BRNO UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Information Systems

# Information Extraction from HTML Documents Based on Logical Document Structure

## Extrakce informace z HTML dokumentů na základě logické struktury dokumentu

by

Radek Burget
M.Sc., Brno University of Technology, 2001

Field: Information Technology

Supervisor: Jaroslav Zendulka, associate professor

Opponent1:

Opponent2:

Opponent3:

Submitted on:

Defended on:

# Abstract

The World Wide Web presents the largest Internet source of information from a broad range of areas. The web documents are mostly written in the Hypertext Markup Language (HTML) that doesn't contain any means for semantic description of the content and thus the contained information cannot be processed directly. Current approaches for the information extraction from HTML are mostly based on wrappers that identify the desired data in the document according to some previously specified properties of the HTML code. The wrappers are limited to a narrow set of documents and they are very sensitive to any changes in the document formatting.

In this thesis, we propose a novel approach to information extraction that is based on modeling the visual appearance of the document. We show that there exist some general rules for the visual presentation of the data in documents and we define formal models of the visual information contained in a document. Furthermore, we propose the way of modeling the logical structure of an HTML document based on the visual information. Finally, we propose methods for using the logical structure model for the information extraction task based on tree matching algorithms. The advantage of this approach is certain independence on the underlying HTML code and better resistance to changes in the documents.

# Contents

# 1 Introduction

The World Wide Web presents currently the largest Internet source of information from a broad range of areas. One of the main reasons of this great expansion is the absence of any strict rules for the information presentation and the relative simplicity of the used technology. The orientation to documents and the HTML language that is mainly used for creating the documents, gives the authors enough freedom for presenting any kind of data with minimal effort and the new technologies such as Cascading Style Sheets (CSS) allow to achieve the desired quality of presentation. Together with the hypertext nature of the documents, these properties make the World Wide Web a distributed and dynamic source of information.

On the other hand, the loose form of the data presentation brings also some drawbacks. With the increasing number of available documents a problem arises, how to efficiently access and utilize all the data they contain. Due to the above properties of the web, related information is often presented at different web sites and in diverse forms. Accessing this data by browsing the documents manually is a time-consuming and complicated task. For this reason, it is desirable to facilitate the access to the web by processing the contents of the documents automatically to certain extent. As a first step, there exists an effort to provide the users with centralized views of related data from various sources in the World Wide Web such as the services for comparing the prices of goods in on-line shops. The next step is presented by the data mining techniques that have been developed for the database branch.

For all these tasks, it is necessary to access the data that is contained in the documents. This is, however, not a trivial problem. As mentioned above, the state-of-the-art web consists mainly of the documents written in the Hypertext Markup Language [32]. This language is suitable for the definition of the presentation aspect of the documents but it lacks any means for the definition of the content semantics. The information contained in the documents can be therefore very hardly interpreted and processed by a computer.

Possible answer to this problem is the proposal of *semantic web* [5] that is based on different technology and has quite different nature. While the classical web can be viewed as a distributed document repository, the semantic web has many characteristics of a distributed object database [22]. Although this technology is very promising and it is being developed rapidly, it doesn't solve processing of the great amount of documents that are already available in the "legacy" web. Moreover, simultaneously with the semantic web, the legacy web is still growing and developing. In contrast to the semantic web, the evolution of the technology has, however, quite different direction. Currently, the most important issue in the development of the classical web technologies is the flexibility of the web design and effective management of the web content. For the information providers, it is not the aim to

allow better automatic processing of the documents on the web and in some cases it is even undesirable. All these facts together with the great amount of information that is virtually available make the automatic HTML document processing an interesting and challenging area of investigation.

## 1.1 Information Extraction

Our aim in the HTML document processing is to identify a particular information that is explicitly contained in the text of the document and to store it in a structured form; e.g. to a database table or a XML document. This process is usually called *information extraction* from the HTML documents [12, 17, 26].

As an example, let's imagine a set of pages containing information about various countries. The task of information extraction in this case is to identify the values of country name, area, population etc. and to store them in a structured way. The result of processing such a set of documents could be a database table containing the appropriate values for each country.

The process of extracting information from the World Wide Web can be split to following steps:

1. *Localization of relevant documents* – an information retrieval phase. This task is done by search engines that index the WWW contents such as Google[1].

2. *Identification of the data in the documents* – the information extraction phase itself.

3. *Data storage* – for example to a relational database or an XML document.

In this thesis, we focus on the parts of the information extraction process that are not sufficiently explored yet. Within the scope of the step 1 it is the problem of the discovery of *logical documents* that may be formed by a set of related HTML documents. We discuss the existing approaches and we propose certain improvements of current techniques. The main focus of this work is on the step 2 – the data identification phase. We analyze the problems of current methods and we propose a novel approach to this problem based on modeling the visual aspect of the documents and their logical structure.

# 2 State of the Art

Our approach to information extraction combines several areas of endeavour that have been intensively investigated recently. Firstly, it is the area of information

[1] http://www.google.com/

extraction from HTML documents that mainly focuses on the problems related to *wrapper generation* and *wrapper induction*. And secondly, it is the area of advanced modeling of the structure of documents and the discovery of semantic structures in the documents.

As mentioned in section 1.1, the hypertext capability of HTML allows to create larger information entities consisting of multiple documents linked together that are usually called *logical documents*. A complex approach to information extraction therefore includes the analysis of the logical documents.

## 2.1 Information Extraction from HTML Documents

The information extraction has been largely investigated in the plain text context – it has been used for processing electronic mail messages, network news archives etc. [30]. The HTML language brings a new look to this problem. In contrast to the plain text messages, HTML allows to define the visual presentation of the content. This possibility is often used for making the documents clearer and easily understandable. The data in HTML documents is presented in a more or less regular and structured fashion. For this reason, the HTML documents are often regarded a *semistructured information resource* [26]. The reader is not forced to read the whole document, he can easily find the part of the document containing the desired information. For this purpose, the documents contain a system of navigational cues that have mostly visual character. During the years of the World Wide Web development, these techniques of data presentation have been brought almost to perfection so that reading the documents using the web browser becomes relatively efficient.

From the point of view of the automated document processing, the situation is different. HTML doesn't contain sufficient means for the machine-readable semantic description of the document content. The techniques for natural language processing that have been used for the information extraction from plain text are not applicable, because HTML documents usually don't contain many whole sentences or blocks of continuous text. On the other hand the HTML tags inserted to the text of the document provide additional information that can be used for identifying the data.

For the data in the HTML documents, a database terminology is usually used in the information extraction context. We assume that the documents contain one or more *data records* where each record consists of some number of *data fields*. Usually, we admit that some records are incomplete; i.e. that the values of some fields are not present in the document.

### 2.1.1 Wrappers

According to Kushmerick [26], a *wrapper* is a procedure that provides the extraction of particular data from an HTML document. The key part of the wrapper is formed by a set *extraction rules* that define the way, how individual data fields are identified in the HTML code. With *wrapper construction*, we mean the process of formulating the extraction rules for a particular task.

Each wrapper is designed for a limited set of documents that correspond to previously defined extraction rules. In literature, such a set of documents is commonly called a *document class*. In most cases, the document class consists of documents of the same topic generated automatically from a back end database by an identical procedure or at least created by the same author. Moreover, the wrapper works until the document changes so that it doesn't correspond more to the extraction rules. Due to the distributed and dynamic nature of the Web, this state cannot be predicted.

The most obvious method of wrapper construction is writing the wrappers *by hand*; i.e. by employing an analyst that analyzes the set of documents to be processed, determines the rules that apply for these documents and creates the wrapper procedure. Since this approach presents a serious scalability problem, many approaches have been developed for an automatic inference of wrappers.

For the inference of the extraction rules, a model of the document text and the embedded tags must be created. The character of this model depends on the used inference algorithm. In following sections, we give an overview of common models of the document and existing methods of the wrapper induction.

## 2.2 Document Code Modeling

The most straightforward model is to represent the document code simply as a string of characters. In this representation, the text of the document is not explicitly distinguished from the embedded tags. For processing the documents represented this way, usually the extraction rules based on delimiting substrings [35] or regular expressions [2] are used. For example, we can assume that the words that end with a colon can introduce an important data value. Such phrases can be found using the regular expression `[A-Za-z0-9_]+[:]`.

For using the machine learning algorithms, it appears that it is more suitable to use a different representation where the basic unit is a word instead of a character. The document is represented as a sequence of words. Each word can be assigned various attributes based some of their orthographical or lexical properties. The embedded HTML tags can be either omitted or used for inferring additional attributes of the individual words [17] (e.g. the text forms a caption). A special case of such model is used by [23]. In this model, on the contrary, the HTML tags are regarded as the symbols of an alphabet $\Sigma$; any text string between each pair of subsequent tags is represented with a reserved symbol $x$.

The most common is a hierarchical model of the HTML code that represents the nesting of the tags in the document [6, 10, 12, 13, 14, 25]. In order to make it possible to create such a model for an HTML, it is necessary to pre-process the document so that we obtain so called *well-formed document* [6] where all opening tags have the corresponding closing tag and they are properly nested. The text content of the document is then contained in the leaf nodes of the tree or it is not included in the model at all. It is the advantage of the hierarchical model that it describes the relations among the tags in addition to the observed properties of individual words and tags.

## 2.3 Wrapper Induction Approaches

Current methods of the wrapper induction are based on the knowledge from various areas of the research work. Many approaches are based on grammatical or automata inference [11, 16, 23, 25, 26], other approaches use the relational machine learning algorithms [10, 12, 17, 35]. Quite different approach is presented by the methods based on conceptual modeling [13, 14]. Note that this is a coarse classification only and the different approaches influence each other.

The wrapper induction approaches require a set of labeled examples of the documents that are used for inferring the extraction rules. According to the artificial intelligence terminology, we call this set of examples a *training set* and the process of inferring the extraction rules the wrapper *training*.

For evaluating the performance of the individual information extraction approaches, there are two commonly used metrics: the *precision* $P$ and *recall* $R$ [25]. They are defined as follows:

$$P = \frac{c}{i} \tag{1}$$

$$R = \frac{c}{n} \tag{2}$$

where $c$ is the number of correctly extracted records, $i$ is the total number of extracted records and $n$ is the real number of the records in the document.

### 2.3.1 Methods Based on the Grammatical Inference

The grammatical inference is a well-known and long studied (early studies in the 60's) problem and its application to information extraction is therefore supported by a large theoretical foundation. The problem is following: We have a finite alphabet $\Sigma$ and a language $L \subseteq \Sigma^*$ (usually, regular or context-free languages are discussed in this context). Given a set $S^+$ of the sentences over $\Sigma$ that belong to $L$ and a (potentially empty) set $S^-$ of the sentences over $\Sigma$ that do not belong to $L$ we want to infer a grammar $G$ that generates the language $L$.

The basic idea behind using grammatical inference for information extraction is that generating a wrapper for a set of HTML documents corresponds to a problem of inferring a grammar for the HTML code of the pages and finally using the inferred grammar for the extraction of the data fields. This idea is however not directly applicable to the document processing. The main obstacle is that there are only positive examples available in the web; i.e. the available documents. As it follows from Gold's work [19], neither regular nor context-free grammars cannot be correctly identified from the positive samples only. This problem can be basically solved by limiting the language class to a subclass of regular languages that is identifiable in the limit (e.g. $k$-reversible languages) or by changing the computational model (artificial negative samples or supplying an additional information).

The grammatical inference approach is used for example in [16] in combination with a Bayes classifier. Kosala [25] applies the grammatical inference theory to tree languages whereas Hong et al. [23] use stochastic context-free grammars in combination with a domain knowledge represented by regular expressions.

A completely new view of the problem is presented by [11]. The presented approach deals with the problem of schema discovery – given a set of HTML documents we are looking for a common hierarchical schema of their content and the extraction rules are based on the discovered schema. The schema discovery is based on comparing the document code – the parts that are present in all the documents are considered as static content whereas the variable parts correspond to the data values.

### 2.3.2 Methods Based on Relational Learning

General principle of these techniques is similar to the above. Again, we assume that there exists a class of documents with similar properties and that there is a training set of the documents from this class available. In the training set, we describe some properties for each data field to be extracted using logical predicates. Then, we use relational learning algorithms for inducing general rules that identify the data fields in documents.

Freitag [17] assigns each word in the documents certain attributes based on the properties of the given portion of the text such as word length, character type (letters, digits) or orthography and adds some additional attributes that describe the relation between the word and the surrounding HTML tags (e.g. the word forms part of a heading or the word forms a table row). Each data field to be extracted is then described by logical predicates based on these attributes and using an algorithm SRV (based on the FOIL algorithm [31]) a general rule is inferred that identifies the data field in the document. DiPasquo [12] extends this approach by modeling the hierarchical structure of HTML tags in the document, which allows describing the relations among the HTML tags more exactly. Similar approach is

used by Soderland [35].

## 2.4 HTML Code Analysis

Wrapper induction is not the only method for automatic wrapper construction. Following techniques are based on specific algorithms for the analysis of the document HTML code. The aim of these techniques is to avoid the training phase and to eliminate the requirement of the training set of document. On the other hand, these techniques are usually based on empirical heuristics and it is often hard to specify exactly for which document the method is suitable. Furthermore, some predefined domain-specific knowledge is often required.

Ashish [2] locates some important words that introduce an important information in the document (e.g. *Geography*, *Transportation*, etc.) – so called *tokens*. Other works [6, 13] are based on the discovery of unified separators of the data records and fields in the document.

## 2.5 Conceptual Modeling

Conceptual modeling approach is more common in the area of the information extraction from plain text documents; however, it can be used for HTML documents too. For example, Embley et al. [13, 14] propose a method where as the first step, an ontological model of the extracted information is created and based on this model, corresponding data records are discovered in the document. It is possible to combine this approach with the HTML code analysis described above. The main difference is that the structure of the information is not inferred from the document but it is known in advance.

## 2.6 Semi-automatic Wrapper Construction

This category is formed by special tools that generate wrappers in collaboration with a human expert. These tools usually provide a graphical user interface that allows the wrapper creator to analyze the documents to be processed and to easily design a wrapper.

The *DoNoSe* tool [1] works mainly with plain text documents. The tool allows hierarchical decomposition of the contained data and mapping selected regions of the text to components of the data model. *LiXto* [4] is a fully visual interactive system for the generation of wrappers based a declarative language for the definition of HTML/XML wrappers. Both tools provide a graphical user interface that allows the user with no programming experience to produce the appropriate wrappers.

## 2.7 Logical Document Structure

The above information extraction methods have been based on a direct analysis of the code of HTML documents. We can say that these methods work with the *physical* realization of the documents. The bottleneck of this approach is too tight binding of the wrapper to the HTML code. The nature of HTML allows to achieve the desired document design by various ways that can be arbitrarily combined, which makes the wrappers limited to a narrow set of documents and a short time period. As an answer to these drawbacks, there have been several attempts to describe the documents from a *logical* point of view.

The notion of logical document structure has been introduced by Summers [36] in the context of processing the PDF, PostScript and scanned-in documents and it is defined as a hierarchy of segments of the document, each of which corresponds to a visually distinguished semantic component of the document. Other authors use the notions of *document structure tree* [24] or *document map* [40] in similar sense.

### 2.7.1 Visual Analysis of HTML Documents

There are several approaches to creating the model of the logical structure for HTML documents. They differ in the granularity of the resulting model that depends on its intended application.

The work of Carchiolo et al. [7] deals with the discovery of the logical schema of a web site that contains multiple documents. For this purpose, basic logical sections are localized in each document such as logical heading, logical footer and logical data where the semantic of the page is mainly placed. The proposed approach is based on a bottom-up HTML code analysis. First, collections of similar code patterns are localized in the document. As the second step, each section is assigned a meaning (e.g. logical header) based on the semantics of the HTML tags (e.g. the `<form>` tag denotes an interactive section) or on some information retrieval techniques (e.g. the header section is the collection that refers to the text in the title of the document or to the URI of the page). Similarly, [39] discovers semantic structures in HTML documents. This approach is based on the observation that in most web pages, layout styles of subtitles or data records of the same category are consistent and there are apparent boundaries between different categories. First, the visual similarity of the HTML content object is measured. Then, a pattern detection algorithm is used for detecting frequent patterns of visual similarity. Finally, a hierarchical representation of the document is built. The method described in [29] is based on similar principle. A key observation of this method is that semantically related items in HTML documents exhibit spatial locality. Again, a tree of HTML tags is built and similar patterns of HTML tags are discovered. Finally, a tree of the discovered structures is built.

While the above methods discover the logical structure of the document to the

level of basic semantic blocks, the work of Chung et al. [8] is more oriented to information extraction. Based on the visual analysis, it attempts locate data fields in the documents and store them in an XML representation. It is assumed that the documents being processed pertain to a particular, relatively narrow ontology. Furthermore, certain domain knowledge provided by the user is necessary in the form of *topic concepts* and optional *concept constraints*. Each concept is described by a set of *concept instances* that specify the text patterns and keywords as they might occur in topic specific HTML documents. By contrast, the topic constraints describe how concepts as information carrying object can be structured. As the first step, a majority schema of the document is inferred in the form of a document type definition (DTD). Next, the data fields corresponding to the individual concepts are discovered.

The last described approach is quite different. In [20], a method for the web content structure discovery is presented that is based on modeling the resulting page layout and locating basic objects in the page through projections and two basic operations – block dividing and merging. The projection allows to detect visual separators that divide the page into smaller blocks and again, adjacent block can be merged, if they are visually similar. This dividing and merging process continues recursively until the layout structure of the whole page is constructed.

## 2.8 Logical Document Discovery

The task of the logical document discovery consists of locating all the physical HTML documents that form a logical entity called logical documents. The input is the URI of the *main page* (sometimes also called the *top page* or the *index page*), which is intended by the author of the document to be an entry point to the logical document and it is usually directly accessed by the users[2]. The output is the list of the URIs of all the HTML documents that form the logical document.

The major problem that has to be solved is how to distinguish the documents that belong to the logical document from the remaining ones. There exist two different types of information that can be used for resolving this task:

- *Document classification.* We assume that the individual physical documents that form the logical document are more similar to each other than the remaining referenced documents. The similarity of documents is usually computed using the methods based on the term frequency in the document such as the $tf \cdot idf$ method [33].

- *Link topology analysis.* In general, the topology of the links among a set of HTML documents can be represented as a directed graph. By the analysis

[2]The URI of the main page is usually publicly available, in contrast to the URIs of the remaining documents

of this graph using specific heuristics we can detect a subgraph with certain properties. This technique is also used for detecting so-called *communities* in the web [18]. Additionally, some limitations can be put to the format of the URIs (e.g. all documents must be placed on the same web server, etc.)

Usually, these types of information are used together. Tajima et al. [37] shows that most of the logical documents are organized into hierarchies. The approach is based on the assumption that the authors include some hypertext links to the documents that are intended to be used by the users as a standard way of getting to a particular document. These links form so-called *standard navigation paths*. There are, of course, other links that either point outside of the logical document or have a lower importance such as links back to the main page. The proposed method of logical document discovery has two steps. First, the hierarchical structure is discovered by identifying paths intended by the authors of the documents to be the *standard navigation routes* from the main page to other pages. Then, the discovered hierarchy is divided into sub-hierarchies corresponding to the logical documents based on comparing the document similarity using the $tf \cdot idf$ method.

# 3 Motivation and Goals of the Thesis

Currently, wrappers are mainly used for obtaining the data from various web sites in order to create a service that provides a centralized view of the data from certain domain available in the WWW. Such a service allows a user to effectively use the data available in the web with no need of locating the appropriate web sites, browsing the documents and locating the appropriate data in the documents. Such a service is most frequently offered for the shopping domain – several services for comparing prices of goods in on-line shops are available (e.g. AmongPrices[3] or Compare Online[4]). Another good example is the financial domain – services for comparing stock quotes, exchange rates etc.

Although many techniques for automatic wrapper construction have been proposed, the most used approach is still the manual or semi-automatic wrapper construction. The cause of this situation are following major problems that appear in different extent in the above mentioned wrapper induction approaches:

- *The necessity of wrapper training.* A sufficiently large set of annotated training data is required. The training set preparation is time-consuming; moreover, in some cases no training data is available at all (e.g. when there is only one instance of the page available).

[3] `http://www.amongprices.com/`

[4] `http://www.compare-online.co.uk`

- *Brittleness.* When some change occurs in the documents being processed, the wrapper can stop working properly. The detection of this situation is not a trivial problem [28]. Moreover, new training data must be prepared and the wrapper has to be re-induced.

- *Low precision.* For practical setting of the wrapper, it is necessary that the produced data is reliable – i.e. the values of *precision* and *recall* are close to 100%. The precision of current methods varies between 30 and 80% [9, 12, 17, 25] depending on the used method and the processed documents.

There are several causes of these problems. The most important one is too tight binding of the produced wrapper to the HTML code, which makes the wrapper very sensitive to any minor irregularity in the HTML code. All the above mentioned wrapper induction methods are based on an assumption that there exists some direct correspondence between the HTML code and its informational content. However, the relevance of this assumption is also quite questionable. As stated in the introduction, HTML allows defining a visual appearance of a document, i.e. a presentation and formatting of the contained text. The relation between this formatting and the semantics of the document content is not defined anywhere. Wrappers based on some assumptions regarding this relation that have been defined based on previous analysis of some documents are therefore based on an information whose relevance is not guaranteed anywhere and it can be (and usually it is) limited to a certain set of document and a certain (unpredictable) time period. Moreover, the document can contain various irregularities and special features so that the rules induced for a document needn't to be completely applicable to another document event if it belongs to the same class of documents.

As a solution of the above mentioned disadvantages of the wrapper approach, we propose developing a new method that would fulfill the following requirements:

1. *The documents are analyzed in time of extraction.* The method shouldn't be based on any features of the document or the set of documents that have been discovered in the past. Using information that was relevant at some time in the past can lead to incorrect results because the document may have been changed in the meantime.

2. *Only the features of the logical document being processed are used for information extraction.* Using some knowledge about the features of other documents that are considered to be "similar" to the currently processed one can lead to incorrect results and imprecision.

3. *The method must be independent on the physical realization of the document.* It must be based on the final document appearance that must respect certain rules rather than the underlying HTML/CSS technology that can be chosen

arbitrarily. This includes the situations when the presented information is split to several physical documents. Always the whole logical document should be analyzed.

These requirements solve the problems of the necessity of the training set of documents by analyzing each logical document individually. Moreover, this feature should improve the precision of extraction because the method is not based on the extraction rules inferred for other documents. And lastly, the independence on the physical realization of the logical document should significantly reduce the brittleness of inferred wrappers. The goals of our work can be summarized to following points:

1. To find a suitable model that describes the documents on sufficient level of abstraction and to define this model in a formal way.

2. To propose a new method of information extraction based on the defined model that fulfills the above requirements and resolves the above mentioned problems.

3. To test the proposed method experimentally on real WWW documents.

In following sections, we present the results of our work that have been achieved while fulfilling the above specified goals.

# 4 Visual Modeling Approach to Information Extraction

The principle of the proposed approach is shown in the figure 1. First, we describe the method for a single HTML document. In further sections we extend the method to logical documents that consist of multiple HTML documents.

In contrast to the traditional wrapper approach, the information extraction process consists of multiple steps. As a first step, we analyze the HTML document that contains the data to be extracted and we create a model of the visual information as it is expected to be presented in a standard web browser. This model consists of two separate components – the model of the page layout and the model of the text visual features. As the next step, we transform these two components to a unified model of the document logical structure. This model describes the document content on significantly more abstract level. After creating the logical structure model, the next step is the information extraction itself. Since the logical structure model is a tree, the information extraction task can be formulated as a problem of locating a particular tree node that contains the desired information.

In following sections we give a detailed description of the individual steps.

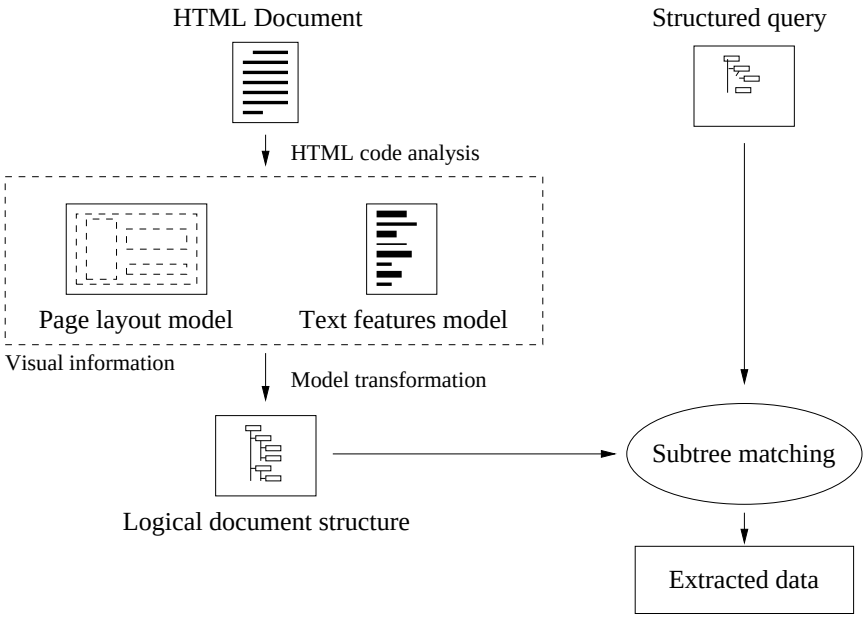


Figure 1: Visual modeling of an HTML document

## 4.1 Visual Information Analysis

The visual information in a document is expressed by the page layout and the attributes of the text. The layout of the page gives the user a basic idea of how the information is organized. Typically, the document is split to several parts, which are visually separated by different background color or by some kind of visual separators. The text attributes such as font or color give more detailed information about the importance and relations among individual parts of the text in a particular area of the document.

We create separate models for both these components. The page layout model $M_l$ describes the hierarchy of the visually separated areas in the documents whereas the text attributes model $M_t$ is used for evaluation of the importance of individual parts of the document text.

### 4.1.1 Page Layout Model

The page layout model is based on the analysis of the HTML code. The main idea of its construction is that there are only a few means for creating a visually distinguished area in HTML. The list of all possibilities is given in table 1. Each of the listed tags can be used for creating a separated area except of the horizontal rule which doesn't form an area itself; it is used to split an existing area in two parts.

The visual areas of the document form a hierarchy where the root node corresponds to the whole document and other nodes correspond to the sub areas that can be arbitrarily nested. Let's assign a unique identifier $v_i \in I$ to each area beginning

| Page object | HTML tags |
|---|---|
| Document | <html> |
| Table, table cell | <table> <th> <td> |
| List, list item | <ul> <ol> <dl> <li> <dt> <dd> |
| Paragraph | <p> |
| Generic area |  |
| Frames | <frameset> <frame> |
| Horizontal rule | <hr> |

Table 1: Visual areas in HTML and corresponding tags

with $v_0 = 0$ for the root area (the document) and assigning $v_i = v_{i-1} + 1$ to each new area encountered while reading the code. The page layout can be modeled as a hierarchy of the area identifiers.

Formally, the model of the page layout can be denoted as a graph:

$$M_l = (V_l, E_l) \tag{3}$$

where $V_l = \{0, 1, \ldots n-1\}$ is the set of all area identifiers and $(v_i, v_j) \in E_l$ iff $v_i$ and $v_j$ are the visual area identifiers and the area identified by $v_j$ is contained in the area identified by $v_i$.

The tables present a special case for the layout modeling. In HTML, the table is a two-dimensional structure but from the logical point of view a table can involve a hierarchical structure that results from the relation between the header and the contents as described in [39]. In this case, this hierarchy must be detected and included to the model of the page layout.

### 4.1.2 Text Attribute Model

An HTML document consists of the text content and the embedded HTML tags. Let's denote $T_{html}$ a set of all possible HTML tags and $S$ an infinite set of all possible text strings between each pair of subsequent tags in a document. Then, an HTML document $D$ can be represented as a string of the form

$$D = T_1 s_1 T_2 s_2 T_3 s_3 \ldots T_n s_n T_{n+1} \tag{4}$$

where $s_i \in S$; $|s_i| > 0$ is a text string that doesn't contain any embedded HTML tags and $n$, $n \geq 0$ is the number of such strings in the document. $T_j$, $1 \leq j \leq n+1$ is a string of HTML tags

$$T_j = t_{j,1} t_{j,2} t_{j,3} \ldots t_{j,m_j} \tag{5}$$

where $t_{j,k} \in T_{html}$ and $|T_j| \geq 1$.

Since the visual attributes of the text can only be affected by the HTML tags, each text string $s_i$ has constant values of the attributes. Let's define the notion of *text element* as text string with visual attributes. Each text element $e_i \in C$, where $C = S \times I \times I \times I$. As usual, we write $e_i$ as

$$e_i = (s_i, v_i, x_i, w_i) \tag{6}$$

and we define

$$e_i < e_j, 1 \leq i \leq n-1, i+1 \leq j \leq n \tag{7}$$

where $s_i \in S$, $v_i, x_i, w_i \in I$. $s_i$ is a text string that represents the content of the element, $v_i$ is the visual area the element belongs to and $x_i$ and $w_i$ are the element $markedness$ and $weight$ which present a generalization of the visual attributes of the text string as described below.

From this point of view, the document with visual attributes can be modeled as a string of the form

$$M_t = e_1 e_2 e_3 \ldots e_n \tag{8}$$

where $e_i = (s_i, v_i, x_i, w_i)$, $1 \leq i \leq n$ are the text elements. $s_i$ corresponds to the appropriate text string in (4). $v_i$ is determined during the tree of visual areas is being built.

The markedness of the element determines how much the element is highlighted comparing to the remaining text of the document. For computing the markedness value $x$ we use a simple heuristic: the font size is adequate to the markedness, the bold font, underlining and color highlighting increases the markedness, whereas the strikeout denotes an insignificant text. The value $x$ can be computed as

$$x = (F \cdot \Delta f + b + o + u + c) \cdot (1 - z) \tag{9}$$

where $b$, $o$, $u$ and $z$ have the value $1$ when the text element is bold, oblique (italics), underlined or striked-out respectively and $0$ if they are not. Similarly, $c$ indicates whether the text element has a color different from the document default. $\Delta f$ is the difference $f - f_d$ where $f$ is the font size of the element and $f_d$ is the default font size for the document in points. The constant $F$ defines the relation between the text size and its markedness. For $F > 4$ the element with greater font size is always more important than the element with lower font size. This corresponds to the usual interpretation.

The element weight expresses the relations of superiority and inferiority among the text elements. We suppose that there exists a hierarchy of headings and labels in the document. The most important heading has the highest weight and the normal text of the document has the lowest weight. For determining the element weight, following heuristics are used:

1. Labels and headings should be highlighted in the text. Therefore the weight of a label or heading is adequate to the markedness of the text element.

2. A bold, underlined or color-highlighted text element placed inside of the continuous text block is not a label. To be considered as a label, such a text element must be placed at the beginning of a text block.

3. Labels are often ended with a colon. Thus a trailing colon increases the weight of the element. An element that ends with a colon should have higher weight that possibly following bold, underlined or color highlighted elements.

Based on the above assumptions, the weight $w$ of a text element is computed as

$$w = [(F \cdot \Delta f) + (b + o + u + c) \cdot l + W \cdot p] \cdot (1 - z) \tag{10}$$

where $F$, $\Delta f$, $b$, $o$, $u$, $c$ and $z$ have the same meaning as in (9), $l$ and $p$ have the value $1$ when the element is preceded by the start of block or a line break and when the element text ends with a colon respectively and $0$ in the opposite case. $W$ is the weight of the trailing colon. The above specification in heuristic 3 is met for $W > 4$.

## 4.2 Logical Document Structure

The notion of logical document structure is used by many authors in quite different ways. In this work we adopt the definition of Chung et al. [8] since their work is based on quite similar observations regarding to the HTML document structure.

In our conception, the logical document structure is a hierarchy of all the text elements in the document where the elements at higher level of abstraction are described by the elements at finer level of abstraction. The level of abstraction of an element is related to its role in the document – the main title of the document has apparently the higher level of abstraction while the normal text in a paragraph has the lower level. Between these two extremes, there lies a hierarchy of other headings and labels.

The model of the logical document structure is based on the transformation of the visual information models. The basic hierarchy is determined by the hierarchy of the visual areas in the document. This hierarchy is further extended by the hierarchy of text elements within each area. Since the logical document structure is a tree of text elements, it can be denoted as

$$S = (V_S, E_S) \tag{11}$$

where $V_S = \{e_1, e_2, \ldots e_n\}$ is the ordered set of all the text elements in the document. The set of edges of the tree $E_S$ is derived from the page layout model $M_l$ (3) and the model of typographical attributes of the text $M_t$ (8). This operation has two phases:

1. Creating the set of graph edges $E_V$ such that $S_V = (V_S, E_V)$ is a tree of text elements and for any $e_i, e_j \in V_S$, $e_i$ is an ancestor of $e_j$ iff the corresponding visual area identifier $v_i$ is an ancestor of $v_j$ in $M_l$.

2. Creating $E_S$ by copying all the edges $(e_i, e_j)$ from $E_V$ and replacing $e_i$ by one of its descendants if needed, so that the element of a higher weight is always an ancestor of all the elements with a lower weight within the visual area.

The resulting model describes the document logical structure as expressed by the visual means.

## 4.3 Information Extraction from the Logical Structure

Once the logical document structure is known, the information extraction task consists of locating the nodes of the tree $S$ that contain the desired data. This task is quite similar to the activity of a reader when looking for some information in the document.

Firstly, the user expects some format of the desired information. For example, when looking for price we are trying to find a number preceded or followed by a code of currency. Further, the reader navigates through the tree of the logical document structure starting at the root node (the title of the document) and tries to choose the path leading to the desired information by classifying individual headings and labels that in our case correspond to the nodes of the structure tree. The last important information is provided by the relations among the nodes. For example, a postal code in an address can be identified by its relative position to the country name.

A normal user can use the natural language understanding for choosing the best path. However, for a range of tasks the classification can be simplified. Let's consider following simple example. From a personal page, we want to extract the name, department and the e-mail address of that person. The expected structure of this information is in the figure 2. The left tree expresses that the name of the person is usually presented as a superior text (heading) and the remaining data is placed further in the document. We extend the tree by replacing the fields with the regular expressions that denote their expected format and we add the expressions that denote expected labels of these data. The resulting tree can be viewed as a structured query and tree matching algorithms can be used for identifying all matching subtrees of the logical document structure.

For tree matching, we use a modification of the **pathfix** algorithm published by Shasha et al. [34]. This algorithm solves approximate searching in unordered trees based on the root-to-leaf path matching. The original problem is to find all data trees $D$ from a database $\mathcal{D}$ that contain a substructure $D'$ which approximately

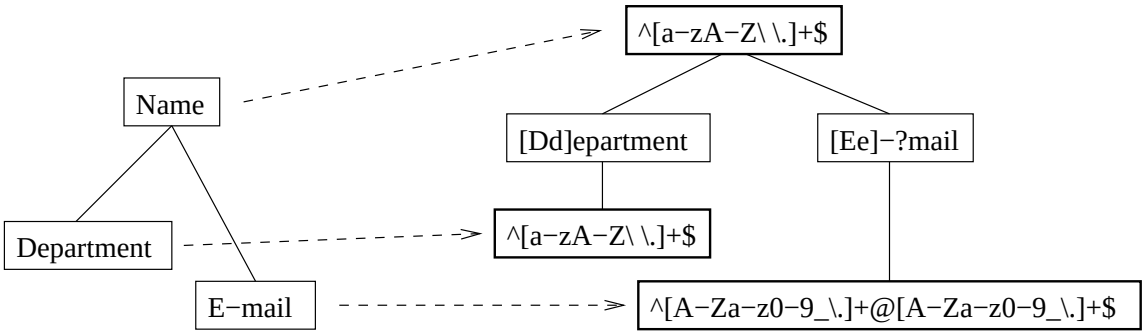


Figure 2: An expected information format and the corresponding query tree with labels and desired data (bold boxes) specified by regular expressions

matches a query tree $Q$ within distance $DIFF$. The distance measure is defined as the total number of root-to-leaf paths from $Q$ that do not appear in $D'$.

In our case, the database $\mathcal{D}$ is formed by all the subtrees of the logical document structure $S$ with the root node that matches the root node of the query tree. The incorporation of regular expressions to this algorithm is straightforward: a node in the logical structure tree matches a node in the query tree iff the appropriate text element matches the regular expression in the query node. Since the exact form of the logical document structure may differ in some range among the web sites, we have extended the path matching algorithm by allowing a number of $QSKIP$ nodes from the query path that don't match any node in the matched path and $MSKIP$ nodes in the matched path that don't match any node in the query path. The accuracy of the results depends on the values of these parameters and on the allowed number of missing paths $DIFF$.

## 4.4 Logical Document Discovery

The extension of our information extraction method from physical to logical documents is straightforward. For discovering the logical documents, we adopt the method published in [37], which is also discussed in the section 2.8. The information extraction process consists of following steps:

- We consider the document whose URI has been specified by the user the URI of the index page of the logical document. We use the method [37] for obtaining the URIs of the remaining physical documents.
- We create the models of the logical structure for each physical document separately using our method proposed in the above sections.
- We create a single model of the logical structure for the whole logical document by joining the logical structure trees of the physical documents to a single tree. We take the logical structure of the main page and we replace

the nodes that correspond to the links to other documents with a subtree that corresponds to the logical structure of the corresponding document. This step is applied recursively to all the subtrees.

After finishing this process, we have created the model of the logical structure for the whole logical document. This model can be used for information extraction as specified in section 4.3. We can see that this approach abstracts from the physical organization of the information to documents by creating an unified model for all the physical documents.

# 5 Experimental Results

We have tested the proposed method on the simple extraction task shown in Figure 2. As a data source we have used sets of staff personal pages from various universities. Table 2 shows values of *precision* (1) and *recall* (2) as defined in section 2.3. The only input for the information extraction is the URI of the document and the query tree, which has been identical for all the data sets.

The results proof that the method doesn't depend on a particular HTML code (see the sets 1, 2 and 5 in the table). There are, however, three basic reasons that may cause the extraction process to fail:

- The extracted data is not labeled as expected; e.g. in the testing set 3 the e-mail addresses are not denoted by any label
- The data to be extracted is contained inside of larger text elements; e.g. the appropriate data at `mit.edu` (4) is presented as unformatted text only
- Various anti-spam techniques used in the documents such as writing e-mail addresses in some non-standard form or presenting is as an image (anti-spam precautions)

During the tests, the parameters have been set to $QSKIP = 0$ and $MSKIP = 1$. Increasing $QSKIP$ and $MSKIP$ can improve the recall in case that the format of the data fields is specific enough; e.g. the e-mail address can be discovered by its format only so that it is not necessary to require any label. Generally, increasing these values causes a significant loss of precision.

# 6 Summary of Contributions

Proposed method presents a novel approach to information extraction from HTML documents. In contrast to current methods based mainly on the direct analysis of the HTML code, our method has following important features:

| # | URI | Precision % | Recall % |
|---|---|---|---|
| 1 | fit.vutbr.cz | 91 | 100 |
| 2 | mff.cuni.cz | 100 | 100 |
| 3 | is.muni.cz | 100 | 50 |
| 4 | mit.edu | - | - |
| 5 | cornell.edu | 100 | 100 |

Table 2: Sample extraction task results

- *Independence on the underlying HTML code of the document.* The document is described by an abstract model, which is then used for extracting information. This abstraction avoids the dependence on particular HTML tags, which is the bottleneck of the wrapper approach.

- *Resistance to the changes of documents.* The use of abstract model ensures that the method is resistant to changes in the data presentation in the document unless the logical structure of the document changes.

- *No training phase required.* The information extraction process can start as soon as the extraction task specification is finished. There is no training set of example documents needed. The method allows processing new, previously unknown documents that correspond to the extraction task specification.

Aside from this main contribution, there are some issues in the method proposal that we consider a significant or novel contributions:

- *Formal models of the visual information in the document.* To the author's best knowledge, this is a **first attempt to formally describe the information that is given to the user by visual means**. We propose formal models of two components of this information – the page layout and the visual attributes of the text.

- *Modeling the logical structure on the basis of the visual model.* This approach is unique for the processing of HTML documents although similar idea has been proposed for other types of documents. The model of the logical structure presents an important information about the document that can be used (aside from information extraction) in many other areas such as *information retrieval* (searching documents based on structured queries instead of single keywords) or *alternative document presentation* (e.g. structure-aware voice readers for blind people).

- *Application of the tree matching algorithms.* Although the hierarchical organization of HTML documents is commonly accepted, the use of tree matching algorithms for this task can be considered a novel contribution to this area.

Finally, the contribution of the thesis is an experimental information extraction system that implements the proposed techniques. This system has been implemented in the Java environment and has been used for verifying the method in the real-world.

# 7 Conclusions

Proposed method of information extraction is usable for real data-intensive documents available through the World Wide Web. With data-intensive documents we mean the documents that are primarily intended for presenting data in a relatively regular and structured form where the visual information plays an important role for the readers' understanding of the document. These documents often contain up-to-date data that are worth extracting and typically, the documents are automatically generated from a back end database. On the other hand, the method is not suitable for processing the documents where the desired information is buried in large blocks of unformatted or poorly formatted text. For such documents, the traditional text document processing and natural language processing methods are more applicable.

There is one more important issue in information extraction as such that has not a technical nature. Quite often, the provider of the information that is presenting it on the web prefers browsing the documents by people rather than the automatic tools, mainly for marketing reasons (advertisement etc.). Such subjects often use various techniques that complicate automatic processing of the documents such as detection and blocking of the client or "hiding" the information in the document. There are, however, still many areas where the information extraction is justifiable and useful.

# References


[1] Adelberg, B. NoDoSe – A Tool for Semi-Automatically Extracting Structured and Semistructured Data from Text Documents. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data* Seattle, Washington, United States, 1998

[2] Ashish, N., Knoblock, C. Wrapper Generation for Semi-structured Internet Sources. In *Workshop on Management of Semistructured Data*. Tucson, Arizona, 1997

[3] Atzeni, P., Mecca, G., Merialdo, P. Semistructured and Structured Data in the Web: Going Back and Forth. In *Proceedings of ACM SIGMOD Workshop on Management of Semi-structured Data*. 1997

[4] Baumgartner, R., Flesca, S., Gottlob, G. Visual Web Information Extraction with Lixto. In *Proceedings of the 27th International Conference on Very Large Data Bases*. Roma, Italy, 2001

[5] Berners-Lee, T. The Semantic Web. *Scientific American*. May 2001

[6] Buttler, D., Liu, L., Pu, C. A Fully Automated Object Extraction System for the World Wide Web. In *Proc. of IEEE International Conference on Distributed Computing Systems.* 2001

[7] Carchiolo, V., Longheu, A., Malgeri, M. Extracting Logical Schema from the Web", In *PRICAI Workshop on Text and Web Mining.* Melbourne, Australia, 2000

[8] Chung, C.Y., Gertz. M., Sundaresan, N. Reverse Engineering for Web Data: From Visual to Semantic Structures. In *18th International Conference on Data Engineering (ICDE 2002).* IEEE Computer Society, 2002.

[9] Ciravegna, F. $(\mathrm{LP})^2$, an Adaptive Algorithm for Information Extraction from Web-related Texts. In *Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining.* Seattle, USA, 2001

[10] Cohen, W.W., Hurst, M., Jensen, L.S. A Flexible Learning System for Wrapping Tables and Lists in HTML Documents. In *Proceedings of the Eleventh International World Wide Web Conference.* Honolulu, Hawaii, USA, 2002

[11] Crescenzi, V., Mecca, G., Merialdo, P. *RoadRunner: Towards automatic data extraction from large web sites.* Technical Report n. RT-DIA-64-2001, D.I.A. Università di Roma Tre, 2001

[12] DiPasquo, D. *Using HTML Formatting to Aid in Natural Language Processing on the World Wide Web.* School of Computer Science, Carnegie Mellon University, Pittsburgh, 1998

[13] Embley, D.W., Campbell, D.M., Jiang, Y.S., Ng, Y.-K., Smith, R.D., Liddle, S.W., Quass, D.W. A conceptual-modeling approach to extracting data from the web. In *Proc. of the 17th International Conference on Conceptual Modeling (ER'98).* Singapore, 1998

[14] Embley, D.W., Jiang, Y.S., Ng, Y.-K. Record-boundary discovery in Web documents. In *Proc. of the 1999 ACM SIGMOD International Conference on Management of Data* 1999

[15] Fielding, R., et al. *Hypertext Transfer Protocol – HTTP/1.1*. RFC2616, The Computer Society, 1999. `http://rfc.net/rfc2616.html`

[16] Freitag, D. Using Grammatical Inference to Improve Precision in Information Extraction. In *ICML-97 Workshop on Automata Induction, Grammatical Inference, and Language Acquisition.* 1997

[17] Freitag, D. Information extraction from HTML: Application of a general learning approach. In *Proc. of the Fifteenth Conference on Artificial Intelligence AAAI-98.* 1998

[18] Gibson, D., Kleinberg, J., Raghavan, P. Infering Web Communities from Link Topology. In *Proc. of 6th Intl. Conference on Database Systems for Advanced Applications (DASFAA'99).* IEEE Computer Society, 1999

[19] Gold, E.M. Language Identification in the Limit. *Information and Control*, 10(5):447-474. 1967

[20] Gu, X.-D., Chen, J., Ma, W.-Y., Chen, G.-L. Visual Based Content Understanding towards Web Adaptation. In *Proc. Adaptive Hypermedia and Adaptive Web-Based Systems.* Malaga, Spain, 2002, pp. 164-173

[21] Guan, T., Wong, K.F. KPS – a Web Information Mining Algorithm. In *The 8th International World Wide Web Conference*. Toronto, Canada, 1999

[22] Güttner, J. Object Database on Top of the Semantic Web. In *Proceedings of the WI/IAT 2003 Workshop on Applications, Products and Services of Web-based Support systems.* Halifax, CA, 2003, pp. 97-102

[23] Hong, T.W., Clark, K.L. Using Grammatical Inference to Automate Information Exraction from the Web. In *Principles of Data Mining and Knowledge Discovery.* 2001

[24] Kan, M.-Y. *Combining visual layout and lexical cohesion features for text segmentation.* Columbia University Computer Science Technical Report, CUCS-002-01. 2001

[25] Kosala, R., Van den Bussche, J., Bruynooghe, M., Blockeel, H. Information Extraction in Structured Documents using Tree Automata Induction, In *Principles of Data Mining and Knowledge Discovery, Proceedings of the 6th International Conference (PKDD-2002).* 2002

[26] Kushmerick, N., Weld, D.S., Doorenbos, R.B. Wrapper Induction for Information Extraction, In *International Joint Conference on Artificial Intelligence*. 1997

[27] Kushmerick, N. Wrapper Induction: Efficiency and Expressiveness. *Arfiticial Intelligence* vol. 118, no. 1-2, pp. 15-68, 2000

[28] Kushmerick, N. Wrapper verification. *World Wide Web Journal* vol. 3, no. 2, pp. 79-94, 2000

[29] Mukherjee, S., Yang, G., Tan, W., Ramakrishnan, I.V. Automatic discovery of Semantic Structures in HTML Documents. In *International Conference on Document Analysis and Recognition (ICDAR).* 2003

[30] Nahm, U.Y., Mooney, R.J. Text Mining with Information Extraction. In *Proceedings of the AAAI 2002 Spring Symposium on Mining Answers from Texts and Knowledge Bases.* 2002.

[31] Quinlan, J.R., Cameron-Jones, R.M. FOIL: A Midterm Report, *Machine Learning: ECML-93*, Vienna, Austria, 1993

[32] Raggett, D., Le Hors, A., Jacobs, I. (editors). *HTML 4.01 Specification* W3C Recommendation 24 December 1999. `http://www.w3.org/TR/1999/REC-html401-19991224`

[33] Salton, G. Recent Studies in Automatic Text Analysis and Document Retrieval. *JACM*, 20(2):258-278, Apr. 1973

[34] Shasha, D., Wang, J.T.L, Shan, H., Zhang, K. ATreeGrep: Approximate Searching in Unordered Trees. In *14th International Conference on Scientific and Statistical Database Management*. Edinburgh, Scotland, 2002

[35] Soderland, S. Learning to Extract Text-based Information from the World Wide Web, In *Proceedings of Third International Conference on Knowledge Discovery and Data Mining (KDD-97).* 1997

[36] Summers, K. Toward a Taxonomy of Logical Document Structures. In *Electronic Publishing and the Information Superhighway: Proceedings of the Dartmouth Institute for Advanced Graduate Studies (DAGS '95).* Boston, USA, 1995, pp. 124-133

[37] Tajima, K., Tanaka, K. New Techniques for the Discovery of Logical Documents in Web. In *International Symposium on Database Applications in Non-Traditional Environments (DANTE'99).* 1999

[38] World Wide Web Consortium (W3C) pages. `http://www.w3.org/`

[39] Yang, Y., Zhang, H. HTML Page Analysis Based on Visual Cues. In *Proc. of 6th International Conference on Document and Analysis.* Seattle, USA, 2001

[40] Zizi, M., Lafon, M. Hypermedia exploration with interactive dynamic maps. *International Journal on Human Computer Interaction Studies*. 1995

# Author's Publications


1. Antoš, R., Burget, R., Jelen, P. Mathematical Data Output Support. In *Proceedings of 7th Conference Student FEI 2001*. Brno, CZ, FEI VUT, 2001, p. 274-276, ISBN 80-214-1859-1

2. Bílek, R., Burget, R., Haša, L., Kutálek, V., Mika, D., Petřek, J., Řezáč, D., Staroba, J., Zacios, D. *Preliminary course for IT study.* Brno, CZ, FIT VUT, 2002, p. 69

3. Burget, R. Analyzing Logical Structure of a Web Site. In *Proceedings of 5th International Conference ISM '02 - Information Systems Modelling*. Ostrava, CZ, MARQ, 2002, p. 29-35, ISBN 80-85988-70-4

4. Burget, R. HTML Document Analysis for Information Extraction. In *Proceedings of 8th EEICT conference.* Brno, CZ, FIT VUT, 2002, p. 426-430, ISBN 80-214-2116-9

5. Burget, R. Information Extraction from WWW Based on the Data Structure Knowledge. In *Sborník příspěvků 2. ročníku konference Znalosti 2003* Ostrava, CZ, FEI VŠB, 2003, p. 271-280, ISBN 80-248-0229-5

6. Burget, R. Hierarchies in HTML Documents: Linking Text to Concepts. Accepted for the *3rd International Workshop on Web Semantics - WebS '04*.

7. Burget, R. Information Extraction from HTML Documents Based on Visual Modeling. Submited to *Jorunal of Information Retrieval, Special Issue on Web Information Retrieval*.

# Author's Curriculum Vitae

Name: Radek Burget
Born: Brno, Czech Republic, 1978

Education: Brno University of Technology, Czech Republic (1996-2004): Faculty of Electrical Engineering and Computer Science (1996-2001), M.Sc. in Computer Science and Engineering (2001), PhD student at the Faculty of Electrical Engineering and Computer Science (2001) and the Faculty of Information Technology (2002-2004).

Participation at EDBT 2002 Summer School: *Distributed Databases on the Net: Models, Languages and Infrastructures*, August 26-31, 2002, Cargèse, Corsica, France

Study stay: Universidad de Valladolid, Spain, Socrates/Erasmus grant (October 2002 - January 2003)

Teaching: Computer exercises of the courses of *Algorithms and Programming*, *Programming Practice* and *Operating Systems*. Individual project correction in the courses of *Operating Systems* and *C and C++ Programming Languages*.