positive and adjacent text segments. If not, the certainty of the heading classification should be decreased if there is no typographic heterogeneity, or variation in modality between a heading and adjacent segments.

False positives generated by contextual ambiguity may be more difficult to solve, because they are generated by, for example, a semi-tabular layout of information that would require a higher awareness of the linearity of a table, or by an unclear conceptual model of the position of document components, for example, in the false positive where a link precedes a form and is perceived as the heading of the form. Had there been a larger visual gap between the two elements, such as two line breaks or an image, they would not have been perceived as components belonging together. This example could have also been a problem for the visual interpretation of the two components. But the content of the link cannot be mistaken for a heading, because it contains a phrase that is almost idiomatic, that is, "In English". The link is referring to a page that displays the information in the current page in English. In order to resolve this type of ambiguity some form of linguistic analysis would be required. One way to resolve the misinterpretation of idioms as headings, which in general are not used in headings, is by incorporating the use of keywords or "key phrases" in the heading classification that restrict the content of a heading.

If all Web page authors would use predefined heading tags `<h1>`-`<h6>`, headings could be extracted from Web documents directly. But rather few Web creators rely just on heading tags. In the training set, 6 of 20 pages use predefined heading tags, and in 4 of them other forms of headings are used as well, differentiated from surrounding text by, for example, the text emphasis tag `<strong>`. A majority of the pages in the training set use heading tags, 14 of 20 pages. But 9 of the 14 pages also use other types of headings; headings embedded in, for example, images. It would be interesting to study the headings syntax more, but also the markup of headings in a larger group of documents, in order to get a clearer picture of how headings are differentiated from other sentence fragments.

## 4.4 Extraction Experiment II

The second extraction experiment extracts "site navigation menus" from all the pages in the test set. The reason why I chose to extract "site navigation menus", is to illustrate how the generic analysis performed by this implementation can be used to extract a more specific, composite Web page component, based solely on the presents of specific features in the menu semantic representation and its location in the semantically transformed parse tree.

Identifying and extracting menus, such as site navigation menus, is relevant for application such as Web page adaptation. Adapting the web content is required when browsing a Web page with a mobile phone as monitor, or when a speech synthesizer reads the content of a page to a visually impaired user. Filtering of noisy information, for example, in document summarization, is another area where the identification of menus could be relevant.

### 4.4.1 Extraction Target: "Site Navigation Menus"

A "menu" in a Web page is a list of links that provide some form of navigation. A menu contains two or more link objects according the analysis proposed here. The main purpose of a link is to navigate from one resource to another. Links are either images, text, or an area of an image that is associated with a navigation resource. There are different resource locations, either page or site internal, or site external. The resources in turn are static documents, or created dynamically, and displayed either in the same browser window or in a new page. All these features are presented in the semantic representation of links, which in turn is the basis of the classification and semantic representation of the menu.

I find that the function of a menu varies depending on what type of resources that are accessed. For example, if all the links point to site internal resources, the menu purpose is to provide "site internal navigation". A menu function is also affected by its context. For example, if a menu has one sister, e.g. a paragraph to its left, and their mother node is a heading, then the links in the menu are likely related to the topic in the heading and the preceding paragraph. In which case the function of the menu would be "text topic navigation", regardless of where and how the recourses are stored and presented.

A "site navigation menu" is a menu that contains links to site internal resources, and no site external links. It may contain links to sections within the same page, but just to a certain percentage. If there are too many links to page sections the menu is a "page navigation menu", not a "site navigation menu".

The Web page displayed figure 4.1 contains three menus, the column to the far left, the row at the top of the page, and the menu with the heading "Månadens länkar" (The Links of the Month). Of the tree menus, "Månadens länkar" is site external and the two others are site internal, which are the ones to be extracted.

#### 4.4.2 Results Experiment II

The *site navigation menu* extraction reached an average recall of 90%, and precision of 90%. The pages contained on average 2.4 site navigation menus, a total of 48 target menus in the whole test set. Out of which the implementation failed to recognize 5 true positives, and created 5 false positive mappings.

<i>Test sites</i>	<i>TP</i>	<i>FP</i>	<i>FN</i>	<i>Recall</i>	<i>Precision</i>
http://svt.se	4	1	0	100%	80%
http://svt.se/svt/jsp/Crosslink.jsp?d=1803	5	4	0	100%	56%
http://www.lif.se	3	0	0	100%	100%
http://www.lif.se/Media/Media.asp	3	0	0	100%	100%
http://www.foreningssparbanken.se	1	0	1	50%	100%
http://www.foreningssparbanken.se/(bara för barn)	1	0	1	50%	100%
http://www.kth.se	2	0	0	100%	100%
http://www.kth.se/utbildning	2	0	0	100%	100%
http://www.arkitekturmuseet.se	3	0	0	100%	100%
http://www.arkitekturmuseet.se/om_museet/	3	0	0	100%	100%
http://skatteverket.se	2	0	1	67%	100%
http://skatteverket.se/servicetjanster/elda/elda.html	0	0	2	0%	0%
http://www.diabetes.se	1	0	0	100%	100%
http://www.diabetes.se/start.asp?sida=1834	1	0	0	100%	100%
http://www.uppsala.se	5	0	0	100%	100%
http://www.uppsala.se/templates/UKLevel2Page_...3005.asp	3	0	0	100%	100%
http://www.funkanu.se	1	0	0	100%	100%
http://www.funkanu.se/start.asp?sida=59	1	0	0	100%	100%
http://www.ho.se	1	0	0	100%	100%
http://www.ho.se/start.asp?lang=sv&sida=418	1	0	0	100%	100%
<i>Total</i>	43	5	5	90%	90%

Table 4.2: The Results of the "Site Navigation Extraction", Presented in Average Recall and Precision for Two Test Pages from each Test Site (February 2004).

## False Negatives in Experiment II

The false negatives generated by the *site navigation menu extraction*, a total of 5 mappings, where caused by *irretrievable data*. The data was irretrievable, in 2 cases, either because it was generated dynamically by scripts (see figure 4.1, the menu in left most column), or by not being able to access the content of frames, 3 cases.

## False Positives in Experiment II

The false positives of the *site navigation menu extraction*, a total of 5 mappings, where generated by *incorrect content* or by *ambiguous content*. In the first case objects are misinterpreted by the implementation because of previous misinterpretations, either by the classification or segmentation. For example, if image headings are recognized as boundaries they are eventually eliminated by the analysis, generating thereby "incorrect content". This specific example is illustrated in the figures, see figure 4.2.

False positives generate by *ambiguous content* are, for example, a selection of links which visually are perceived as a single unit, but are realized as a CO, that is, two or more objects (see figure 4.3).



Figure 4.3: Example an Ambiguous Component that generates a False Positive in Experiment II, the Logo from <http://svt.se/>.

Figure 4.3 displays the logo of the site <http://svt.se/>. The logo is created of two "link images" both of which have alt-text, but visually they appear as one unit. Since the two logo elements are links, and they are adjacent, the unit is recognized as a menu. The implementation does not check for homogeneity of adjacent links that are siblings, in neither the URL nor the alt-text. If that would be the case, the two links could be seen as 'homogeneous', hence, one and the same component. Their URL's are identical, and their alt-texts are very similar: "SVT.SE - Till förstasidan på svt.se" and "SVT's blomma - Till förstasidan på svt.se".

### 4.4.3 Discussion Experiment II

The pages from three sites affect the results or the menu extraction: <http://svt.se>, <http://skatteverket.se>, and <http://www.foreningssparbanken.se>. The false negatives are caused by irretrievable data, embedded in scripts ([foreningssparbanken.se](http://www.foreningssparbanken.se)), or frames ([skatteverket.se](http://skatteverket.se)). Data embedded in scripts could be accessed with a script parser. But accessing the content in frames is not quite solved by this implementation yes. The content of the pages from "skatteverket" is displayed in three different frames. When analyzing the front-page two frames are accessed, and the content is extracted, but the access to one of the frames is denied, in which case the content can clearly not be retrieved. When analyzing the second page of the site, the URL to the main frame must be given directly to this implementation. The result of which is that just the main frame is analyzed, because this implementation does not store the result of the frame extraction for the front-page. Consequently the menu extraction performance is very poor, since the two

menus of the page are stored in the top and left frame of the page and not in the main frame. The implementation must be improved in order to handle pages that are embedded in frames, especially when analyzing other pages than the front-page of the site.

The false positives of the site navigation menu extraction are generated by previously misinterpreted content (see figure 4.2), or ambiguous content see figure (4.3). In both these cases homogeneity values of adjacent siblings could reveal that their content is very similar. In the case of the "incoherent menu", the links are lexically similar but their URLs are not, knowledge which could indicate that a mistake has occurred in the segmentation. In the case of the "logo menu", the URLs and the content of the two links are homogeneous, indicating that the CO should be rather classified as a BO rather than a CO.

## 4.5 Results compared to Related Work

It is quite difficult to compare the performance of this implementation to related work, since the use of Web Document Analysis and Information Extraction systems varies greatly, but also because not all researchers present their results. In most cases the precision and recall rates for extracting information from a Web page are above 90%.

For example, Lin and Ho (2002) extract content blocks based on a similar document analysis approach, but their classification is based on entropy values. They achieve a recall and precision rate of about 95%. Fong et al. (2002) extract publication information, that is, authors names and the title of an article from Web publications. Their document analysis is strongly based on structural HTML tags. They achieve an extraction accuracy of 93%. Mukherjee et al. (2003) test their Web document analysis approach by extracting headings from New Sites. Their approach differs from this implementation in that they focus just on "News headings", that is, domain specific headings not generic headings as proposed here, and in that they classify headings based on keyword mappings to an ontology for News headings. They reach an accuracy rate of approximately 95% for extracting "Major Headline News" from 7 different News Web pages.

The performance of this implementation reaches a precision rate that is slightly higher than the recall rate in the "heading extraction" experiment, 91% vs. 88%. In the "menu extraction" experiment recall and precision are both 90%. The results of the experiment cannot be seen as a very accurate measure of performance since the test and the training sets of the implementation are very small, each 20 pages from 10 different sites. The extraction performance is rather an indication on whether the notion of Web page analysis proposed by this thesis is realistic or not. Since the average recall and precision rate for this implementation is around 90% the Web document analysis approach proposed in this thesis seems promising. But, a lot of improvements can be made with the implementation and also further testing is required in order to determine whether this analysis approach is a good or not.

# 5 Conclusions

In this thesis, a generic approach to Web document analysis is proposed and implemented. The document analysis approach transforms a layout-based parse tree of a Web page, derived from the HTML markup of the page, into a logically annotated parse tree. The transformation is generated based on mappings to a domain specific ontology, and segmentation cues given by the positions of visual boundaries and heading segments.

Experimental results gained from extracting headings and site internal navigation menus achieve a recall and precision rate of approximately 90%. These results can be seen as an indication for that the proposed Web document analysis approach is promising. But further testing and improvements to the implementation are required. A limitation to the suggested approach is that the analysis requires well-formed code, and a consistent use of HTML tags. It also requires a concise and coherent conceptual model of the document components within a page.

There are two main reasons to why the implementation misinterprets data, ambiguity and irretrievability. The main factor that affects recall negatively is irretrievable data, either because data is embedded in scripts, created dynamically, or lacking (alt-text). But both recall and precision are affected by ambiguous syntax and/or ambiguous document structure.

Solving the problem with irretrievable data could be difficult. A script parser could, to a certain extent, retrieve data embedded in scripts. But in cases where, for example, images lack alt-text, more extensive image analysis would be required in order to classify the content and extract possible text segments in the image. Analyzing the text segments adjacent to the image could possibly be used to generate an image description. In cases where "link images" lack alt-text text content from the links source could be analyzed in order to provide the "link image" with a linguistic description.

The analysis of ambiguous content could be improved by expanding the context analysis, for example, by regarding certainty estimations of the classification. Probability values, which reflect the certainty of the classification of an object, together with contextual constraints are already used in the implementation, in cases where images could function as headings and have ambiguous "image heading" dimensions. But a more extensive use of probability estimations could be used in order to increase or decrease the certainty of both BO and CO classification.

Content analysis of ambiguous objects could also be improved by estimating the lexical and/or structural homogeneity of adjacent objects. Homogeneity could reveal segmentation violations as in the example illustrated in figure 4.2, where the content of a CO contains rather redundant information, that is, the granularity of the document segment is too fine.

Further testing of rules that deal with the classification of ambiguous components is required, and may reveal if they are necessary or not. For example, the necessity of allowing sentence final punctuations such as ? in headings is, for example, questionable. Further tests should examine whether this specific rule over-generates the heading classification more than under-generates, if it were to be removed.

It would be interesting to test the performance and usefulness of the generic Web document analysis for applications within Language Engineering. For example, by filtering document components from documents such as non-content sections, containing components such as menus and forms, from content-rich documents in order to retain just the main document content. From which, for example, query expansion terms could be derived for pseudo-relevance feedback.

Applying the generic document analysis to documents in other languages than English and Swedish, as tested so far, would also be interesting. The similarity between two documents in two different languages could possibly be estimated based on the generic document structure. The components annotations and "position numbers", which reveal the position of a component in the parse tree, could be used to align whole documents or just specific document components in order to create "document structure-based" parallel alignments. The content of the alignments could then be analyzed further, applying syntactic and semantic analysis methods, in order to locate translation candidates.

# Bibliography

- Aas, G. (2003). HTML::Parser.  
Available at: <http://search.cpan.org/~gaas/HTMLParser3.34/Parser.pm> .
- Appelt, D. and Israel, D. (1999). Introduction to Information Extraction Technology, *Tutorial for the International Joint Conference on Artificial Intelligence*, Stockholm, Sweden.  
Available at: <http://www.ai.mit.edu/people/jimmylin/papers/introtoie.pdf> .
- Boguraev, B. and Neff, M. (2000). Discourse Segmentation in Aid of Document Summarization.  
Available at <http://portal.acm.org/citation.cfm?id=820269&dl=ACM&coll=portal>.
- Burke, S. M. (2002). *Perl & LWP*, O'Reilly, Sebastopol, CA, USA.
- Cai, D., Yu, S., Wen, J.-R. and Ma, W.-Y. (2003). Extracting Content Structure for Web Pages based on Visual Representation, *Asia Pacific Web Conference (APWeb 2003)*, pp. 406-417.  
Available at: <http://www.dbs.informatik.unimuenchen.de/~spyu/paper/VIPSAPWeb.pdf> .
- Chen, J., Zhou, B., Shi, J., Zhang, H. and Wu, Q. (2001). Function-based Object Model Towards Website Adaptation, *Proc. of the 10th International World Wide Web Conference*, Hong Kong, China, pp. 587-596.  
Available at: <http://www10.org/cdrom/papers/pdf/p296.pdf> .
- Doermann, D., Rivlin, E. and Rosenfeld, A. (1997). The Function of Documents, *4th International Conference Document Analysis and Recognition (ICDAR '97)*, Ulm, Germany.  
Available at: <http://csdl.computer.org/comp/proceedings/icdar/1997/7898/00/78981077abs.htm>.
- Embley, D. W., Jiang, Y. and Ng, Y. K. (1999a). Record-Boundary Discovery in Web Documents, *Proc. SIGMOD'99*, Philadelphia, PA.
- Embley, D. W., Tao, C. and Liddle, S. W. (1999b). Automatically Extracting Ontologically Specified Data from HTML Tables with Unknown Structure, *Lecture Notes in Computer Science*, Volume 2503, Issue , pp 322-337.  
Available at: <http://www.deg.byu.edu/papers/er02.pdf> .
- Fong, A. C. M., Hui, S. C. and Vu, H. L. (2002). Effective Techniques for Automatic Extraction of Web Publications, *Online Information Review*, vol.26, no.1.
- Gu, X.-D., Chen, J., Ma, W.-Y. and Chen, G.-L. (2002). Visual Based Content Understanding towards Web Adaptation, *Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive WebBased Systems*.  
Available at: <http://portal.acm.org/citation.cfm?id=647458.728405&dl=GUIDE&dl=ACM>.

- Gupta, S., Kaiser, G., Neistadt, D. and Grimm, P. (2003). DOM-Based Content Extraction of HTML Documents, *Proceedings of the twelfth international conference on World Wide Web*, Budapest, Hungary.  
Available at: <http://portal.acm.org/citation.cfm?id=775182&dl=ACM&coll=GUIDE>.
- Kushmerick, N. (1997). *Wrapper Induction for Information Extraction*, PhD thesis, University of Washington.
- Leander, A., Ribeiro-Nato, B., da Silva, A. and Texeira, J. (2002). A Brief Survey of Web Data Extraction Tools, in ACM SIGMOD, Volume 31, Issue 2, New York, NY.  
Available at: <http://portal.acm.org/citation.cfm?id=565137&dl=ACM&coll=GUIDE> .
- Lin, S.-H. and Ho, J.-M. (2002). Discovering Informative Content Blocks from Web Documents, *In Proceedings of the eighth AMC SIGKDD international conference on Knowledge discovery and data mining*.  
Available at: <http://portal.acm.org/citation.cfm?id=775134&dl=ACM&coll=portal> .
- Mukherjee, S., Yang, G. and Ramakrishnan, I. (2003). Automatic Annotation of Content-Rich HTML Documents: Structural and Semantic Analysis, *In Proceedings of the Second International Semantic Web Conference*, Florida, U.S.A.
- Penn, G., Hu, J., Luo, H. and McDonald, D. (2001). Flexible Web Document Analysis for Delivery to Narrow-Bandwidth Devices, *ICDAR 01*, Seattle, USA.
- Resnik, P. and Smith, N. A. (2003). The Web as a parallel corpus, in Computational Linguistics, Volume 29, MIT Press, Cambridge, MA, USA.  
Available at: <http://portal.acm.org/citation.cfm?id=964751.964753&dl=portal&dl=ACM> .
- Robie, J. (1998). What is the Document Object Model.  
Available at: <http://www.w3.org/TR/WDDOM/introduction.html>.
- Rosenfeld, L. and Morville, P. (1998). *Information Architecture for the World Wide Web*, O'Reilly.
- Soderland, S. (1999). Learning Information Extraction Rules for Semi-structured and Free Text, *Machine Learning 34*, Kluwer Academic Publishers, Boston, MA.
- Summers, K. (1998). *Automatic Discovery of Logical Document Structure*, PhD thesis, Cornell University.
- Tang, Y., Cheriet, M., Liu, J., Said, J. and Suen, C. (1999). Document Analysis and Recognition by Computers, *Handbook of Pattern Recognition and Computer Vision*, Chapter 8, World Scientific Publishing Company.
- Yang, Y. and Zhang, H. (2001). HTML Page Analysis Based on Visual Cues, *In Proceedings of the 6th International Conference on Document Analysis and Recognition*, Seattle, USA, Sept. 1013.
- Yu, S., Cai, D., Wen, J.-R. and Ma, W.-Y. (2003). Improving Pseudo-Relevance Feedback in Web Information Retrieval Using Web Page Segmentation, *In Proceedings of the twelfth international conference on World Wide Web*, Budapest, Hungary.  
Available at: <http://portal.acm.org/citation.cfm?id=775155&dl=ACM&coll=portal>.