
**BRNO UNIVERSITY OF TECHNOLOGY**
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**FACULTY OF INFORMATION TECHNOLOGY**
**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**

**DEPARTMENT OF INFORMATION SYSTEMS**
**ÚSTAV INFORMAČNÍCH SYSTÉMŮ**


# INFORMATION EXTRACTION FROM THE WEB USING VISUAL PRESENTATION ANALYSIS

**EXTRAKCE INFORMACÍ Z WWW S VYUŽITÍM ANALÝZY VIZUÁLNÍ PREZENTACE**

**HABILITATION THESIS**
**HABILITAČNÍ PRÁCE**


**AUTHOR** **RADEK BURGET**
**AUTOR PRÁCE**



**BRNO 2020**

# Abstract

Today's World Wide Web is a widely used platform for publishing and sharing information from a wide range of areas. Due to the nature of the Web, information is typically presented in loosely structured documents, limiting the ways in which it can be accessed to simple browsing and reading documents. Computer support for information retrieval on the web is then practically limited to keyword-based indexing and full-text search. Automatic extraction of the information contained in web documents would allow its better accessibility (for example by contextual search queries) or its integration with structured databases and information systems. However, despite the progress made in this area in recent years, it still represents a challenging research problem.

The main feature of the web that makes it especially difficult to automatically process published information is that web documents are primarily designed to be interpreted by human readers. This primary objective is also reflected in the technological implementation of the documents that focuses on aspects of organization and visual presentation of information without further computer processing in mind. Therefore, the documents typically do not contain explicit information on the semantics of the data presented. In order to recognize and explicitly represent the information contained in documents, current approaches are mostly based on the analysis of the document code using various combinations of manually defined and automatically inferred rules and heuristics. More recent research, however, shows that visual cues provided to human readers can provide more reliable information about the author's understanding of the presented content also for computer processing. For their use, however, it is necessary to develop comprehensive document models that capture all aspects of content, its organization and presentation, as well as methods for further analysis and interpretation of these models.

This thesis summarizes my research performed so far in the related sub-tasks of information extraction from web documents based on the analysis of visual content presentation. These tasks include document pre-processing depending on source format and implementation, detection of basic information blocks with page segmentation methods, overall logical structure analysis, discovery of structured data records and the integration of the extracted data to domain-focused applications. The focus on visual presentation analysis allows application of the developed methods to heterogeneous document sets and differentiates our approach from most state-of-the-art methods. As I believe, the approaches and methods developed together contribute to efforts to make information buried in web documents accessible to computer applications and ultimately to make it easier to find and use.

# Acknowledgments

I would like to thank my co-authors for the excellent cooperation leading to the published results: Jan Zelený, Martin Milička and especially Jaroslav Zendulka, who has been not only a co-author but above all my valuable mentor throughout my academic career. I also greatly appreciate the support of other members of the Department of Information Systems, in particular Dušan Kolář and Tomáš Hruška, who supported me in my work. Last but not least, I would like to thank my whole family for their support, love and patience.

# Contents

# Chapter 1

# Introduction

World Wide Web is a popular information source widely used to publish information from a wide range of areas. It is made up of a huge number of interlinked documents with different content, structure and in diverse formats. Although it was originally designed mainly as a technical and scientific document sharing platform [19], today's web has much wider use. In addition to providing a simple access to published documents, it is often used as an interface for accessing different types of applications including online databases.

As given by the design and purpose of the web, a document represents the smallest unit of information that can be uniquely identified by its URI. No further restrictions are applied to the internal structure and contents of the documents themselves which allows great flexibility in publishing any type of information. On the other hand, this feature limits the access to the presented information to simple browsing using a web browser and interpretation of the displayed documents by human readers. Existing search engines facilitate the discovery of resources related to given keywords by analyzing the full text of the documents and building their indexes; final consideration of the context of the keywords in the discovered documents and finding the desired information in them still requires human interpretation.

Given the vast and growing number of available documents and the amount of information potentially available, this limitation in the information access was recognized soon after the web started to be used in larger scale [13] and the possibilities of more formal representation and access to the information on the web have been subject of research for more than twenty years. Inspired by existing database systems, many query languages have been proposed for querying the web [73, 79, 91]; however, the documents were still just treated as unstructured objects organized in a graph [17]. The next step would be to provide more fine-grained access to the information available inside the documents, which would allow efficient indexing and querying the contained data and its integration with other data sources such as structured databases and information systems. To achieve this goal, it is necessary to acquire more structured and formal representation of the document content. Due to the nature of current technology used in web design and common ways of its use, this is still a challenging research problem.

Most of text documents on the web are written in the Hypertext Markup Language (HTML); other document types such as PDF are commonly used as well. These documents are often characterized as *semistructured documents* [17, 96] as opposed to unstructured (plain) text documents at one side and fully structured data stored for example in relational databases on the other side [89]. Their text content is usually divided into smaller, hierarchically organized logical blocks that represent the overall structure of the document.

However, this structure is very loose, it does not follow any pre-defined schema and it provides only a basic information about the individual parts of the content. This corresponds to the primary objective of web design, which is making the presented information easily understandable and usable for human readers. This is typically achieved by a carefully designed visual organization and presentation of the contents on the displayed page. In other words, the focus is given on presentational aspects of the contents and the document creators very rarely provide any kind of explicit machine-readable annotations that would allow the integration of the presented data to computer applications.

The research efforts in the area of web data source modeling and integration try to bridge the gap between the human-centric, loosely structured web pages and the computer applications (based on more structured data models) generally in two ways:

- The development of new technology that allows the document creators to include semantic annotations about the contained information and its structure directly in the documents. Several extensions of the HTML language have been proposed that allow to link the elements of the document contents with different ad-hoc dictionaries such as Microformats [67], HTML Microdata [23] or in a more general way with semantic web resources using for example RDFa [61].

- The *information extraction* approach that consists on analyzing the state-of-the-art input documents in the same form as they have been published on the web and transforming them to a form that is more suitable for further processing and integration to computer applications [74].

The first approach is technically cleaner and easier. However, it requires an active involvement of the web document creators that are often not motivated to add work by carefully adding additional semantic annotations to their documents. As the result, the usage of semantic annotations is still not very widespread; the statistics from the Common Crawl corpus[1] show that some kind of semantic annotations was present in 37.1 % of web pages (out of 2.1 billion pages contained in the data set) or 29.3 % of domains (out of 32.8 million) in November 2019 [21].

Therefore, information extraction from plain (mostly HTML) documents is most often the only available option for accessing the contained data. Despite the long and extensive research that has been devoted to the automation of this task, current practice in web information extraction still consists mainly of creating specialized procedures (traditionally called *wrappers*) [107]. These procedures are typically written in a general-purpose programming language and tailored for a particular data source (a single web site or even every document)[2]. This typically results in maintaining large sets of wrappers that must be periodically updated manually as the source documents change and evolve.

Current research in this area (that is gradually cited in subsequent chapters) focuses on the development of more general information extraction methods, that are not tightly bound to a single input document and the way of its implementation. One of the possible solutions is to focus on the visual organization and presentation of the content that is provided by the document author to the document reader to facilitate understanding of the content. As mentioned above, this is the primary goal of web design and is therefore

[1] http://www.commoncrawl.org/

[2] Let us mention for example the Apify service (https://apify.com/) as an example of a state-of-the art approach used in large scale. It allows users (or domain experts) to create "scrapers" that are written mostly in JavaScript separately for each specific web source and later executed for extracting the data.

given much more attention than the document code. In the same time, it must follow some commonly accepted rules and patterns in order to be comprehensible to the target readers. As such, visual presentation is a promising source of additional information about published content that is potentially less volatile than the underlying document code and is therefore applicable to larger heterogeneous sets of documents.

In order to use the visual presentation for information extraction, it is necessary to develop methods and algorithms that analyze the complete available information in the documents in order provide as accurate as possible estimate of the semantics of the individual parts of the document content. This information includes both the document code (the implementation aspect) and the resulting visually presented contents of the document (the result of the author's intentions regarding the content presentation). Along with additional background knowledge such as the knowledge of commonly used language and presentation patterns and possibly knowledge of the subject domain, these methods may estimate the probability that a human reader would assign a particular meaning to the given part of the document. Depending on the particular task, this estimate can be made at different granularity levels, from identifying individual content blocks (such as the page header, footer, navigation and main article) to recognizing structured data records and individual data fields that may contain, for example, product or personal data. As a next step, related parts of the document content may be transformed into a structured form and integrated into other applications.

My research performed in collaboration with my colleagues that is summarized in this thesis focuses on the related information extraction approaches with the ultimate goal of making the information buried in countless diverse web pages available for different web applications with as little human effort as possible. More specifically, all my research efforts described further aim at automatically recognizing and extracting specific content from web documents, which includes the main content of the page (as for example the published articles) and, above all, structured data records. A common feature of my research in this area is the emphasis on using available information about the visual presentation of content in documents to achieve the greatest possible independence on the technical realization and implementation details of the input documents. This makes the proposed methods different from most already published works. Details, benefits of the proposed approaches and comparison with the current state of the art are discussed in the individual chapters.

## 1.1 The Information Extraction Task in the Web Context

Information extraction (IE) as a general task is not specific for Web environment. It has been extensively studied in the context of processing and understanding continuous text documents in English and other human languages based mainly on natural language processing methods (NLP) [41]. In the context of web documents, the NLP methods are used as well as for example named entity recognition (NER) [29, 82]. However, compared to plain text documents, the information extraction task from web (mainly HTML) documents has certain specifics:

- HTML documents contain additional markup that divides the document to individual *elements* that are hierarchically organized [147]. A visual style may be defined for the individual elements which allows to create a rich visual presentation of the content. Similarly for PDF and other styled document formats, the content is divided to separate parts with visual style assigned.

- Structured information on the web is typically not presented as a continuous text in English or other language. Instead, a structured presentation using lists or tables is more common [37, 60]. As the result, the content to be extracted is typically not presented in full sentences and the advanced natural language processing methods may not be entirely applicable.

On a closer look, the entire information extraction task from web documents involves a number of separate research problems that include but are not limited to the following:

- Design of suitable models of the input documents that would allow to abstract from irrelevant implementation-related details while preserving the important structure- and presentation-related aspects of the documents.
- Analysis of the document structure that includes the discovery of separate content blocks and their possible mutual relationships.
- Content classification for distinguishing the main content from auxiliary parts such as navigation or advertisement or even for assigning more fine-grained roles to the individual parts of the content.
- Discovery of repeating presentation or code patterns that may indicate regular data records.
- Alignment of the assumed data records with existing data models in order to integrate the extracted data into existing applications.

The order of these points corresponds to the usual way of processing web documents when extracting specific information. First, the input documents must be represented by models suitable for further processing. Subsequently, their overall structure is analyzed and finally, the relevant parts or even individual data records are located and extracted. In my research done so far, I have gradually dealt with all the mentioned problems, which is reflected by the structure of this thesis.

## 1.2 Structure of the Thesis

The order of chapters corresponds to the individual document processing steps described above. In chapter 2, new document models are introduced that allow capturing both the document content and the information about its visual presentation. These models are later used for subsequent steps of document processing. Section 3 focuses on page segmentation which is an important document pre-processing step. In chapter 4, we discuss the methods of classification and extraction of larger specific content object such as the main document content (for example published articles) and its individual parts. Finally, chapter 5 is devoted to the identification and extraction of fine-grained structured data records from the documents. All sections provide a summary and context of the proposed methods and achieved results. The most important part of the thesis is formed by journal and conference papers in Appendices A to C. Appendix D describes the most important software tools developed during the research mostly with the purpose of evaluation the proposed methods.

# Chapter 2

# Web Document Modeling

With a document model, we understand a representation of an input document suitable for further analysis by information extraction methods. The choice of the model depends on the types of information available in the input document and the methods of its further processing.

Standard web pages are mostly created using some version of the Hypertext Markup Language (HTML) [44] that allows to define the document content and its structure, which includes the differentiation of basic document sections, headings, paragraphs and other commonly used content parts. Visual presentation of the content is typically defined separately from the HTML code using Cascading Style Sheets (CSS) definitions [22]. Both the HTML and CSS languages represent a formally defined web standard maintained by the World Wide Web Consortium (W3C)[1]. For this reason, most existing information extraction methods mentioned further assume HTML documents on their input and they use document models that are HTML-specific.

Documents that cannot be easily represented or converted to HTML (such as complex documents with other than web origin) may be available in other formats; the Portable Document Format (PDF) [1] is often used for this purpose.[2] Unlike HTML, PDF does not explicitly describe the content structure; it focuses on a detailed representation of the displayed or printed page at the printer instruction level and therefore, the reconstruction of the complete content and its structure may be a challenging problem [60]. However, some document models may be generalized even for PDF documents as we mention in section 2.4.

In this chapter, we describe the state-of-the-art, mainly HTML-specific models in section 2.1. In the next sections, we introduce advanced models that we designed for various applications as a part of our research work.

## 2.1 Related Work

Early approaches to information extraction represent the documents as unstructured text strings containing the document text combined with HTML markup [16, 113] or sequences of tokens that represent the individual words and tags [36, 52, 63]. When using these flat models, information extraction from web pages may be performed based on the recognition of substrings or token sequences that are used for delimiting the relevant data fields in the

[1] https://www.w3.org/

[2] PDF is standardized as an ISO 32000-2 standard

source document. A set of such delimiters discovered for a particular document defines a *wrapper*, which is a procedure that performs the extraction itself [78]. The delimiters themselves may be discovered manually; however, most of the research work focuses on an automatic inference of the wrappers from a set of sample documents in order to reduce the labor intensity and increase their reliability. This approach is commonly referred to as *wrapper induction* [52, 63, 78, 74].

The advantage of these simple models is that the input document may be processed sequentially using simple algorithms with a very low time and space complexity. On the other hand, HTML documents naturally exhibit a hierarchical structure that may be used to provide an additional context to wrappers and increase the extraction precision [74]. Therefore, many wrapper induction approaches use hierarchical document models that represent the nesting of HTML elements in the document code [37, 48, 74]. Currently, almost all algorithms that expect HTML documents on their input use the Document Object Model (DOM) as the document representation [90, 124, 105, 120]. DOM is a widely used W3C standard that represents the HTML or XML documents as a tree of *nodes* (objects with a standardized interface) that represent the whole document (the root node), the individual HTML elements, their nesting and contents [65].

The code-oriented document representations (which means almost exclusively DOM in current methods) present a solution that is standardized and widely implemented in web browsers and other software tools and libraries. However, they describe the document on the implementation level and although some rough estimate about the purpose of the individual parts of the document may be made from the usage of certain HTML tags, the HTML code itself actually does not contain a complete information about how the content would be finally presented to the user [72, 135, 145]. To get a full picture, the code must be interpreted by a web browser while considering additional information such as the attached CSS definitions [14]. Since the visual presentation is an important cue provided to the reader that is potentially useful for the approximation of the document structure, a logical next step in document modelling is to enrich DOM with additional information about the visual style of the individual elements which typically includes font properties, colors, element sizes and spacing [9, 14, 34, 49, 84, 85, 129, 135]. This approach preserves the simplicity of DOM while still limiting the applicability of the methods to HTML documents only.

The opposite approach is to represent the input documents as images, which allows to extend the set of processed documents to any format. Image processing algorithms may be later used for a precise segmentation documentation such as edge detection and Bayesian methods [40, 39] or graph grammars [72] applied on web pages or even convolutional neural networks [136] applied on PDF documents.

In order to combine the simplicity of the hierarchical DOM model that allows a direct access to the document text content and its structure with the possibility of using a complete visual information, we have proposed more abstract document models that are described in the following sections. Our models represent the document content as a tree or a generic graph of abstract data structures that are not directly bound to the document code. This allows to represent input documents in any format; however, we assume mainly HTML and PDF documents in the applications presented further. Similar approach has been later used by other authors for processing PDF documents as well [54, 106].

## 2.2 Visual Document Model

Our proposed visual model of documents [25, 26, 31] is used mainly in the page segmentation methods for representing both the input document as well as the result of the segmentation. Details are discussed in Chapter 3. Another application is the extraction of semantic objects described in Chapter 4. We have designed the model with the following goals in mind:

- Technology independence – the model should be independent on the technology used for implementing the source documents (for example HTML or PDF) and the way of its usage (for example the particular choice of used HTML tags).
- Representation of complete visual information – the model should represent the document content together with the complete available information about the visual presentation of the individual parts of the content such as the text size, position, fonts, colors and all available visual properties.
- Extensibility – as mentioned above, the model is used in different stages of the document processing to represent both the input document as well as the results of the segmentation and/or object classification. It is therefore assumed that during the document processing, new derived information is added to the model.

Unlike the code-oriented models mentioned in the previous section, the goal of the visual model is to describe how the information is *visually presented to the user* rather than how it is implemented using a particular markup language.

The model consists of two levels of abstraction:

1. *Box model* – represents a general rendered page on the lowest level of abstraction. The model itself is independent on the input document format but it is directly created by rendering the page by the corresponding (HTML or PDF) renderer.
2. *Visual area model* – describes the page layout – the division of the page to individual, visually separated and potentially nested visual areas. It represents the product of page segmentation as described later in chapter 3.

The box model represents the document contents as a set of *boxes*:

$$B = \{b_1, b_2, b_3, \ldots b_n\} \tag{2.1}$$

where each box represents a rectangular area in the rendered page with a given position and size containing an arbitrary part of the document content. For HTML documents, the box creation is defined in detail by the Cascading Style Sheets specification [22]. Typically, a box is created for each visible HTML element; some elements may generate multiple boxes. For PDF documents, the boxes are reconstructed by interpreting the corresponding PDF operators as defined in the specification [1] (e.g. a sequence of a *Tf* operator for setting the font properties and a *Tj* operator for displaying a text string with the selected font).

A box is then defined as a tuple:

$$b = (type, x, y, width, height, bgcolor, fontsize, weight, style, border, text) \tag{2.2}$$

where *type* denotes the type of content (text, graphics or a general HTML element), $x$, $y$, *width* and *height* define the position and the size of the box in the rendered page in pixels.

The background color (*bgcolor*) is represented as a RGB value or a special *transparent* value, if there is no background color defined for the given box. The *weight* and *style* properties have the value of 0 for normal font and 1 for bold or italic font respectively.

Very often, several boxes overlap in the page in order to create a more complex visual structure. For example, a box creates a rectangular area filled with a background color and other boxes create the text displayed inside of the area. Therefore, we define a *box tree* that represents the visual nesting of the boxes in the rendered page as follows:

$$T_b = (B, E_b) \tag{2.3}$$

where $B$ is the set of boxes defined in (2.1) and $E_b$ is the set of edges that represent the nesting; $\forall b_1, b_2 \in B : (b_1, b_2) \in E_b$ iff $b_2$ is visually enclosed in $b_1$ (given their coordinates, widths and heights). The root node of the resulting tree corresponds to the whole rendered page area and each parent box encloses all its descendant boxes.

The visual area model is built from the box model using page segmentation. Its structure is similar to the box model but unlike the box model that describes the content and its visual properties as a direct result of the document rendering, the visual area model represents the division of the page to abstract regions created by visual means – *visual areas*. Similarly to a box, each visual area is defined as a tuple:

$$a = (x, y, width, height, bgcolor) \tag{2.4}$$

where $x$, $y$, *width*, *height* and *bgcolor* have the same meaning as for boxes (2.2). The whole visual area model is hierarchically organized, and it is also defined as a tree that represents the visual nesting of the areas:

$$T_a = (A, E_a) \tag{2.5}$$

where $A = \{a_1, a_2, a_3, \ldots a_m\}$ is the set of all visual areas detected in the page, and $E_a$ is the set of edges; $\forall a_1, a_2 \in A : (a_1, a_2) \in E_a$ iff $a_2$ is visually enclosed in $a_1$.

The proposed models provide a format-independent description of the source document and are used for representing the document content during different phases of page segmentation (as described in chapter 3) and later for further classification of the individual page objects (see chapter 4).

More details on the visual document model and its applications can be found in Appendix A.1 and Appendix B.2.

## 2.3 Ontological Model

The visual document model presented in the previous section is based on an abstract definition of the boxes, areas and their hierarchy that has been later implemented[3] for the applications mentioned above. Such model implementations (in our case the corresponding data structures in Java) are used internally by the respective software tools and since they have no explicit representation, they cannot be easily shared among software components. This makes it impossible to easily reuse the document analysis and preprocessing methods for other applications.

The ontological model that we proposed in [93, 95] provides an explicit representation of the visual document model that is based the RDF framework [132] and a set of ontologies specified in the OWL language [99] that define a shared vocabulary for the RDF data.

[3] As a part of our FitLayout framework described in Appendix D.3.

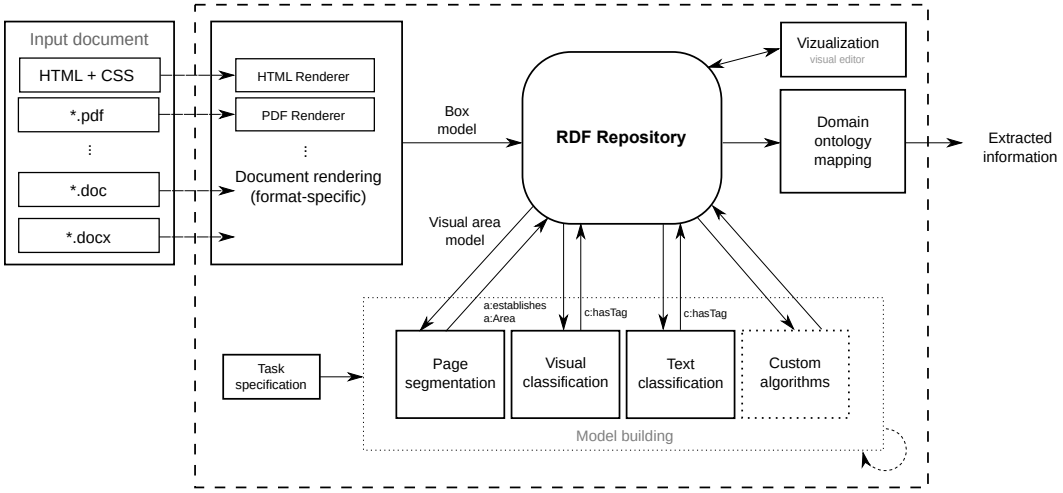



Figure 2.1: General architecture of a document processing system based on the ontological model with a central RDF repository [95].

The RDF formalism allows representation of different relationships among the individual document parts and their features using a generic graph structure. The goals of the model design are the following:

- To allow sharing the obtained document models between the document preprocessing algorithms on the one hand (such as the page segmentation and layout analysis algorithms) and specific applications such as document indexing or classification on the other hand in a standard way.
- To allow semantic reasoning and querying the document model using the existing software tools and languages such as SPARQL [108] and SWRL [64] in order to infer new knowledge about the document content.
- To simplify the integration of the document content with existing knowledge bases such as DBPedia [29] or creating new datasets with a given target RDF schema [94].

The chosen RDF technology makes the model very flexible and extensible and moreover, it also allows to use existing software tools for querying the modeled data and reasoning. The assumed architecture of a document processing system based on the proposed ontological model is shown in figure 2.1. We assume a central RDF repository that stores the RDF graph representing the document model. The document may be processed in a variable number of steps. The first step typically involves document rendering; the obtained box model is represented as an RDF graph and stored in the repository. Subsequently, page segmentation step may follow that takes the box model as its input and produces a visual area model, that is represented as an RDF graph again and it is added to the central repository alongside the box model. The remaining steps (e.g. the content classification steps) are application-specific and they may add any additional RDF triplets providing additional information about the document content elements obtained in the respective steps (see section 5.2 for an example of its application for a structured record extraction task).

The generic document model discussed in this section covers the page rendering and page segmentation steps that are commonly used in all the document processing tasks

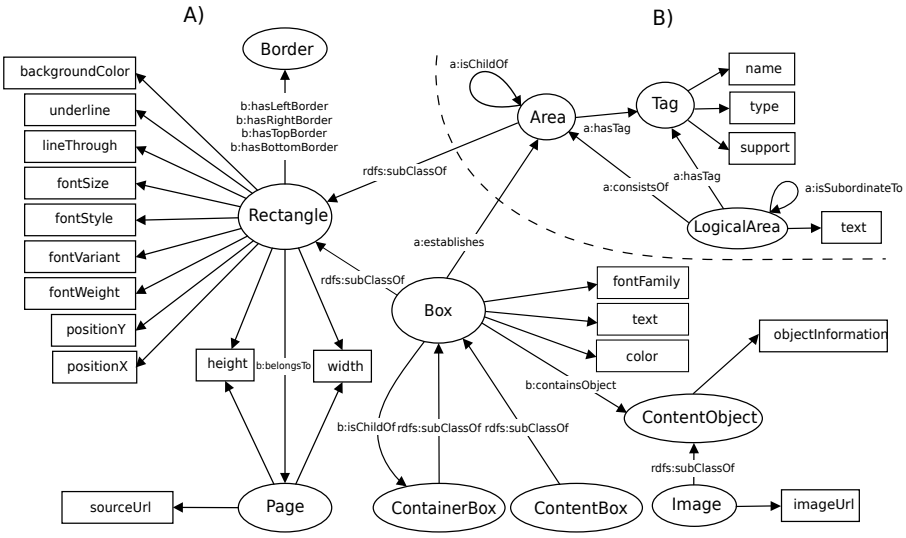



Figure 2.2: A) Box tree ontology B) Visual area ontology

discussed in further chapters. We define two ontologies that correspond to the two levels of document description introduced in Section 2.2:

1. *Box tree ontology*[4] – describes the box model of the given document that represents the result of the page rendering.

2. *Visual area ontology*[5] – describes the visual area model (page layout) as a product of page segmentation.

Both ontologies define the basic concepts and properties that are used in the document description as shown in Figure 2.2. The most important concepts are *Box* that corresponds to the box definition (2.2) and *Area* corresponding to the visual area definition (2.4). Moreover, the visual areas may be assigned different tags that may be used for the classification of the individual parts of the document content and its integration to existing datasets as discussed in section 5.2.

More details about the ontological model can be found in Appendix C.1. Formal definitions of the box tree ontology and the visual area ontology are available[6] in our FitLayout framework described in Appendix D.3.

## 2.4 Graph-Based Model

Although the visual and ontological models presented in the previous sections are independent on document format, they still rely on the content structure information provided in the source documents. For HTML documents, this includes mainly the division of the content to individual HTML elements that are treated as atomic content units. For some applications, such granularity may not be sufficient because the extracted information may only form a part of the content element [30]. For PDF documents, this division may not be

[4]URL prefix: `http://fitlayout.github.io/ontology/render.owl#`
[5]URL prefix: `http://fitlayout.github.io/ontology/segmentation.owl#`
[6]https://github.com/FitLayout/FitLayout.github.io

even available at all; their content presentation is often described by individual words or even characters. Therefore, for the purpose of structured records extraction, we have proposed a graph-based model of the content. Its goal is to capture the possibly relevant parts of the document contents and their mutual relationships based on their visual presentation independently on the content division provided in the document code.

The document content model is defined as a graph

$$D_c = (C, E) \tag{2.6}$$

where $C$ is a set of *text chunks* that represent the relevant content parts together with their visual formatting and form the vertices of the graph; $E \subseteq C \times C$ is a set of graph edges, that represent the *relationships* among the chunks expressed by the document layout.

With a *text chunk*, we understand any substring of the document text, that possibly represents a value to be extracted. The discovery of the chunks depends on the target application domain; they may be identified for example by regular expressions, NER classifiers or other means as discussed in section 5.4. A chunk is defined as a tuple

$$c = (t_c, s_c, p_c) \tag{2.7}$$

where $t_c$ is the text of the chunk (the actual substring of the document text), $s_c = (fs, w, st, c, bc)$ represents the visual style of the text and $p_c$ where $fs$ is the average font size, $w \in [0, 1]$ is the average font weight from 0 (normal font) to 1 (bold font), $st \in [0, 1]$ is the average font style (1 for italic font, 0 for regular font) and $c$ and $bc$ are the computed foreground and background colors of the displayed chunk. $p_c = (x, y, width, height)$ represents the $x$ and $y$ coordinates of the chunk in the page and its width $w$ and height $h$ in the rendered page.

The set of edges $E$ of the graph represents *relationships* between pairs of chunks. Given the set $C$ of discovered chunks, we analyze all the chunk pairs $(c_1, c_2) \in C \times C$ and we investigate whether there is a relationship between $c_1$ and $c_2$ given by their mutual positions $(x_1, y_1)$ and $(x_2, y_2)$. We have identified several spatial relationships that are interesting for further analysis. Each relationship is represented by a relation $E_x \subseteq C \times C$ and we say that there is a spatial relationship $x$ between $c_1$ and $c_2$ iff $(c_1, c_2) \in E_x$. For the purpose of structured record extraction, we consider the relationships such as *onRight*, *below*, *sameLine* and several more as described in section 5.3.

The resulting graph model represents the page content and its layout in a way that is completely independent on the document format and the particular implementation including the division of the content to elements. In the same time, the graph representation is suitable for further analysis for example by graph matching algorithms as discussed in section 5.4.

More details about the document graph model and its usage for structured record extraction can be found in Appendix C.3.

# Chapter 3

# Web Page Segmentation

An elaborate visual organization of the content is a typical feature of web pages. In addition to the main content that corresponds to the primary purpose of the page, the web documents typically contain additional information of different kinds such as navigation (menus), links to related sources, headers and footers or advertisement [10, 27, 40, 46, 139]. Even the main content of the page may be divided into multiple thematically different units, such as different articles published on one page of a news portal. This inhomogeneity significantly complicates accessing the page content by non-visual means [40] or mobile devices [38], document indexing [115, 139], classification [70], duplicate content detection [35], content integration [25, 106, 120, 131] and other tasks of document content processing.

The goal of web designers is to provide enough visual hints about the actual content organization, which makes it trivial for a human reader to recognize the different content units and their purpose. However, the content division to semantic blocks is almost never explicitly annotated in the document code [5, 20]. The task of web page segmentation is therefore to analyze the input document and the contained hints intended for the readers and to provide an explicit decomposition of the page to separate content blocks. This is regarded an essential task in web information mining [20]. In our work, we use page segmentation mainly as a document pre-processing step for the classification and extraction of semantic page objects (as discussed in Chapter 4) and structured records extraction (Chapter 5). Due to specific requirements of the individual applications, we have developed two new methods that meet these requirements and can be easily integrated with the subsequent steps. These methods are introduced in sections 3.2 and 3.3. In section 3.4, we discuss the possibility of efficient segmentation of large sets of documents based on the detection of shared templates.

## 3.1 State of the Art

Since page segmentation represents usually the first step in document processing, it greatly depends on the used document model as discussed in Chapter 2. From this point of view, existing segmentation methods may be divided to several categories [47]:

- *DOM-based approaches* that directly process HTML documents represented as a DOM and attempt to discover a mapping between some HTML elements or their sequences and visually separated blocks that appear in the resulting page. Since DOM actually models the code structure rather than the displayed page (as discussed in Section 2.1), DOM-based approaches typically rely on different heuristics regarding the common

usage of individual HTML elements and repeating code patterns in the documents in order to estimate the role of the element in the resulting visual presentation [62, 35, 110, 66, 123, 124, 127, 134].

- *Text-based approaches* that focus on the properties of the text contained in different HTML elements while analyzing the DOM. Due to the hierarchical structure of the DOM, it is possible to find subtrees containing text with consistent properties such as text density or frequency of specific characters (for example punctuation characters). Such DOM subtrees then represent the discovered page segments [24, 46, 70, 69, 120].

- *Vision-based approaches* that employ page rendering in order to obtain additional information about the visual presentation of the individual content elements that is not directly available in DOM. The usage of this visual information spans from a simple extension of the DOM-based heuristics by considering the visual properties of the nodes [34] up to purely graphical representation of the rendered page [40] as we discuss below in more detail.

- *Hybrid approaches* that combine the DOM-based and vision-based ones [53, 62, 104, 105] or even include text-based approaches [114] in order to obtain higher segmentation accuracy or for specific applications.

Additionally, page segmentation methods may be divided into two groups based on the segment analysis direction: In *top-down approaches* [3, 34, 127, 150], the algorithm starts with a single segment representing the whole page and tries to divide the segments recursively until a predefined segmentation granularity is reached. In contrary, the *bottom-up approaches* [10, 35, 70] start with the smallest atomic units detected in the page and build the hierarchy of segments by grouping them into larger units while considering their visual, textual or DOM properties.

DOM-based and text-based approaches are typically very fast because no complex document preprocessing (such as style analysis or rendering) is required. However, their accuracy greatly depends on the code properties and the used heuristics [144]. For both the web page designers and readers, the visual information plays an important role in expressing the actual structure of the document. Therefore, the effort of many authors focuses on exploiting the visual information in page segmentation methods.

One of the first and definitely the most popular vision-based algorithm is VIPS [34]. It segments the page based on the detection of visual separators represented by the content elements placed in the rendered page. Although the visual features of the individual elements (such as font sizes and colors) and their positions in the rendered page are considered, the segmentation itself depends on a number of heuristic rules that are based on the underlying DOM and expected usage of particular HTML elements. Therefore, VIPS is sometimes considered a DOM-based method with visual cues [145]. Newer methods extend VIPS by adding new heuristics that reflect the evolution of HTML [5, 3] (new and deprecated tags) or text-related heuristics [80, 150]. Modern vision-based segmentation approaches avoid the heuristic rules completely and replace them by general graph metrics that include the visual properties of the DOM elements [10], analysis of the spatial relationships between DOM elements [85] or the visual similarity and distance of the rendered elements [135, 145].

While the above-mentioned approaches still rely on the DOM representation of the source document, a few vision-based approaches use entirely graphical representation of the input document that allows to abstract from the HTML-related implementation details.

The segmentation then consists of the discovery of graphical separators in the page image. This may be achieved by employing edge detection algorithms [40], Hough transform [129] or a recognition of atomic graphical objects and their grouping [72].

In the following sections, we briefly describe our contribution to the range of page segmentation methods. Main motivation of our research presented in the sections below is to achieve the independence on the document format and implementation, which includes avoiding the usage of DOM for the input document representation and in the same time, to avoid the complexity of the image processing methods. In Section 3.2, we propose a method based on hierarchical grouping of abstract content boxes and in Section 3.3, a non-hierarchical flat model is used together with box clustering methods.

## 3.2 Hierarchical Page Segmentation

Our page segmentation algorithm used in [25, 26, 31] is built upon the visual page model introduced in section 2.2. It takes the *box tree* $T_b = (B, E_b)$ (2.3) as its input and builds an *area tree* $T_a = (A, E_a)$ (2.5) in a bottom-up manner in the following steps [26]:

1. Extraction of the smallest visual areas from the box tree. For each box $b \in B$ that is *visible* in the rendered page, we create a *basic visual area* $a$ with the same position and size as the respective box in the source page. The box is considered visible if it contains a non-empty content (text or image) or it has a background color different from its parent box or a visible border. By applying this process recursively on the input box tree from the root node to its leaf nodes, we obtain a tree of basic areas $T_a$.

2. First clustering phase. We detect groups of areas in $T_a$ that share the same parent area and they are not visually separated by different background color or a visible border. Such areas are joined into a single one. Optionally (depending on the target application), it may be required that the joined areas share the same font size, weight and style. This step represents the detection of atomic content blocks (for example text paragraphs) that consist of several boxes in $T_b$.

3. Second clustering phase. We look for groups of visual areas that are not separated by visual separators, but they are delimited with other visually separated areas around. For every non-leaf area $a$ in $T_a$, we first detect the visual separators between its child areas that may be created by visible box borders or rivers of white space as defined in [34]. Then, we try to find the largest new rectangular *covering areas* that cover multiple child areas of $T_a$ and in the same time, they don't cover any visual separators. When found, a new covering area $a_c$ is inserted into $T_a$ as a child node of $a$ and a parent area of all the covered areas. Then, the whole process is applied recursively.

In order to detect the visual separators and the covering areas, we use an auxiliary structure of topographical grid $g$ which is built for every non-leaf area $a$ in $T_a$. It represents mutual positions of the child areas of $a$ as illustrated in figure 3.1.

The details of our hierarchical bottom-up segmentation method can be found in Appendix A.1 and Appendix B.2.

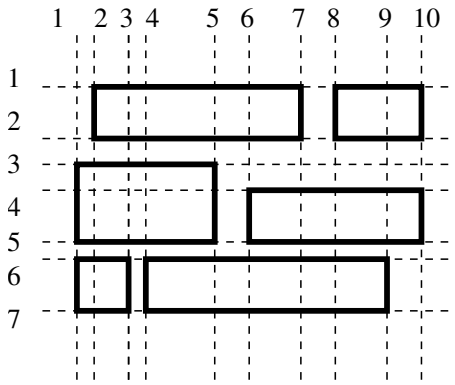



Figure 3.1: Mutual positions of the areas in a grid [31].

## 3.3 Flat Segmentation Model

The segmentation method described in the previous section (as well as most of the above-mentioned vision-based methods) produces a hierarchy of areas represented by a tree that models the nesting of the detected visual segments. This corresponds to the usual organization of the page content as it is expected to be perceived by human readers. However, for many applications, the complete hierarchy of the detected segments is not useful; only the leaf areas of the area tree are used because they correspond to the segments of the expected granularity [28, 35, 94, 106]. Based on this observation, we have developed a new *Box Clustering Segmentation* method (BCS) [144], that produces a flat model – a list of detected visual segments of desired granularity instead of a hierarchical model. The flat segmentation model allows to avoid repeated iterative merging of the visual areas and re-calculation of visual separators, which significantly reduces the time complexity of the whole segmentation process.

Simultaneously, we strictly avoid using DOM and the HTML-based heuristics, in the BCS method. We use a purely visual representation of the documents using the visual box model (similarly to [40, 129]) and our hierarchical segmentation method mentioned in section 3.2. On the other hand, BCS does not include an explicit detection of visual separators which makes the approach closer to the Web Content Clustering [10].

The whole segmentation process consists of the following steps:

1. *Box extraction* – we render the page and create the Box model as defined in section 2.2. Then, we choose the leaf boxes that represent a visible text or image in the rendered page. We obtain a set $B$ of content boxes in the page. For each box $b \in B$, we compute its color, position and size.

2. *Area graph creation* – we consider all pairs $(b_1, b_2) \in B \times B$ and based on their mutual positions, we check whether $b_1$ and $b_2$ are horizontally or vertically aligned so that we can say that $b_1$ is above ($a$), below ($b$), to the right ($r$) or to the left ($l$) of $b_2$. For the box pairs that are aligned, we compute their absolute pixel distance $abs(b_1, b_2)$ in the corresponding direction. Then, we may define a *direct neighborhood* $N_b$ of a box $b$ as a set of boxes $b_x \in B$ for which $abs(b_1, b_x)$ is minimal for some of the four directions ($a$, $b$, $r$ or $l$). As the result, we obtain a graph of boxes and their direct neighborhoods.

3. *Box clustering* – based on several box similarity metrics described below, we try to find a set $C$ of clusters that contain the most similar boxes. The process starts with

a set of initial clusters formed by pairs of the most similar neighboring boxes and subsequently, new boxes are incrementally added to their adjacent clusters in the order of their similarity with the remaining boxes in the cluster. The cluster creation is finished when the similarity of all the remaining boxes in the direct neighborhood of the cluster is lower than a predefined threshold $CT$ (clustering threshold).

For the clustering step, we define an overall similarity of two boxes $bsim(b_1, b_2)$ where $b_1, b_2 \in B$ that is defined as an average value of the following three similarity metrics:

- $distance(b_1, b_2)$ – a relative distance between the boxes within their direct neighborhoods.
- $sim\_shape(b_1, b_2)$ – the similarity of the box shapes that includes the size similarity $size(b_1, b_2)$ and aspect ratio similarity $ratio(b_1, b_2)$.
- $sim\_color(b_1, b_2)$ – color similarity of the two boxes computed from the values of the color channels in the RGB model.

The similarity $csim(c, b)$ of a cluster $c \in C$ and a box $b \in B$ is later computed based on the similarity of the boxes that are already contained in the cluster $c$ with the box $b$. The values of the $bsim$ and $csim$ similarity functions range from 0 to 1 and they are used for creating the initial clusters and selecting new boxes for the addition to the clusters respectively. Simultaneously, the possible overlaps among the clusters and/or unclustered boxes must be considered as overlaps among the detected clusters are not permitted.

We evaluated the proposed segmentation algorithm on a set of 2400 web pages of different types (e.g. complex index pages, article pages, etc.) and we computed the segmentation accuracy by comparing the results with reference results obtained by manual annotation of the pages by three independent volunteers. Additionally, we have compared the segmentation accuracy and the performance of the BCS algorithm with the reference VIPS algorithm. The results show that our algorithm is almost 90 % faster than the reference algorithm and the segmentation accuracy is between 47 % and 133 % of the reference algorithm accuracy depending on the page type.

The detailed description of the BCS method and its experimental evaluation can be found in Appendix A.2.

## 3.4 Template-Based Segmentation of Large Sets of Pages

The vision-based methods allow to achieve greater accuracy of page segmentation at the cost of significantly more complex page processing that includes the acquisition of additional data files (mainly referenced style sheets) and complete page rendering. Required time for segmentation may span from 15 ms to 700 ms for BCS and even more for VIPS and other methods [144], which may be acceptable for processing single pages but it makes the methods hardly applicable to large web sites containing hundreds or thousands of documents. Therefore, in [141, 143], we have proposed an optimization approach for page segmentation based on re-using the segmentation results for complete sets of documents coming from the same source instead of performing the segmentation individually on each of them.

Our *Cluster-based page segmentation* (CBS) approach is based on the observation that in modern web design, the documents contained in large web sites are almost never created independently on each other. Most often, they are generated using pre-defined templates

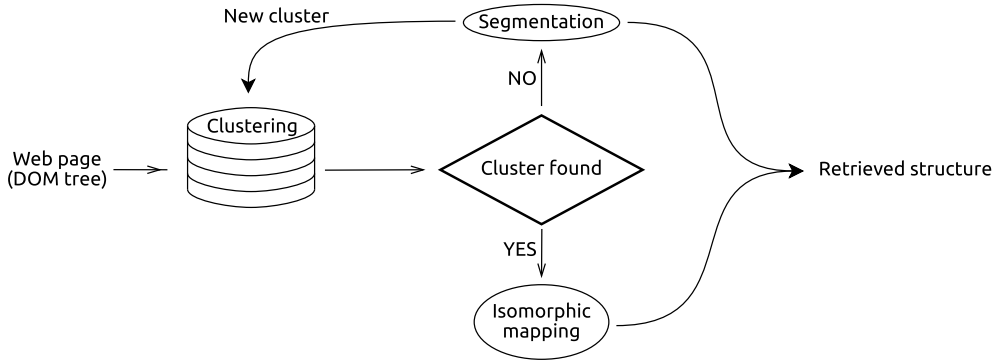



Figure 3.2: The cluster-based page segmentation approach [143].

that describe the basic organization of the page which is later filled with the content specific for the individual documents [18]. The problem of template detection itself (i.e. to distinguish the part of the page generated from the template from the main content) is a separate research area which is closely related to page segmentation [57, 126, 125]. However, in the CBS approach, our goal is different: We only need to recognize pages generated from the same template in order to reuse the segmentation result for the whole set of pages.

The basic idea behind our approach is shown in figure 3.2. We assume that the pages coming from the same source (e.g. a website) may be grouped to a few clusters that correspond to different templates used within the source web site. For each processed page, we first start with a *clustering step*, which results in the decision whether the page belongs to an existing page cluster, i.e. it corresponds to an already known page template. If no such cluster is found, the page is segmented by a chosen segmentation method and the segmentation result is stored together with the new page cluster created. On the contrary, when the page belongs to an existing page cluster, the stored segmentation result is just applied to the new pages by finding a mapping between the DOM nodes of the already segmented page and the newly processed one. This makes the template-based segmentation independent on the actual segmentation method used; it just provides an additional optimization layer on top of the segmentation itself.

The discovery and matching of the page clusters is based on a DOM-to-DOM comparison using a modified Common Paths Distance algorithm [57]. Each cluster is then represented by a *cluster representative*, which is a structure that contains a simplified DOM of a representative page of the cluster. It contains only the element nodes that are the most important for further comparison with newly processed pages. The expected number of page clusters (individual templates) that may be discovered within a single web site is quite low; according to our experiments [143], it does not exceed 50 clusters even for large web sites. Figure 3.3 shows the dependency of the number of discovered clusters on the number of processed pages for several large web sites. Since the number of clusters is low and the comparison method is fast enough, any newly processed page may be compared with all the already existing cluster representatives sequentially in order to decide whether it belongs to an existing cluster or a new cluster should be created.

The mapping of the already segmented pages to the newly processed ones is also performed on the DOM level based on our previous work published in [142]. It is based on finding common subtrees in both DOM trees with an assumption that the order of child nodes of any parent node in the DOM tree must be preserved if the page layouts are to

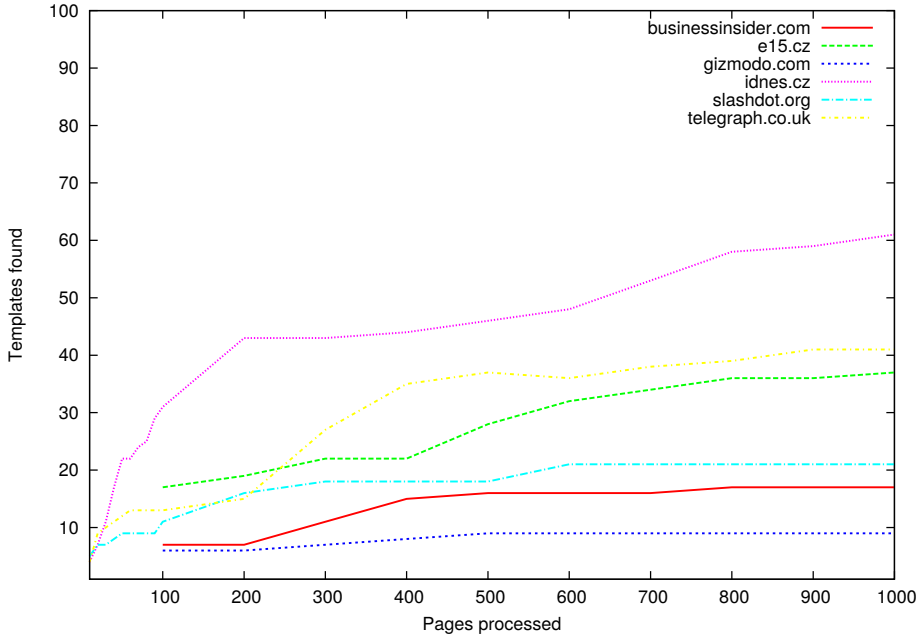



Figure 3.3: Dependency of cluster count on the number of processed pages [143].

be equal. Therefore, we search for common subtree root nodes by matching their indices within their ancestor nodes.

More details about the cluster-based segmentation method and its experimental evaluation can be found in Appendix A.3.

# Chapter 4

# Classification and Extraction of Semantic Objects

Page segmentation methods introduced in the previous chapter represent typically the first step of web page processing. They provide a coarse information about basic division of the page to separate content blocks or even finer content elements depending on the chosen segmentation granularity. This information allows to perform subsequent analytical steps aimed at obtaining more detailed information about the possible roles of individual page elements and their mutual relations. Typical tasks in this area include the following:

- *Main content extraction* (or *web page cleaning*) – extraction of the main content (which is typically a published article, product information, etc.) while discarding additional content that does not correspond to the main purpose and topic of the whole page. This is a typical pre-processing step for web content mining and information retrieval methods where the additional information acts as noise that decreases the precision of the methods [9, 46, 114, 137, 139].

- *Recognition of important content elements* – the recognition of the parts of the contents that have a specific role in the document. This may include the discovery of important parts of the documents such as headers and footers and menus [59, 87, 75, 76], or at finer level annotating the generic elements such as headings, lists or structure of presented tables [4, 111, 131] or even the elements containing a domain-specific information [31, 98, 102].

- *Logical structure discovery* – estimation of mutual relationships among the individual content elements, mainly the relationships between the headings or labels and the corresponding contents [45, 90, 121].

As we may notice, these tasks often overlap to some extent and they share similar methods and approaches. In this chapter, we present our contribution in all the mentioned areas. In section 4.1, we deal with a general approach to content element classification based on visual features of the individual elements. In section 4.2 we focus on a specific application of main content extraction from web pages. Finally, in section 4.3, we discuss the possibility of an explicit representation of logical relationships among the content elements using a tree model.

| | |
|---|---|
| `h1` | main article heading |
| `h2` | second-level heading in the article |
| `subtitle` | the article subtitle |
| `perex` | the leading paragraph of the article |
| `paragraph` | an ordinary paragraph |
| `date` | publication date |
| `author` | author name |
| `authordate` | author and the date in a single area |
| `none` | remaining areas that do not belong to the article |

Table 4.1: The classes assigned to the individual visual areas

## 4.1 Web Page Element Classification

The classification of web page elements represents a possible next step in web information extraction that follows the page segmentation. Its task is to provide an initial estimation of the semantic roles of individual parts of the document content, which may be directly used for the extraction of specific information as we propose in [31] or followed by additional refinement steps aimed at identifying larger units [27] or structured data records [29] in the page.

Content classification may be performed based on different content properties that include the characteristics of the text itself such as the frequencies of the specific character groups [15], DOM properties [6, 122, 124, 134] or visual features [115]. In our work, we focused on a pure visual approach that uses the visual presentation features of the contents such as font properties and positions in the page for recognizing specific page elements using a classifier that has been previously trained on a manually annotated training set of web pages.

In [32, 31], we propose a classification approach for extracting news articles published on the web and their general metadata that include the title of the article, authors and date of publication (see the complete list in table 4.1). The entire approach is based on an assumption that the way of visual presentation of the mentioned data follows some commonly accepted rules that include mainly the relative font sizes and mutual positions of the individual content elements (for example, headings use larger font than the rest of the text) and that allow the reader to easily recognize the given information without actually studying the text content. The whole approach consists of a training and classification phase as shown in figure 4.1.

In both phases, the first step involves page segmentation using the hierarchical algorithm described in section 3.2. As the result of page segmentation, we obtain a tree of visual areas detected in the page (as defined in (2.5) on page 12) where the root area represents the whole rendered page and leaf areas correspond to the smallest visually consistent elements of the content. For each visual area, we compute its visual properties that include font properties (relative size, weight and style), spatial features (its relative position within the whole page and the numbers of areas above, below, left and right of the given area) and text features (numbers of lines and different groups of characters such as digits, letters or punctuation).

For the training phase, we created training examples by manually annotating the visual areas that contain the particular information in the rendered page. For this purpose, we

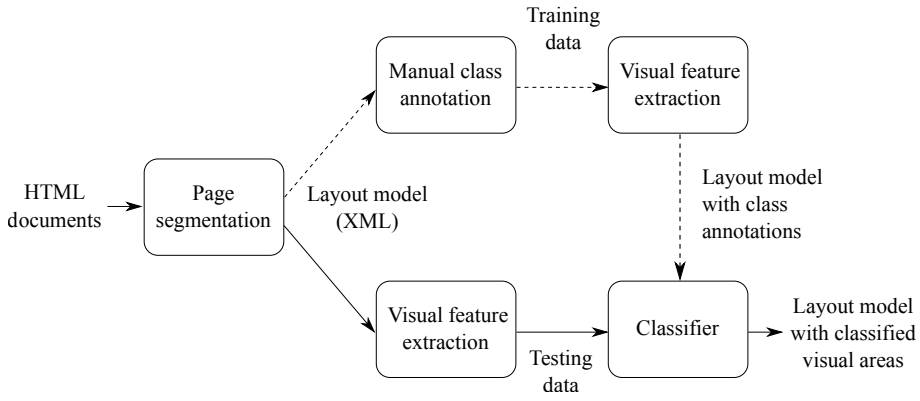



Figure 4.1: Visual area classification workflow [31]. Dashed lines denote the training phase, solid lines correspond to the classification phase.

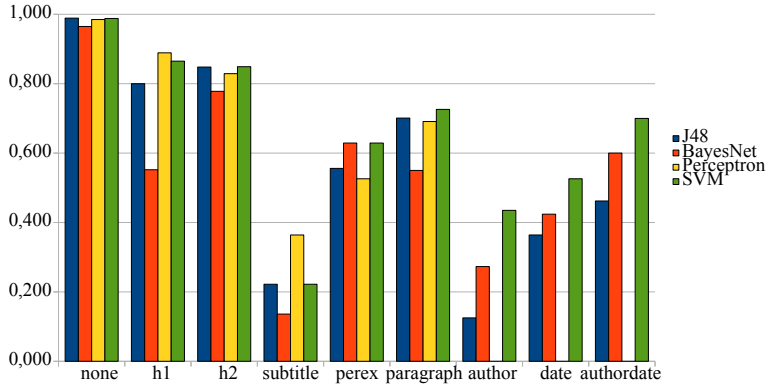



Figure 4.2: Comparison of the F-scores of the algorithms for individual classes – training and testing data mixed from all web sites [31].

implemented a graphical tool that displays the segmented page and allows to assign classes to the individual areas. The remaining areas which have no specific meaning are marked with a special class *none*. The assigned classes together with the computed values of the visual features for the respective areas are used as the training data for the classifier. Later, in the classification phase, the trained classifier is used for assigning classes to all visual areas detected in new documents that have not been previously annotated.

We tested our approach with four classification algorithms available in the WEKA[1] package: The J48 tree classifier which is an implementation of the Quinlan's C4.5 decision tree algorithm ([101]), Bayesian network, Multilayer perceptron and Support vector machines. For evaluating the applicability of these algorithms, we collected 559 documents from 20 different news web sites worldwide. Using our annotation tool, we have manually annotated four documents from each website and used them as the training data set; the remaining unannotated documents were used as the testing data set. Figure 4.2 shows the resulting F-scores obtained for the individual classes and algorithms.

As we may notice, although the documents come from different web sites with different visual presentation styles, some parts of the articles (such as headings) may be recognized quite reliably (the F-score obtained for the *h1* class using the SVM algorithm is 0.865). On

[1] https://www.cs.waikato.ac.nz/ml/weka/

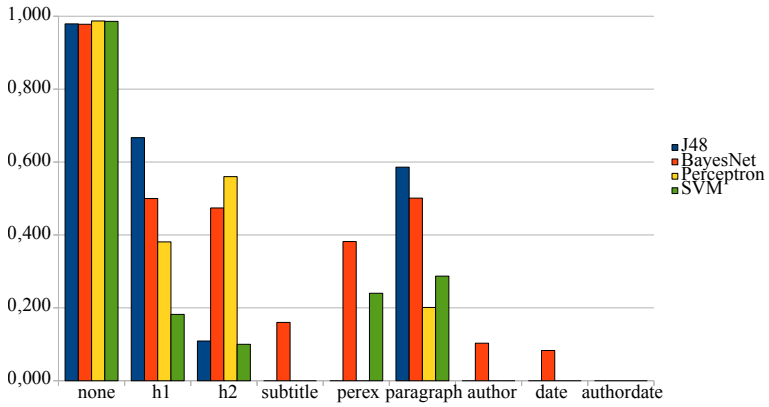



Figure 4.3: Comparison of the F-scores of the algorithms for individual classes – training data from different web sites than the testing data [31].

the other hand, the *subtitle* and *author* fields are presented in different ways or may not be even used in the documents so that the obtained F-scores are lower.

As the second alternative, we split the training and testing datasets so that the training data come from different web sites that the testing data. This represents a very "pessimistic" scenario where no training data is available for the web site being processed and the classifier must be trained on the page examples from other web sites. As we may notice in figure 4.3, the resulting F-scores are significantly lower.

As the most common usage scenario, we considered the situation where the training and testing document sets come from the same web site; i.e. the classifier is trained on a few manually annotated example pages and later, it is used for extracting the article contents from a large number of pages from the same web site. For testing this scenario, we have used the SVM and BayesNet classifiers and the obtained results are shown in table 4.2.

| | BayesNet | | | SVM | | |
|---|---|---|---|---|---|---|
| Class | P | R | F | P | R | F |
| none | 0.999 | 0.917 | 0.957 | 1.000 | 0.999 | 0.999 |
| h1 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| h2 | 0.727 | 1.000 | 0.842 | 1.000 | 1.000 | 1.000 |
| subtitle | 0.500 | 1.000 | 0.667 | 1.000 | 1.000 | 1.000 |
| perex | 0.286 | 1.000 | 0.444 | 0.800 | 1.000 | 0.889 |
| paragraph | 0.381 | 0.971 | 0.547 | 0.986 | 0.986 | 0.986 |
| author | 0.273 | 1.000 | 0.429 | 0.750 | 1.000 | 0.857 |
| date | 0.200 | 0.750 | 0.316 | 1.000 | 0.500 | 0.667 |
| authordate | 0.571 | 1.000 | 0.727 | 1.000 | 1.000 | 1.000 |

Table 4.2: Achieved results (precision, recall and F-score) for the training documents coming from the same source as the later annotated documents [31].

In this scenario, the precision of the classification is sufficient for extracting most information about the articles, their basic structure and metadata from the web pages. The results practically demonstrate the possibility of using the visual features for the recognition of specific parts of the text content, which has multiple applications. In the following sec-

tion 4.2, we describe possible extensions and applications of this approach for main content identification in web pages.

More information about the proposed visual element classification method and its experimental evaluation is available in Appendix B.2.

## 4.2 Main Content Extraction from Web Pages

Main content extraction is one of the most common tasks of web page processing. Most often, it is related to processing of news (or other) articles that are published on the web with text mining algorithms [103, 134, 137, 139] but many other applications exist as for example accessing the article content on small screens [2]. As we have mentioned above, the web pages typically contain different kinds of additional information which is not directly related to the main content of the published article and from the text mining view, it may be viewed as noise that should be removed before the documents are further processed [9, 137]. Therefore, the same task may be referred to as *main content* or *article extraction* [8, 148, 128] or *document cleaning* [88, 116]. Often, the additional information in the web page is actually generated by pre-defined templates which are dynamically filled with the content to be published. Then, the document cleaning consists of *boilerplate detection and removal* [7, 69].

The methods used for the main content extraction are based on similar principles as the page element classification methods mentioned in the previous section that are more general. Again, the methods may be roughly divided into text-based (statistical) [57, 69, 134], DOM-based [7, 8, 125, 137] and vision-based [148, 128]. However, these approaches typically overlap combining some kind of page segmentation (DOM or vision-based) as the first step and a subsequent statistical evaluation of the detected blocks as the second step [114].

In [27], we presented an approach to web content restructuring with the goal of improving the accessibility of the main content of the page. In contrast to the main content extraction methods, our approach preserves the complete content of the source web page; it just detects the consistent content blocks and reorders them so that the main content (e.g. a news article) becomes the first one in the web page. For the detection of consistent content blocks, we use our page segmentation algorithm described in section 3.2. To prevent the segmentation method from dividing the main content into multiple sections that could be incorrectly reordered later, we added a new *logical block detection* step. It consists of estimating the possible headings in the content using simple font size-based heuristics followed by content layout analysis, which detects content columns that belong to each heading. Similar approach based on heading detection was later proposed by other authors in [90]. Finally, the detected logical blocks are ordered based on the average font size of the detected headings (assuming that the importance of the headings is visually expressed by the used font size).

In [28] we have combined the above logical block detection method with our page element classification methods described in section 4.1 to create a new algorithm for article extraction from web pages. Our algorithm is based purely on the analysis of the visual aspects of the rendered page, which is processed in the following steps:

1. Page rendering and segmentation – creation of a visual model of the page.

2. Visual area classification – estimation of the page elements (visual areas) that may represent a part of the main article for some content.

| | (A) | (B) |
|---|---|---|
| Left boundary | 0 % | 6 % |
| Right boundary | 4 % | 17 % |
| Top boundary | 0 % | 1 % |
| Bottom boundary | 19 % | 27 % |
| (incomplete article) | 2 % | 13 % |
| (additional content) | 17 % | 19 % |
| Overall failure rate | 21 % | 29 % |
| without add. content cases | 6 % | 19 % |

Table 4.3: The percentages of erroneous identification of the individual article boundaries for the articles from a single source (A) and from mixed sources (B) [28].

3. Article identification – determining the exact bounds of an area on the page that most probably contains the published article.

The first two steps are almost identical with the page element classification process described in section 4.1; the classifier is used to assign the classes listed in table 4.1 to the individual areas. The most important step is the subsequent article identification. It is based on the following assumptions regarding the article presentation on the web:

- The article forms a rectangular area in the page.
- It usually starts with a heading.
- It consists mainly of a consistent flow with text paragraphs with occasional inserted boxes that may be further structured (for example images or information boxes).
- The article content is left-aligned (or right-aligned for languages written in the right-to-left direction) or block-aligned.

During the article identification, each visual area that has been assigned the *h1* class in the classification step is considered as a possible heading of the article and we try to identify the largest rectangular area that meets the above assumptions; mainly, that contains an aligned flow of paragraphs (visual areas with the *paragraph* class assigned) while allowing certain number of exceptions (areas not recognized as paragraphs) given by a configurable threshold.

We evaluated the proposed approach on the dataset of news portal web pages described in the previous section. Similarly to the classifier evaluation described above, we tested article extraction for a single web source (i.e. the pages used for training the classifier come from the same source as the testing pages) and for mixed sources (the training and testing sets contain mixed documents from all the sources). Table 4.3 shows the error rates in the identification of the four article boundaries. As we may notice, the bottom boundary detection is the least reliable, which corresponds to the fact that there is often an additional information provided below the article (such as links to related articles, discussion forum, information about the author, etc.) for which it is not always easy to decide (even for the human reader) whether it does form part of the article or it does not. Therefore, on the


Figure 4.4: A conference program [33].

last row of table 4.3, we include the error rates obtained when the additional content at the end of the article is not considered as an error.

In [29], we have proposed an alternative approach where the published articles are treated as structured data records that consist of the basic information to be extracted (title, author name, publication date and the content represented as a sequence of paragraphs). We assume that there are multiple articles from the same source available and they share a common presentation style. We use the page element classification method described in section 4.1 for an initial approximate recognition (*tagging*) of possible occurences of the individual article parts and we apply the structured record extraction method presented further in section 5.3 for extracting the complete articles. The extraction itself is based on the comparison of multiple articles and the discovery of possible visual presentation patterns that are later used for the disambiguation of the initial classification results. As we also show in [29], this additional disambiguation step significantly improves the precision of the article identification.

More details about both the web content restructuring and the article extraction methods and their experimental evaluation are available in Appendix B.1. The application of the structured record extraction method for article identification is described in detail in Appendix C.2.

## 4.3 Discovery of Logical Relationships among the Page Elements

In the previous sections, we addressed the discovery of individual units in the documents that included atomic page elements (such as specific parts of news articles in section 4.1) or larger content regions (such as the complete articles in section 4.2). Looking closer, we may notice that the discovered articles have some internal structure, which is apparent to the human reader: Each article has the main title and author and its content may be structured to sections and subsections. In other words, there exist some relationships among the page elements that form the article.

For another example, let's consider a conference program published on the web (see a sample program in figure 4.5). It consists of speech titles, times, places and author names with some relationships among them. However, these relationships are usually not explicitly annotated in the document itself. They are expressed by different, mostly visual means and the reader is expected to interpret the visually presented information appropriately in order to assign for example an appropriate author and time to a speech title.

The *logical document structure* describes the roles of the individual page elements and their mutual relationships as they are obtained by analysis and interpretation of the *layout structure* [68, 97, 102, 117, 136]. Depending on the target application, the roles of the page elements may refer to generic document parts such as headings of different levels, lists, footnotes, etc. [97, 117, 90] or domain-specific data fields [98, 102]. Therefore, we may say that the logical structure discovery methods are on the borderline between the content classification as already presented in section 4.1 and structured record extraction as discussed further in chapter 5. Logical document structure discovery is usually performed in two steps:

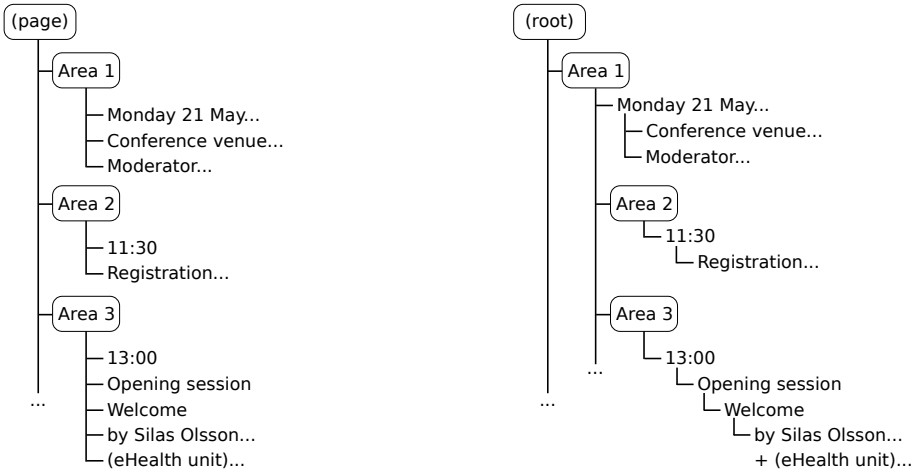
1. *Layout analysis* – creation of a document layout model by applying some kind of page segmentation on the source document.

2. *Logical structure analysis* – analysis of the layout model for discovering the roles and relationships among the content elements.

As for the logical structure analysis, the existing methods take into account different visual properties of the content that include positions within the page, spacing and indentation and font properties [117]. Their values are then evaluated using different heuristic rules [90, 97, 117] or machine learning methods such as Support Vector Machines [68] or Conditional Random Fields [87] that assign the roles from a pre-defined set to the individual content elements. In most mentioned approaches, the relationships among the elements are not represented explicitly; they are silently expected to arise from the assigned roles and the ordering of the content elements. As the result, the role assignment may be ambiguous mainly when multiple articles (or generally logical entities) are present in the document.

In [33], we proposed to formalize the mutual relationships between the individual content elements in the document, as they are presented by visual means, as an additional step that precedes the role estimation itself and that provides additional information, which is later usable for the completion of structured data records. Our proposed *Logical Relationships Model* (LRM) describes the relationships of logical subordination (see below) among the content elements as they are visually presented in the document. Unlike the complete logical structure discovery process, which depends on the target domain as noted above, LRM is domain-independent; it just represents how the visually presented relationships among elements can be interpreted without actually assigning any roles or semantics to the elements.

With *logical subordination*, we understand the logical relationship that occurs between a heading and the corresponding article contents, a term and its definition, etc. When we consider the tree $T_a = (A, E_a)$ of visual areas as defined in (2.5) on page 12 and two visual areas $a_1, a_2 \in A$, we say that $a_2$ is logically subordinate to $a_1$ if the content of $a_2$ elaborates or concertizes the content of $a_1$. In web documents, this relationship is visually presented by some commonly accepted visual patterns that are culturally determined (as noted in [111]). They include mutual positions (including indentation) and visual properties of the visual areas. In [33], we also proposed a simple way of determining the logical subordination




(a) Layout structure

(b) Logical relationships

Figure 4.5: A layout and logical structure of a conference program [33].

based on the order of elements in the document ($a_1$ precedes $a_2$ in the document contents) and the *visual weight* of the areas ($weight(a_1) > weight(a_2)$), where the visual weight is computed from their indentation and font properties (mainly the font size, boldness and color properties).

Figure 4.5 shows the layout structure and the corresponding logical structure of the conference program shown in figure 4.4. The layout structure corresponds to our visual document model $T_a$ introduced in section 2.2. The edges of the tree represent the nesting of visual areas and the tree itself is obtained by page segmentation. On the other hand, in the logical relationships model, the edges of the tree represent the logical subordination as indicated by the visual presentation. We may notice that the visual weight of the main title is higher than the visual weights of the remaining content (mainly because the largest font size and bold font used) and similarly, the visual weights of the speech titles are higher than the visual weights of the speaker names. This is reflected by the edges of the LRM in figure 4.4.

The LRM may be defined as another tree

$$T_l = (A, E_l) \tag{4.1}$$

where $(a_1, a_2) \in E_l$ if $a_2$ is logically subordinate to $a_1$ according to their visual presentation.

We evaluated the method of the Logical Relationship Model construction on a dataset of conference programs and during the experiments 86 % of expected logical subordination relationships were recognized in the documents. The information about the content structure represented by the proposed model has applications in logical structure discovery algorithms as well as in the information retrieval and extractions areas.

More details about the proposed Logical Relationships Model, its evaluation and applications are available in Appendix B.3.

# Chapter 5

# Structured Record Extraction

In addition to other types of content mentioned before, the World Wide Web is also a rich source of structured data from different domains including product information on e-shops, real estate information, stock quotes, sports results, timetables and many more. Such data is potentially interesting for further processing (indexing, comparison, integration with other data sources, etc.) but although it is often stored in a structured way on the provider's side, it is only accessible via a web interface designed for manual browsing. As we already mentioned in the introduction of this thesis, due to the variability and loose structure of web documents, current practice in web data extraction is still based on procedural wrappers, that are specific for each data source and their creation and maintenance is laborious [107]. Therefore, much effort has been dedicated to the research of methods that would automate the process of the discovery and extraction of structured data records from web documents.

Structured record extraction is one of the most challenging areas of web document processing. In addition to a single entity identification discussed in the previous chapter, the goal of record extraction is to identify *data records* that consist of multiple *data fields* composed in flat [112] or even hierarchical [119] structures. Thus, the task requires both the data field and complete data record identification and extraction. The actual structure of the records to be extracted may be specified in advance [48, 100, 118] or discovered dynamically based on the analysis of the source documents [62, 110].

In our research in this area, we focused mainly on the integration of domain-specific data presented on the web to structured databases and information systems. Therefore, we assume that the expected structure of the records to be extracted is available in the form of a conceptual model obtained from the target domain analysis and we focus on the discovery of the records defined this way in source documents. In section 5.2, we discuss a specific approach designed for the domain of scholarly publications that uses a set of heuristics that are based on the visual presentation of the content. Subsequently in 5.4, we generalize this approach by replacing the heuristic rules by a model of the target domain.

## 5.1 State of the Art

The extraction of structured data from web pages has been the subject of research in many contexts. Since in many cases, structured data records on the web are produced as a result of a user's query (usually by filling a form in the web browser), extracting data records from query result pages is one the most popular applications [11, 56, 58, 62, 118, 130, 131, 149]. Other applications include the processing of large data sets that are available as collections

of web documents such as conference proceedings [43, 71], extraction of product information from e-commerce web pages [12, 119, 146] or from online marketing flyers [14, 54], event data extraction [81] or even published obituaries [48].

Similarly to page segmentation and other previously discussed areas of web document processing, most approaches are based on processing DOM representation of the document code [48, 62, 71, 81, 92, 110, 119, 146, 149]. Many newer approaches extend the DOM model with some kind of visual features gathered from the documents [14, 50, 56, 83, 100, 112, 138], often by including the VIPS-based page segmentation in the preprocessing phase [11, 84, 131]. Only a few approaches use purely visual document models independent on the input document format [49, 133].

Regarding the structure of the extracted records, most approaches infer the schema from the source documents themselves, i.e. the structure of the information is not explicitly specified in advance [50, 110, 140]. An usual approach is to perform a top-down analysis of the document structure starting with an identification of the most probable regions that contain the data records (called a result section [118], data sections [131] or data region [50, 56]). Then, the individual data records are identified based on the detection of repeating structures in the model by frequency measures [62] or visual pattern detection [11, 118, 131]. The structure of the extracted information is inferred from the discovered records while using additional information such as explicit labels present in the page [11, 118, 131, 149] or even the query interface in case of the query result extraction [118, 130]. This presentation-driven information extraction is suitable for general data mining; however, since the structure of extracted data may be arbitrary, this approach is not suitable for integration with existing structured databases.

Significantly less attention has been paid to the research of the model-driven approaches where the expected structure of extracted data records is based on a previously known domain model. Embley et al. [48] use a conceptual domain model that is directly mapped to HTML code based on different heuristics. In [100] a flat list of extracted data fields is used and [86] integrates the extracted data with an existing knowledge base. In [109], a domain ontology is used that combines both the extracted data fields and the classes related to the data presentation (pages, columns, etc.)

In our work presented in the following sections, we focused on the development of structured record extraction methods from web documents while pursuing the following goals:

- The method should not rely on the document code directly in order to maintain its applicability to different documents in different formats and languages that are very variable may evolve over time.

- We assume integration of extracted data records into existing domain-oriented information systems or databases with a fixed structure. Therefore, in the considered scenario, the structure of the information is always known in advance and the developed methods should use it when performing the extraction.

As the result, the presented methods gradually aim to a general approach to visual identification of data records with fixed structure that are contained in a document or set of input documents that may be very variable in both their visual presentation and technical implementation.

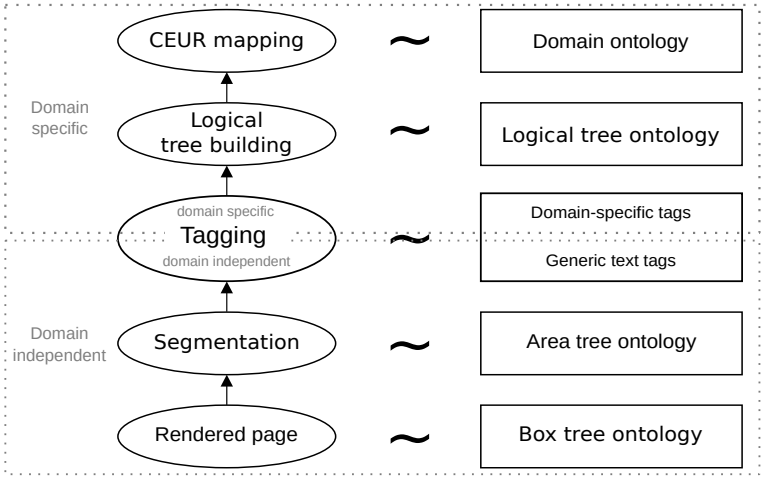



Figure 5.1: Document processing steps and the corresponding ontologies [94].

## 5.2 Heuristics-based Record Extraction

In [94], we published our solution of the Semantic Publishing Challenge at the ESWC 2015 conference. The task of the challenge consisted of extracting data about the workshops published at the CEUR Workshop Proceedings (CEUR-WS) website[1] and its transformation to a structured RDF representation that later allows to answer given questions about the workshops, their organization, publications and authors as summarized by the challenge organizers in [43]. Our approach was evaluated as both the best-performing and the most innovative one in the challenge.

The source CEUR-WS web site consists of HTML documents representing the individual workshops that have been added gradually for many years (since 1994) and they are very variable regarding both their code and visual style as web technology and the way of its use evolved during that time. Because of this variability, it is not possible to construct a single DOM-based wrapper for extracting the desired information directly from the HTML code. Successful solutions from the previous years [71] therefore used a set of templates that covered all different data presentation styles used in the source documents at HTML level, which requires a laborious maintenance of the template set as new documents are added. In our solution, we chose a different approach based on the visual presentation analysis and the ontological document model presented in section 2.3.

The architecture of our information extraction system is based on a central RDF repository as shown in figure 2.1 on page 13. The repository stores the results of individual document processing steps, from the visual model of the rendered document, through page segmentation and basic content classification, to the resulting domain-specific knowledge. The individual steps and the corresponding ontologies that are used for describing its results are shown in figure 5.1.

The initial steps of page rendering and segmentation have been described in detail in previous chapters and the produced ontology-based description of their output corresponds to the ontological document model presented in section 2.3. The subsequent *tagging* step represents the transition from the domain-independent document representation to the domain-dependent estimation of the purpose of the individual content elements which is

[1] http://ceur-ws.org/

further refined in subsequent steps. The *tags* represent named classes that are assigned to the individual visual areas to indicate its possible purpose; each visual area may have multiple tags assigned. Two levels of tagging are used:

- *Domain-independent tags* are assigned to the visual areas that contain specific kinds of data that may be easily recognized by text analysis. We recognize the probable occurrences of *dates*, *personal names*, *numbers* (that may later represent the page numbers) and *titles*.
- *Domain-specific tags* represent the data fields occur in the target domain: the *volume title*, *location*, *editor name*, *workshop date*, *paper title*, *paper author*, *page numbers* and some more.

The domain-independent tags are approximately assigned using a combination of the Stanford NER classifier [51] and simple regular expressions. Subsequently, they are mapped to domain-specific tags. As we may notice, the domain-independent tagging may be ambiguous from the point of view of the target domain; e.g. the recognized personal name may belong to a paper author as well as to a volume editor and similarly, a recognized title may belong to both a paper and the whole workshop. In addition, the accuracy of the entity recognition is limited. Therefore, we have defined a number of heuristics based the analysis of the source documents that are used for disambiguation and correct assignment of the domain-specific tags. They are based on the following aspects of the content and its presentation:

- *Common visual presentation rules* – application of some commonly used rules for visual formatting of the presented information in a document. E.g. a title or subtitle is written in larger font or at least bolder than a normal text.
- *Regularity in presentation style* – we assume that all the information of the same meaning (e.g. all paper titles) is presented with the same visual style (fonts, colors, etc.) in a single proceedings page.
- *Regularity in layout* – some proceedings put author names before the paper title, some put them below or on the same line. However, this layout is again consistent through the whole proceedings page.
- *Locality of the information* – information of the same kind is presented in a single area of the page. We can identify an area containing editors, papers, etc. The order of these areas remains the same in all the proceedings pages.
- *Textual hints* – some key phrases such as "Edited by" or "Table of Contents" are commonly used in most proceedings. When used, they may help identify the section.

The purpose of the subsequent *logical tree building* step is to extract the particular information from the contents of the tagged visual areas and to complete the logical relationships among the extracted data fields that include the relationships between the paper title and the corresponding authors and page numbers and similarly, between the workshop title and its editors, location, presented papers, etc. This process is again implemented as a set of domain-specific rules that take into account the expected cardinality of the relationships and specific presentation habits (e.g. separating author names with commas). The resulting *logical tree* consists of nodes that have a text content and a single domain-specific

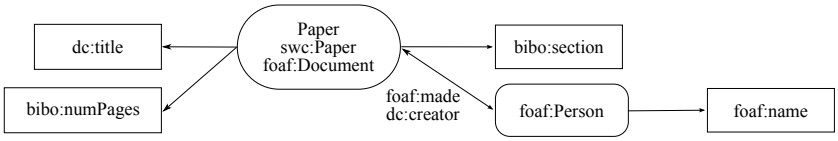



Figure 5.2: Sample ontology representing a concept (Paper) and its data and object properties. The ovals represent the object properties and the rectangles represent its data properties. [29].

tag assigned. Finally, in the CEUR mapping step, the logical tree is transformed to the target RDF description based on a specific domain ontology.

More details about the proposed approach may be found in Appendix C.1.

## 5.3 Ontology-Based Record Extraction

The structured record extraction approach presented in the previous section was tailored for the domain of workshop proceedings and it was based on a number of heuristics that are specific for this particular domain. In further research, we have been working to generalize this approach so that the whole information extraction task is specified by a conceptual model of the target domain and the goal of the extraction algorithm is to find a mapping between the concepts or properties in the conceptual model and content elements in the source document.

In [29], we proposed an approach based on ontologies. The target domain is represented by an ontology that defines relevant entities and their properties that include object properties and data properties (also called lexical properties [48]). Figure 5.2 shows a part of such an ontology for the workshop proceedings domain related to the *Paper* concept. Similarly to [48], we assume that each source document contains multiple *data records* that represent the instances of the given concept and that consist of individual *data fields*. The data fields are then substrings of the document content that correspond to the individual data properties in the ontology. Then, the information extraction task consists of finding a mapping between the data properties in the ontology and the content elements in the source document.

Our approach is based on the analysis of the visual presentation of the content again and it is a generalization of the approach described in the previous chapter that avoids the use of domain-specific heuristics. We assume that the data records are presented in a visually consistent way in the source document and we attempt to discover the most frequent visual presentation patterns that occur in the source documents and that are used for presenting the data records. This is performed in the following steps:

1. *Document preprocessing* – page rendering and segmentation. The segmentation algorithm is simplified to the extraction of *visual areas* formed by neighboring content boxes with consistent visual style; the information about larger content blocks is not useful for the following steps. We obtain a list of visual areas as the result.

2. *Initial tagging* – we assign tags to the individual visual areas that indicate the possible contents of each area. This replaces the domain-independent tagging step described in section 5.2 with a domain-specific classification of the visual areas as described below. The initial tagging is approximate due to the limited precision of the used

techniques and therefore, it is only used as initial estimation that is further refined in the next step.

3. *Tag disambiguation* – the relationships among the data fields that result from the domain model are applied in order to refine the tagging. Based on the expected structure of the data and the visual presentation of tagged data fields, we find the occurrences of the data records in the source documents that are visually consistent and correspond to the initial tagging result the most.

As we may notice, in addition to the domain ontology, which describes the expected structure of the data records to be extracted, an additional knowledge about the target domain is required, which is the specification of the possible values for each data property. In our approach, this information is represented by taggers, where a *tagger* is a procedure that is able to recognize the occurrence of the given property in the document with some precision. In other words, a tagger performs a binary classification on the visual area text content, which decides whether the given text may or may not represent a value of the given property and there is always a single tagger assigned to each data property. Different methods may be used for implementing the taggers including NER classifiers (e.g. for personal names), DBPedia concept annotation [42], visual classification as presented in chapter 4, occurrences of keywords or even regular expressions.

During the initial tagging, we mark the visual areas whose content potentially corresponds to the individual data fields. Each visual area may be assigned any number of tags based on the results of its classification using all individual taggers. It is important to note that the tagging is approximate at this stage – for example, the titles (as noted in section 5.2) are recognized based on simple regular expressions and they cannot be distinguished precisely from other content in many cases. Therefore, further refinement is performed in the next step.

In the tag disambiguation step, we consider pairs of data fields that result from the domain model (e.g. *title* – *authors* or *title* – *pages*) and we try to find a repeating visual presentation pattern in the source document that consistently presents the given pair. With consistent presentation, we understand both the *visual style consistency*, which means the same data fields are presented with the same font style in all occurrences, and the *layout consistency*, i.e. mutual positions of both fields in the pair must be consistent for all considered data records. For evaluating mutual positions of two visual areas, we define four relations $R_{side}, R_{after}, R_{below}, R_{under} \subseteq A \times A$ (where $A$ is the set of all visual areas in the document) as follows [29]:

- $(a_1, a_2) \in R_{side}$ when $a_1$ and $a_2$ are on the same line (their $y$ coordinates overlap), $a_2$ is placed to the right of $a_1$ without any other visual area being placed between $a_1$ and $a_2$ and the horizontal distance between $a_1$ and $a_2$ is not larger than 1 em[2] (shortly, $a_2$ placed next to $a_1$).
- $(a_1, a_2) \in R_{after}$ when $a_1$ and $a_2$ are on the same line and $a_2$ is placed to the right of $a_1$ anywhere on the line ($a_2$ is on the same line after $a_1$).
- $(a_1, a_2) \in R_{under}$ when $a_1$ and $a_2$ are placed roughly in the same column (their $x$ coordinates overlap) and $a_2$ is placed below $a_1$ without any other visual area being placed between $a_1$ and $a_2$ and the vertical distance between $a_1$ and $a_2$ is not larger than 0.8 em ($a_2$ is placed just below $a_1$).

[2] In typography, 1 em is a length corresponding to the point size of the current font.

- $(a_1, a_2) \in R_{below}$ when $a_1$ and $a_2$ are placed roughly in the same column (their $x$ coordinates overlap) and $a_2$ is placed anywhere below $a_1$.

For each pair of data fields, we may discover multiple visually consistent presentation patterns that are defined by the style used for presenting the two data field values (e.g. title is presented by a 12pt bold font and author is presented by a 10pt normal font) and their mutual positions (e.g. author is below the title). Such pattern represents a possible mapping between the expected structure of the data (the domain model) and the visually presented data records in the source documents. Usually, it is possible to find multiple such mappings for each property pair. Therefore, as a next step, we evaluate all discovered mappings according to their visual consistency and the number of corresponding data records in the source document and select the best performing one that will be used for the actual record extraction.

In order to demonstrate its the applicability to different domains, we have experimentally evaluated the method on the following applications:

- *Conference papers* – workshop paper data extraction from CEUR web pages as described in section 5.2. The taggers are based on regular expressions and NER classifiers.
- *Sports results* – the extraction of tennis cycling rankings from different web sources. For recognizing the occurences of the individual players (riders) and their tagging, the corresponding tagger uses the DBPedia concept annotation using DBPedia Spotlight[3].
- *Timetables* – extraction of timetable data (times, stops) from PDF documents by various publishers in very different formats. The taggers are based on a simple recognitions of numbers in the given range in the text (for tagging hours and minutes) and regular expressions (stop names).
- *News articles* – extraction of news articles from newspaper websites. For initial tagging, the visual area classification was used as described in section 4.1. Then, the published articles are treated as individual data records and we try to find the visual presentation patterns used to publish the individual parts. We already mentioned more details as a part of the main content extraction topic in section 4.2.

For each application, we prepared a simple model of the target domain and the taggers for the individual fields as mentioned above and we performed the extraction. The achieved precision and recall for the individual domains indicate a good overall applicability of the proposed methods.

Detailed and formal explanation of the whole process as well as the details of its evaluation and achieved results can be found in Appendix C.2.

## 5.4 Graph-Based Record Extraction

In [30], we further extend and generalize the approach presented in the previous section in order to make it applicable to more complex domains, where the mapping of individual pairs of data fields is not sufficient, and more complex source documents, where the visually defined boundaries of the data fields (based on consistent style of visual areas) are not

[3] https://www.dbpedia-spotlight.org/

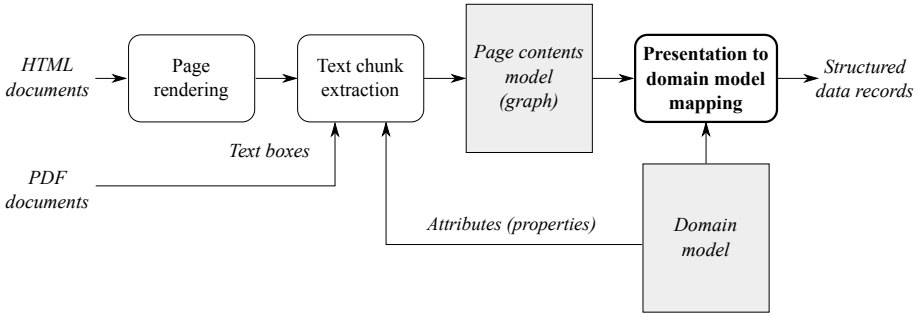



Figure 5.3: Graph-based information extraction process [30].

usable for extracting the values from the source documents. We proposed the following modifications:

- We replace the page segmentation step in the document pre-processing step with a domain-specific *chunk extraction* and we represent the document contents with the graph model introduced in section 2.4 that allows to consider several alternative interpretations of a single part of the document text. This corresponds to the observation that the same domain-specific information that is encoded in the taggers introduced in the previous section (NER classifiers, regular expressions, etc.) may not only influence the possible interpretations of the contents (which is expressed by the tag assignment) but also the way of extraction of the corresponding values from the document text. The used graph representation therefore allows to consider the domain-specific knowledge about the possible values of the individual data fields in addition to the visual properties considered in the ontology-based approach above.

- We generalize the matching of data property pairs to a general graph matching problem. We transform the domain specification to a graph as well and we formulate the information extraction task as a problem of finding the best mapping between the domain model graph at one side and the document content graph on the other side.

An overview of the whole approach is shown in Figure 5.3. In the chunk extraction step, the taggers introduced in the previous sections are used for the discovery of the substrings of the document text (which we call *chunks*) that represent the possible values of the individual data fields. A graph model of the document contents is then created from the discovered chunks as defined in section 2.4. We obtain a graph $D_c = (C, E)$ (2.6) where $C$ is the set of chunks and $E$ represents spatial relationships between pairs of chunks.

A second graph is created from the domain model. As a domain model, we assume an Entity-Relationship Diagram (ERD) $D = (E, P, R)$, where $E$ is a set of entity sets, $P$ is a set of properties (attributes in ERD) and $R \subset (E \times (E \cup P))$ is the set of relationships where each relationship has a specified cardinality. We divide the properties into groups where each group represents a set of properties that are always connected with a 1:1 relationship (including transitive relationships). The resulting *domain graph model*, is defined as a graph

$$D_g = (G, R_g) \tag{5.1}$$

where $G = \{G_1, G_2, \ldots, G_n\}$ is a set of property groups, $G_i \subset P$ and $G_i \cap G_j = \varnothing$ for any $1 \leq i, j \leq n$ and $R_g \subset G \times G$ is a set of relationships between groups. The individual

groups in $G$ and their relationships as $R$ are transformed from the domain model in the following way:

- All properties of a single entity set belong to the same group.
- If two entity sets are in a 1:1 relationship, all their properties belong to a single group.
- The 1:M relationships are transformed to the relationships between the respective groups.

Having the document contents graph $D_c$ at one side and the domain graph model $D_g$ on the other side, the next step is to find an appropriate mapping between $D_c$ and $D_g$ that maps the individual properties from $P$ to some chunks in $C$. Similarly to the ontology-based method presented above, we assume that the document contains multiple data records that are presented in a visually consistent way. The mapping has two components:

- *Group mapping* – for each group $G_i \in G$, it assigns a chunk style (as defined in (2.7)) to each property and a spatial relationship (mutual position discussed above) to each property pair.
- *Inter-group mapping* – for a pair of groups $G_i, G_j \in G$, it defines a spatial relationship between a pair of properties $p_i, p_j$, where $p_i \in G_i$ and $p_j \in G_j$.

In other words, the group mapping defines the considered visual presentation of a group of data fields in a 1:1 relationship and the inter-group mapping defines how the 1:M relationship between groups is presented in the source document. Again, multiple mappings can be found for a given source document: We consider all combinations of visual styles that are applicable to the given document and the number and visual consistency of the data records that would correspond to each mapping in order to choose the most probable one.

The details of the method are described formally in Appendix C.3.

# Chapter 6

# Conclusions

This thesis summarizes my research conducted over the past nearly 15 years in the area of using the information about visual organization and presentation of document content for extracting information from web documents. Although the technology has evolved significantly during this time, problems related to the poor accessibility of information published in web documents for computer applications are still important. In my research, I have addressed the problems of acquiring and representing information about visual presentation of document content and application of created models for page segmentation, classification and restructuring of content and detection of visual patterns with the ultimate goal of automatically recognizing and extracting desired information from documents.

The main contributions of my work in this area may can be summarized as follows: (1) We have designed ways in which information about the visual presentation of content can be acquired from different types of documents and explicitly represented so that it can be efficiently used in subsequent content processing tasks. This information includes both the basic visual organization and style of content elements directly expressed by the document code, as well as derived document layout and logical structure information obtained by page segmentation. (2) We have created new methods of extracting information from web documents based on the designed models that include identifying specific content elements, extracting the main content of a document, and finally extracting structured data records. (3) We have developed ways of integration of the web data with structured information systems based on domain models. (4) We have implemented the proposed solutions as open-source software libraries and tools (many of which have already been used in third-party applications) and we have evaluated them on real-world documents on the web.

Our practical results, such as the successful application of the developed tools in the SemPub 2015 Challenge, indicate that the use of visual presentation models allows to abstract from the implementation details of the documents and cope with the complexity and variability of their code. This makes it possible to develop information extraction methods that are applicable to large sets of heterogeneous documents in terms of their format and the particular details of their implementation.

There are many open research problems remaining. Especially in the area of structured record extraction, there are often more equivalent ways of interpreting the presented data records, and in choosing the right interpretation, human experience has so far outperformed the proposed algorithms. Similarly, assessing the significance of various anomalies in the visual presentation of content often requires some practice that our methods do not have. One possible direction for further research is, for example, the involvement of machine learning algorithms in this context. Another potentially interesting direction of research

may be the greater use of natural language processing methods and the integration of existing domain-oriented or general knowledge bases that would at least partially substitute human knowledge of the target domain.

# Bibliography


[1] Adobe Systems Incorporated: *PDF Reference, Version 1.4*. Addison-Wesley. 2002. ISBN 0-201-75839-3.

[2] Ahmadi, H.; Kong, J.: Efficient Web Browsing on Small Screens. In *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '08. New York, NY, USA: Association for Computing Machinery. 2008. ISBN 9781605581415. pp. 23–30. doi:10.1145/1385569.1385576.

[3] Akpınar, E.; Yeşilada, Y.: Vision Based Page Segmentation: Extended and Improved Algorithm. Technical Report eMINE Technical Report Deliverable 2 (D2). Middle East Technical University. Ankara, Turkey. 2012.

[4] Akpınar, M. E.; Yeşilada, Y.: Heuristic Role Detection of Visual Elements of Web Pages. In *Web Engineering*, edited by F. Daniel; P. Dolog; Q. Li. Berlin, Heidelberg: Springer Berlin Heidelberg. 2013. ISBN 978-3-642-39200-9. pp. 123–131.

[5] Akpınar, M. E.; Yeşilada, Y.: Vision Based Page Segmentation Algorithm: Extended and Perceived Success. In *Revised Selected Papers of the ICWE 2013 International Workshops on Current Trends in Web Engineering - Volume 8295*. New York, NY, USA: Springer-Verlag New York, Inc.. 2013. pp. 238–252.

[6] Alarte, J.; Insa, D.; Silva, J.: Webpage Menu Detection Based on DOM. In *SOFSEM 2017: Theory and Practice of Computer Science*, edited by B. Steffen; C. Baier; M. van den Brand; J. Eder; M. Hinchey; T. Margaria. Cham: Springer International Publishing. 2017. ISBN 978-3-319-51963-0. pp. 411–422.

[7] Alarte, J.; Insa, D.; Silva, J.; et al.: Site-Level Web Template Extraction Based on DOM Analysis. In *Perspectives of System Informatics*, edited by M. Mazzara; A. Voronkov. Cham: Springer International Publishing. 2016. ISBN 978-3-319-41579-6. pp. 36–49.

[8] Alarte, J.; Insa, D.; Silva, J.; et al.: Main Content Extraction from Heterogeneous Webpages. In *Web Information Systems Engineering – WISE 2018*, edited by H. Hacid; W. Cellary; H. Wang; H.-Y. Paik; R. Zhou. Cham: Springer International Publishing. 2018. ISBN 978-3-030-02922-7. pp. 393–407.

[9] Alassi, D.; Alhajj, R.: Effectiveness of template detection on noise reduction and websites summarization. *Information Sciences*. vol. 219. 2013: pp. 41 – 72. ISSN 0020-0255.

[10] Alcic, S.; Conrad, S.: Page segmentation by web content clustering. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*. WIMS

'11. New York, NY, USA: ACM. 2011. ISBN 978-1-4503-0148-0. pp. 24:1–24:9. doi:10.1145/1988688.1988717.

[11] Anderson, N.; Hong, J.: Visually Extracting Data Records from Query Result Pages. In *Web Technologies and Applications: 15th Asia-Pacific Web Conference, APWeb 2013, Sydney, Australia, April 4-6, 2013. Proceedings*. Berlin, Heidelberg: Springer. 2013. ISBN 978-3-642-37401-2. pp. 392–403.

[12] Anderson, N.; Hong, J.: Evaluation of Information Extraction Techniques to Label Extracted Data from E-Commerce Web Pages. In *Proceedings of the 23rd International Conference on World Wide Web*. WWW '14 Companion. New York, NY, USA: Association for Computing Machinery. 2014. ISBN 9781450327459. pp. 1275–1278. doi:10.1145/2567948.2579703.

[13] Apers, P. M. G.: Identifying Internet-related Database Research. In *East/West Database Workshop*, edited by J. Eder; L. Kalinichenko. Workshops in Computing. London: Springer. 1994.

[14] Apostolova, E.; Pourashraf, P.; Sack, J.: Digital Leafleting: Extracting Structured Data from Multimedia Online Flyers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics. May–June 2015. pp. 283–292. doi:10.3115/v1/N15-1032.

[15] Apostolova, E.; Tomuro, N.: Combining Visual and Textual Features for Information Extraction from Online Flyers. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics. October 2014. pp. 1924–1929. doi:10.3115/v1/D14-1206.

[16] Ashish, N.; Knoblock, C. A.: Wrapper Generation for Semi-structured Internet Sources. *SIGMOD Rec.*. vol. 26, no. 4. December 1997: pp. 8–15. ISSN 0163-5808. doi:10.1145/271074.271078.

[17] Atzeni, P.; Mecca, G.; Merialdo, P.: Semistructured and Structured Data in the Web: Going Back and Forth. *SIGMOD Rec.*. vol. 26, no. 4. December 1997: pp. 16–23. ISSN 0163-5808. doi:10.1145/271074.271080.

[18] Bar-Yossef, Z.; Rajagopalan, S.: Template Detection via Data Mining and Its Applications. In *Proceedings of the 11th International Conference on World Wide Web*. WWW '02. New York, NY, USA: ACM. 2002. ISBN 1-58113-449-5. pp. 580–591. doi:10.1145/511446.511522.

[19] Berners-Lee, T.; Cailliau, R.: WorldWideWeb: Proposal for a HyperText Project. Proposal. CERN. 1990.
Retrieved from: http://www.w3.org/Proposal

[20] Bing, L.; Guo, R.; Lam, W.; et al.: Web Page Segmentation with Structured Prediction and Its Application in Web Page Classification. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '14. New York, NY, USA: ACM. 2014. ISBN 978-1-4503-2257-7. pp. 767–776.

[21] Bizer, C.; Meusel, R.; Primpeli, A.: Web Data Commons - RDFa, Microdata, and Microformat Data Sets - Section 3.2 Extraction Results from the November 2018 Common Crawl Corpus. 2019. accessed on 2019-10-25.
Retrieved from: http://webdatacommons.org/structureddata/index.html#toc4

[22] Bos, B.: Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification. W3C working draft. W3C. April 2016.
http://www.w3.org/TR/2016/WD-CSS22-20160412/.

[23] Brickley, D.; Hickson, I.; Nevile, C.: HTML Microdata. W3C working draft. W3C. April 2018. https://www.w3.org/TR/2018/WD-microdata-20180426/.

[24] Bu, Z.; Zhang, C.; Xia, Z.; et al.: An FAR-SW based approach for webpage information extraction. *Information Systems Frontiers*. vol. 16, no. 5. 2014: pp. 771–785.

[25] Burget, R.: Automatic Document Structure Detection for Data Integration. In *Business Information Systems*, edited by W. Abramowicz. Berlin, Heidelberg: Springer Berlin Heidelberg. 2007. ISBN 978-3-540-72035-5. pp. 391–397.

[26] Burget, R.: Layout Based Information Extraction from HTML Documents. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2. Sep. 2007. ISSN 2379-2140. pp. 624–628. doi:10.1109/ICDAR.2007.4376990.

[27] Burget, R.: Automatic Web Document Restructuring Based on Visual Information Analysis. In *Proceedings of the 6th Atlantic Web Intelligence Conference – AWIC'2009*. Advances in Intelligent and Soft Computing , Vol. 67. Springer Verlag. 2009. ISBN 978-3-642-10686-6. page 10.

[28] Burget, R.: Visual Area Classification for Article Identification in Web Documents. In *21st International Workshop on Databases and Expert Systems Applications*. IEEE Computer Society. 2010. ISBN 978-0-7695-4174-7. pp. 171–175.

[29] Burget, R.: Information Extraction from the Web by Matching Visual Presentation Patterns. In *Knowledge Graphs and Language Technology: ISWC 2016 International Workshops: KEKI and NLP&DBpedia*. Lecture Notes in Computer Science vol. 10579. Springer International Publishing. 2017. ISBN 978-3-319-68722-3. pp. 10–26. doi:10.1007/978-3-319-68723-0_2.

[30] Burget, R.: Model-Based Integration of Unstructured Web Data Sources Using Graph Representation of Document Contents. In *15th International Conference on Web Information Systems and Technologies*. SciTePress. 2019. ISBN 978-989-758-386-5. pp. 326–333.

[31] Burget, R.; Burgetová, I.: Automatic Annotation of Online Articles Based on Visual Feature Classification. *International Journal of Intelligent Information and Database Systems*. vol. 5, no. 4. 2011: pp. 338–360. ISSN 1751-5858.

[32] Burget, R.; Rudolfová, I.: Web Page Element Classification Based on Visual Features. In *1st Asian Conference on Intelligent Information and Database Systems ACIIDS 2009*. IEEE Computer Society. 2009. ISBN 978-0-7695-3580-7. pp. 67–72. doi:10.1109/ACIIDS.2009.71.

[33] Burget, R.; Smrz, P.: Extracting Visually Presented Element Relationships from Web Documents. *International Journal of Cognitive Informatics and Natural Intelligence*. vol. 7, no. 2. April 2013: pp. 13–29. ISSN 1557-3958.

[34] Cai, D.; Yu, S.; Wen, J.-R.; et al.: *VIPS: a Vision-based Page Segmentation Algorithm*. Microsoft Research. 2003.

[35] Chakrabarti, D.; Kumar, R.; Punera, K.: A Graph-theoretic Approach to Webpage Segmentation. In *Proceedings of the 17th International Conference on World Wide Web*. WWW '08. New York, NY, USA. 2008. ISBN 978-1-60558-085-2. pp. 377–386. doi:10.1145/1367497.1367549.

[36] Ciravegna, F.: $(LP)^2$ an Adaptive Algorithm for Information Extraction from Web-related Texts. In *In Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*. 2001.

[37] Cohen, W. W.; Hurst, M.; Jensen, L. S.: A flexible learning system for wrapping tables and lists in HTML documents. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*. New York, NY, USA: ACM. 2002. ISBN 1-58113-449-5. pp. 232–241. doi:http://doi.acm.org/10.1145/511446.511477.

[38] Coondu, S.; Chattopadhyay, S.; Chattopadhyay, M.; et al.: Mobile-enabled content adaptation system for e-learning websites using segmentation algorithm. In *The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014)*. Dec 2014. ISSN null. pp. 1–8. doi:10.1109/SKIMA.2014.7083570.

[39] Cormer, M.; Mann, R.; Moffatt, K.; et al.: Towards an Improved Vision-Based Web Page Segmentation Algorithm. In *2017 14th Conference on Computer and Robot Vision (CRV)*. May 2017. pp. 345–352. doi:10.1109/CRV.2017.38.

[40] Cormier, M.; Moffatt, K.; Cohen, R.; et al.: Purely vision-based segmentation of web pages for assistive technology. *Computer Vision and Image Understanding*. vol. 2016. 2016. ISSN 1077-3142.

[41] Cowie, J.; Lehnert, W.: Information Extraction. *Commun. ACM*. vol. 39, no. 1. January 1996: pp. 80–91. ISSN 0001-0782. doi:10.1145/234173.234209.

[42] Daiber, J.; Jakob, M.; Hokamp, C.; et al.: Improving Efficiency and Accuracy in Multilingual Entity Extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*. 2013.

[43] Di Iorio, A.; Lange, C.; Dimou, A.; et al.: Semantic Publishing Challenge–Assessing the Quality of Scientific Output by Information Extraction and Interlinking. In *Semantic Web Evaluation Challenges*. Springer. 2015. pp. 65–80.

[44] Eden, T.; O'Hara, S.; Wu, X.; et al.: HTML 5.3. W3C working draft. W3C. October 2018. https://www.w3.org/TR/2018/WD-html53-20181018/.

[45] El-Shayeb, M. A.; El-Beltagy, S. R.; Rafea, A.: Extracting the Latent Hierarchical Structure of Web Documents. In *Advanced Internet Based Systems and Applications*, edited by E. Damiani; K. Yetongnon; R. Chbeir; A. Dipanda. Berlin, Heidelberg: Springer Berlin Heidelberg. 2009. ISBN 978-3-642-01350-8. pp. 305–313.

[46] Eldirdiery, H. F.; Ahmed, A. H.: Detecting and Removing Noisy Data on Web Document using Text Density Approach. *International Journal of Computer Applications*. vol. 112, no. 5. February 2015: pp. 32–36. ISSN 0975-8887.

[47] Eldirdiery, H. F.; Ahmed, A. H.: Web Document Segmentation for Better Extraction of Information: A Review. *International Journal of Computer Applications*. vol. 110, no. 3. January 2015: pp. 24–28.

[48] Embley, D. W.; Campbell, D. M.; Jiang, Y. S.; et al.: Conceptual-model-based Data Extraction from Multiple-record Web Pages. *Data Knowl. Eng.*. vol. 31, no. 3. November 1999: pp. 227–251. ISSN 0169-023X.

[49] Estuka, F.; Miller, J.: A Pure Visual Approach for Automatically Extracting and Aligning Structured Web Data. *ACM Trans. Internet Technol.*. vol. 19, no. 4. November 2019: pp. 51:1–51:26. ISSN 1533-5399. doi:10.1145/3365376.

[50] Figueiredo, L. N. L.; de Assis, G. T.; Ferreira, A. A.: DERIN: A data extraction method based on rendering information and n-gram. *Information Processing & Management*. vol. 53, no. 5. 2017: pp. 1120 – 1138. ISSN 0306-4573. doi:https://doi.org/10.1016/j.ipm.2017.04.007.

[51] Finkel, J. R.; Grenager, T.; Manning, C.: Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. ACL '05. 2005. pp. 363–370.

[52] Freitag, D.: Information Extraction from HTML: Application of a General Machine Learning Approach. In *AAAI/IAAI*. 1998. pp. 517–523.

[53] Fumarola, F.; Weninger, T.; Barber, R.; et al.: Extracting General Lists from Web Documents: A Hybrid Approach. In *Proceedings of the 24th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems Conference on Modern Approaches in Applied Intelligence - Volume Part I*. IEA/AIE'11. Berlin, Heidelberg: Springer-Verlag. 2011. ISBN 978-3-642-21821-7. pp. 285–294.

[54] Gallo, I.; Zamberletti, A.; Noce, L.: Content Extraction from Marketing Flyers. In *Computer Analysis of Images and Patterns*, edited by G. Azzopardi; N. Petkov. Cham: Springer International Publishing. 2015. ISBN 978-3-319-23192-1. pp. 325–336.

[55] Gavankar, C.; Kulkarni, A.; Ramakrishnan, G.: Efficient Reuse of Structured and Unstructured Resources for Ontology Population. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik, Iceland: European Language Resources Association (ELRA). May 2014. pp. 3654–3660.

[56] Goh, P. L.; Hong, J. L.; Tan, E. X.; et al.: Region based data extraction. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*. May 2012. pp. 1196–1200.

[57] Gottron, T.: Bridging the gap: from multi document Template Detection to single document Content Extraction. In *Proceedings of the IASTED International Conference on Internet and Multimedia Systems and Applications*. EuroIMSA 2008. ACTA Press. 2008. pp. 66–71.

[58] Guo, J.; Crescenzi, V.; Furche, T.; et al.: RED: Redundancy-Driven Data Extraction from Result Pages? In *The World Wide Web Conference*. WWW '19. New York, NY, USA: ACM. 2019. ISBN 978-1-4503-6674-8. pp. 605–615. doi:10.1145/3308558.3313529.

[59] Gupta, S.; Kaiser, G.; Neistadt, D.; et al.: DOM-based Content Extraction of HTML Documents. In *WWW2003 proceedings of the 12 Web Conference*. 2003. pp. 207–214.

[60] Hassan, T.; Baumgartner, R.: Table Recognition and Understanding from PDF Files. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2. Sep. 2007. ISSN 2379-2140. pp. 1143–1147. doi:10.1109/ICDAR.2007.4377094.

[61] Herman, I.; Sporny, M.; Adida, B.; et al.: RDFa 1.1 Primer – Third Edition. W3C note. W3C. March 2015. http://www.w3.org/TR/2015/NOTE-rdfa-primer-20150317/.

[62] Hong, J. L.; Siew, E.-G.; Egerton, S.: Information Extraction for Search Engines Using Fast Heuristic Techniques. *Data Knowl. Eng.*. vol. 69, no. 2. February 2010: pp. 169–196. ISSN 0169-023X. doi:10.1016/j.datak.2009.10.002.

[63] Hong, T. W.; Clark, K. L.: Using Grammatical Inference to Automate Information Extraction from the Web. *Lecture Notes in Computer Science*. vol. 2168. 2001: pp. 216–227.

[64] Horrocks, I.; Patel-Schneider, P. F.; Boley, H.; et al.: SWRL: A Semantic Web Rule Language. W3C member submission. W3C. May 2004. https://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.

[65] Hors, A. L.; Hegaret, P. L.; Wood, L.; et al.: *Document Object Model (DOM) Level 3 Core Specification*. The World Wide Web Consortium. 2004.

[66] Jiang, K.; Yang, Y.: Noise Reduction of Web Pages via Feature Analysis. In *Information Science and Control Engineering (ICISCE), 2015 2nd International Conference on*. April 2015. pp. 345–348.

[67] Khare, R.; Çelik, T.: Microformats: A Pragmatic Path to the Semantic Web. In *Proceedings of the 15th International Conference on World Wide Web*. WWW '06. New York, NY, USA: ACM. 2006. ISBN 1-59593-323-9. pp. 865–866. doi:10.1145/1135777.1135917.

[68] Kim, T.; Kim, S.; Choi, S.; et al.: A Machine-Learning Based Approach for Extracting Logical Structure of a Styled Document. *TIIS*. vol. 11. 2017: pp. 1043–1056.

[69] Kohlschütter, C.; Fankhauser, P.; Nejdl, W.: Boilerplate Detection Using Shallow Text Features. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*. WSDM '10. New York, NY, USA: ACM. 2010. ISBN 978-1-60558-889-6. pp. 441–450.

[70] Kohlschütter, C.; Nejdl, W.: A Densitometric Approach to Web Page Segmentation. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. CIKM '08. New York, NY, USA: ACM. 2008. ISBN 978-1-59593-991-3. pp. 1173–1182. doi:10.1145/1458082.1458237.

[71] Kolchin, M.; Kozlov, F.: A Template-Based Information Extraction from Web Sites with Unstable Markup. In *Semantic Web Evaluation Challenge*, *Communications in Computer and Information Science*, vol. 475, edited by V. Presutti; M. Stankovic; E. Cambria; I. Cantador; A. Di Iorio; T. Di Noia; C. Lange; D. Reforgiato Recupero; A. Tordai. Springer International Publishing. 2014. ISBN 978-3-319-12023-2. pp. 89–94. doi:10.1007/978-3-319-12024-9_11.

[72] Kong, J.; Barkol, O.; Bergman, R.; et al.: Web Interface Interpretation Using Graph Grammars. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*. vol. 42, no. 4. July 2012: pp. 590–602. ISSN 1094-6977.

[73] Konopnicki, D.; Shmueli, O.: Information Gathering in the World-Wide Web: The W3QL Query Language and the W3QS System. *ACM Trans. Database Syst.*. vol. 23, no. 4. December 1998: pp. 369–410. ISSN 0362-5915.

[74] Kosala, R.; Bussche, J. V. d.; Bruynooghe, M.; et al.: Information Extraction in Structured Documents Using Tree Automata Induction. In *PKDD '02: Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*. London, UK: Springer-Verlag. 2002. ISBN 3-540-44037-2. pp. 299–310.

[75] Kravchenko, A.: *BERyL: A System for Web Block Classification*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2018. ISBN 978-3-662-58039-4. pp. 61–78. doi:10.1007/978-3-662-58039-4_4.

[76] Kravchenko, A.: Large-scale holistic approach to Web block classification: assembling the jigsaws of a Web page puzzle. *World Wide Web*. vol. 22, no. 5. Sep 2019: pp. 1999–2015. ISSN 1573-1413. doi:10.1007/s11280-018-0634-6.

[77] Kulkarni, A.; Gavankar, C.; Ramakrishnan, G.; et al.: Semi-automatic dictionary curation for domain-specific ontologies. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*. Nov 2013. ISSN 2375-0197. pp. 727–734. doi:10.1109/ICTAI.2013.112.

[78] Kushmerick, N.: *Wrapper induction for information extraction*. PhD. Thesis. University of Washington. 1997.

[79] Lakshmanan, L. V. S.; Sadri, F.; Subramanian, I. N.: A declarative language for querying and restructuring the Web. In *Proceedings RIDE '96. Sixth International Workshop on Research Issues in Data Engineering*. Feb 1996. pp. 12–21.

[80] Li, L.; Zhou, A. M.; Fang, Y.; et al.: An Improved VIPS-Based Algorithm of Extracting Web Content. In *Material Science, Civil Engineering and Architecture*

*Science, Mechanical Engineering and Manufacturing Technology II*, *Applied Mechanics and Materials*, vol. 651. Trans Tech Publications. 11 2014. pp. 1806–1810.

[81] Liao, C.; Hiroi, K.; Kaji, K.; et al.: Event.Locky: System of Event-Data Extraction from Webpages based on Web Mining. *Journal of Information Processing*. vol. 25, no. 0. 2017: pp. 321–330. doi:10.2197/ipsjjip.25.321.

[82] Lima, R.; Espinasse, B.; Oliveira, H.; et al.: Information Extraction from the Web: An Ontology-Based Method Using Inductive Logic Programming. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*. Nov 2013. ISSN 2375-0197. pp. 741–748. doi:10.1109/ICTAI.2013.114.

[83] Liu, W.; Meng, X.; Meng, W.: Vision-based web data records extraction. In *ACM Ninth International Workshop on the Web and Databases (WebDB 2006)*. Chicago, Illinois, USA. 2006. pp. 20–25.

[84] Liu, W.; Meng, X.; Meng, W.: ViDE: A Vision-Based Approach for Deep Web Data Extraction. *IEEE Trans. on Knowl. and Data Eng.*. vol. 22, no. 3. March 2010: pp. 447–460. ISSN 1041-4347.

[85] Liu, X.; Lin, H.; Tian, Y.: Segmenting Webpage with Gomory-Hu Tree Based Clustering. *Journal of Software*. vol. 6, no. 12. 2011: pp. 2421–2425.

[86] Lockard, C.; Dong, X. L.; Einolghozati, A.; et al.: CERES: Distantly Supervised Relation Extraction from the Semi-structured Web. *Proc. VLDB Endow.*. vol. 11, no. 10. June 2018: pp. 1084–1096. ISSN 2150-8097.

[87] Luong, M.-T.; Nguyen, T. D.; Kan, M.-Y.: Logical Structure Recovery in Scholarly Articles with Rich Document Features. *Int. J. Digit. Library Syst.*. vol. 1, no. 4. October 2010: pp. 1–23. ISSN 1947-9077. doi:10.4018/jdls.2010100101.

[88] M. Baroni, A. K., F. Chantree; Sharoff, S.: CleanEval: a competition for cleaning Webpages. In *Proceedings of the sixth International conference on Language Resources and Evaluation (LREC 2008)*. Marrakech, Morocco. 2008.

[89] Madani, A.; Boussaid, O.; Zegour, D. E.: Semi-structured Documents Mining: A Review and Comparison. *Procedia Computer Science*. vol. 22. 2013: pp. 330 – 339. ISSN 1877-0509. 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013.

[90] Manabe, T.; Tajima, K.: Extracting Logical Hierarchical Structure of HTML Documents Based on Headings. *Proc. VLDB Endow.*. vol. 8, no. 12. August 2015: pp. 1606–1617. ISSN 2150-8097. doi:10.14778/2824032.2824058.

[91] Mendelzon, A. O.; Mihaila, G. A.; Milo, T.: Querying the World Wide Web. In *First Int. Conf. on Parallel and Distributed Information Systems (PDIS'96)*. 1997.

[92] Miao, G.; Tatemura, J.; Hsiung, W.-P.; et al.: Extracting Data Records from the Web Using Tag Path Clustering. In *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. New York, NY, USA: Association for Computing Machinery. 2009. ISBN 9781605584874. pp. 981–990. doi:10.1145/1526709.1526841.

[93] Milička, M.; Burget, R.: Web document description based on ontologies. In *2013 Second International Conference on Informatics Applications (ICIA)*. Sep. 2013. pp. 288–293.

[94] Milička, M.; Burget, R.: Information Extraction from Web Sources based on Multi-aspect Content Analysis. In *Semantic Web Evaluation Challenges, SemWebEval 2015 at ESWC 2015*, *Communications in Computer and Information Science*, vol. 2015. Springer International Publishing. 2015. ISBN 978-3-319-25517-0. ISSN 1865-0929. pp. 81–92.

[95] Milička, M.; Burget, R.: Multi-aspect Document Content Analysis using Ontological Modelling. In *Proceedings of 9th Workshop on Intelligent and Knowledge Oriented Technologies (WIKT 2014)*. Vydavateĺstvo STU. 2014. ISBN 978-80-227-4267-2. pp. 9–12.

[96] Muslea, I.; Minton, S.; Knoblock, C. A.: Hierarchical Wrapper Induction for Semistructured Information Sources. *Autonomous Agents and Multi-Agent Systems*. vol. 4, no. 1. Mar 2001: pp. 93–114. ISSN 1573-7454. doi:10.1023/A:1010022931168.

[97] Namboodiri, A.; Jain, A.: Document Structure and Layout Analysis. In *Digital Document Processing*, edited by B. B. Chaudhuri. Advances in Pattern Recognition. Springer London. 2007. ISBN 978-1-84628-726-8. pp. 29–48.

[98] Nojoumian, M.; Lethbridge, T. C.: Extracting Document Structure to Facilitate a Knowledge Base Creation for The UML Superstructure Specification. In *Proceedings of the International Conference on Information Technology*. ITNG '07. Washington, DC, USA: IEEE Computer Society. 2007. ISBN 0-7695-2776-0. pp. 393–400.

[99] Parsia, B.; Patel-Schneider, P.; Motik, B.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition). W3C recommendation. W3C. December 2012.
Retrieved from: http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/

[100] Potvin, B.; Villemaire, R.: Robust Web Data Extraction Based on Unsupervised Visual Validation. In *Intelligent Information and Database Systems*. Cham: Springer International Publishing. 2019. ISBN 978-3-030-14799-0. pp. 77–89.

[101] Quinlan, J. R.: *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.. 1993. ISBN 1-55860-238-0.

[102] Rauf, R.; Antkiewicz, M.; Czarnecki, K.: Logical structure extraction from software requirements documents. In *Requirements Engineering Conference (RE), 2011 19th IEEE International*. 2011. pp. 101 –110.

[103] Reis, D. C.; Golgher, P. B.; Silva, A. S.; et al.: Automatic Web News Extraction Using Tree Edit Distance. In *Proceedings of the 13th International Conference on World Wide Web*. WWW '04. New York, NY, USA: ACM. 2004. ISBN 1-58113-844-X. pp. 502–511. doi:10.1145/988672.988740.

[104] Safi, W.; Maurel, F.; Routoure, J.-M.; et al.: A Hybrid Segmentation of Web Pages for Vibro-Tactile Access on Touch-Screen Devices. In *3rd Workshop on Vision and Language (VL 2014) associated to 25th International Conference on Computational Linguistics (COLING 2014)*. dublin, Ireland. Aug 2014. pp. 95 – 102.

[105] Sanoja, A.; Gancarski, S.: Block-o-Matic: A web page segmentation framework. In *Multimedia Computing and Systems (ICMCS), 2014 International Conference on*. April 2014. pp. 595–600. doi:10.1109/ICMCS.2014.6911249.

[106] Sarkhel, R.; Nandi, A.: Visual Segmentation for Information Extraction from Heterogeneous Visually Rich Documents. In *Proceedings of the 2019 International Conference on Management of Data*. SIGMOD '19. New York, NY, USA: ACM. 2019. ISBN 978-1-4503-5643-5. pp. 247–262. doi:10.1145/3299869.3319867.

[107] Schulz, A.; Lässig, J.; Gaedke, M.: Practical Web Data Extraction: Are We There Yet? – A Short Survey. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. Oct 2016. pp. 562–567. doi:10.1109/WI.2016.0096.

[108] Seaborne, A.; Harris, S.: SPARQL 1.1 Query Language. W3C recommendation. W3C. March 2013. http://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[109] Shi, J.; Liu, L.: Web information extraction based on news domain ontology theory. In *Web Society (SWS), 2010 IEEE 2nd Symposium on*. Aug 2010. pp. 416–419.

[110] Shi, S.; Liu, C.; Shen, Y.; et al.: AutoRM: An effective approach for automatic Web data record mining. *Knowledge-Based Systems*. vol. 89. 2015: pp. 314–331.

[111] Shreve, G. M.: Corpus Enhancement and computer-assisted localization and Translation. In *Perspectives on Localization*, edited by K. J. Dunne. Amsterdam/Philadelphia: John Benjamins Publishing Company. 2006. pp. 309–332.

[112] Simon, K.; Lausen, G.: ViPER: Augmenting Automatic Information Extraction with Visual Perceptions. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*. CIKM '05. New York, NY, USA: Association for Computing Machinery. 2005. ISBN 1595931406. pp. 381–388. doi:10.1145/1099554.1099672.

[113] 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.

[114] Song, D.; Sun, F.; Liao, L.: A hybrid approach for content extraction with text density and visual importance of DOM nodes. *Knowledge and Information Systems*. vol. 42, no. 1. 2015: pp. 75–96. ISSN 0219-3116. doi:10.1007/s10115-013-0687-x.

[115] Song, R.; Liu, H.; Wen, J.-R.; et al.: Learning block importance models for web pages. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*. New York, NY, USA: ACM. 2004. ISBN 1-58113-844-X. pp. 203–211.

[116] Spousta, M.; Marek, M.; Pecina, P.: Victor: the Web-Page Cleaning Tool. In *Proceedings of the 4th Web as Corpus Workshop (WAC4) at the sixth International conference on Language Resources and Evaluation (LREC 2008)*. Marrakech, Morocco. 2008.

[117] Stoffel, A.; Spretke, D.; Kinnemann, H.; et al.: Enhancing document structure analysis using visual analytics. In *Proceedings of the 2010 ACM Symposium on Applied Computing*. SAC '10. New York, NY, USA: ACM. 2010. ISBN 978-1-60558-639-7. pp. 8–12.

[118] Su, W.; Wang, J.; Lochovsky, F. H.: ODE: Ontology-assisted Data Extraction. *ACM Trans. Database Syst.*. vol. 34, no. 2. July 2009: pp. 12:1–12:35. ISSN 0362-5915. doi:10.1145/1538909.1538914.

[119] Su, W.; Wang, J.; Lochovsky, F. H.; et al.: Combining Tag and Value Similarity for Data Extraction and Alignment. *IEEE Transactions on Knowledge and Data Engineering*. vol. 24, no. 7. July 2012: pp. 1186–1200. ISSN 2326-3865. doi:10.1109/TKDE.2011.66.

[120] Sun, F.; Song, D.; Liao, L.: DOM Based Content Extraction via Text Density. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '11. New York, NY, USA: ACM. 2011. ISBN 978-1-4503-0757-4. pp. 245–254. doi:10.1145/2009916.2009952.

[121] Tatsumi, Y.; Asahi, T.: Analyzing Web Page Headings Considering Various Presentation. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*. WWW '05. New York, NY, USA: ACM. 2005. ISBN 1-59593-051-5. pp. 956–957. doi:10.1145/1062745.1062816.

[122] Utiu, N.; Ionescu, V.: Learning Web Content Extraction with DOM Features. In *2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP)*. Sep. 2018. ISSN null. pp. 5–11. doi:10.1109/ICCP.2018.8516632.

[123] Uzun, E.; Agun, H. V.; Yerlikaya, T.: Web content extraction by using decision tree learning. In *2012 20th Signal Processing and Communications Applications Conference (SIU)*. April 2012. ISSN 2165-0608. pp. 1–4. doi:10.1109/SIU.2012.6204476.

[124] Uzun, E.; Agun, H. V.; Yerlikaya, T.: A hybrid approach for extracting informative content from web pages. *Information Processing & Management*. vol. 49, no. 4. 2013: pp. 928 – 944. ISSN 0306-4573.

[125] Vieira, K.; da Costa Carvalho, A. L.; Berlt, K.; et al.: On Finding Templates on Web Collections. *World Wide Web*. vol. 12, no. 2. Jun 2009: pp. 171–211. ISSN 1573-1413. doi:10.1007/s11280-009-0059-3.

[126] Vieira, K.; da Silva, A. S.; Pinto, N.; et al.: A Fast and Robust Method for Web Page Template Detection and Removal. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*. CIKM '06. New York, NY, USA: ACM. 2006. ISBN 1-59593-433-2. pp. 258–267. doi:10.1145/1183614.1183654.

[127] Vineel, G.: Web page DOM node characterization and its application to page segmentation. In *2009 IEEE International Conference on Internet Multimedia Services Architecture and Applications (IMSAA)*. Dec 2009. pp. 1–6. doi:10.1109/IMSAA.2009.5439444.

[128] Wang, J.; He, X.; Wang, C.; et al.: News Article Extraction with Template-Independent Wrapper. In *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. New York, NY, USA: Association for

Computing Machinery. 2009. ISBN 9781605584874. pp. 1085–1086. doi:10.1145/1526709.1526868.

[129] Wei, T.; Lu, Y.; Li, X.; et al.: Web page segmentation based on the Hough transform and vision cues. In *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE. 2015. pp. 865–872.

[130] Weng, D.; Hong, J.; Bell, D. A.: Extracting Data Records from Query Result Pages Based on Visual Features. In *Advances in Databases: 28th British National Conference on Databases, BNCOD 28, Manchester, UK, July 12-14, 2011, Revised Selected Papers*. Berlin, Heidelberg: Springer. 2011. ISBN 978-3-642-24577-0. pp. 140–153. doi:10.1007/978-3-642-24577-0_16.

[131] Weng, D.; Hong, J.; Bell, D. A.: Automatically Annotating Structured Web Data Using a SVM-Based Multiclass Classifier. In *Web Information Systems Engineering – WISE 2014: 15th International Conference, Thessaloniki, Greece, October 12-14, 2014, Proceedings, Part I*. Cham: Springer International Publishing. 2014. ISBN 978-3-319-11749-2. pp. 115–124. doi:10.1007/978-3-319-11749-2_9.

[132] Wood, D.; Cyganiak, R.; Lanthaler, M.: RDF 1.1 Concepts and Abstract Syntax. W3C recommendation. W3C. February 2014. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[133] Wu, S.; Hsiao, L.; Cheng, X.; et al.: Fonduer: Knowledge Base Construction from Richly Formatted Data. In *Proceedings of the 2018 International Conference on Management of Data*. SIGMOD '18. New York, NY, USA: ACM. 2018. ISBN 978-1-4503-4703-7. pp. 1301–1316. doi:10.1145/3183713.3183729.

[134] Wu, Y.-C.: Language independent web news extraction system based on text detection framework. *Information Sciences*. vol. 342. 2016: pp. 132 – 149. ISSN 0020-0255.

[135] Xu, Z.; Miller, J.: Identifying semantic blocks in Web pages using Gestalt laws of grouping. *World Wide Web*. 2015: pp. 1–22.

[136] Yang, X.; Yumer, E.; Asente, P.; et al.: Learning to Extract Semantic Structure from Documents Using Multimodal Fully Convolutional Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017. ISSN 1063-6919. pp. 4342–4351. doi:10.1109/CVPR.2017.462.

[137] Yi, L.; Liu, B.; Li, X.: Eliminating Noisy Information in Web Pages for Data Mining. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM. 2003.

[138] You, Y.; Xu, G.; Cao, J.; et al.: Leveraging Visual Features and Hierarchical Dependencies for Conference Information Extraction. In *Web Technologies and Applications*, *Lecture Notes in Computer Science*, vol. 7808, edited by Y. Ishikawa; J. Li; W. Wang; R. Zhang; W. Zhang. Springer Berlin Heidelberg. 2013. ISBN 978-3-642-37400-5. pp. 404–416. doi:10.1007/978-3-642-37401-2_41.

[139] Yu, S.; Cai, D.; Wen, J.-R.; et al.: *Improving Pseudo-Relevance Feedback in Web Information Retrieval Using Web Page Segmentation*. Microsoft Research. 2002.

[140] Yuliana, O. Y.; Chang, C.-H.: A novel alignment algorithm for effective web data extraction from singleton-item pages. *Applied Intelligence*. vol. 48, no. 11. Nov 2018: pp. 4355–4370. ISSN 1573-7497. doi:10.1007/s10489-018-1208-0.

[141] Zeleny, J.; Burget, R.: Cluster-based Page Segmentation-a Fast and Precise Method for Web Page Pre-processing. In *Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics*. WIMS '13. New York, NY, USA: ACM. 2013. ISBN 978-1-4503-1850-1. pp. 7:1–7:12.

[142] Zeleny, J.; Burget, R.: Isomorphic mapping of DOM trees for Cluster-Based Page Segmentation. In *Proceedings of the Twelfth International Conference on Informatics INFORMATICS'2013*. The University of Technology Košice. 2013. ISBN 978-80-8143-127-2. pp. 256–261.

[143] Zeleny, J.; Burget, R.: Accelerating the Process of Web Page Segmentation via Template Clustering. *Int. J. Intell. Inf. Database Syst.*. vol. 9, no. 2. March 2016: pp. 134–154. ISSN 1751-5858.

[144] Zeleny, J.; Burget, R.; Zendulka, J.: Box clustering segmentation: A new method for vision-based web page preprocessing. *Information Processing & Management*. vol. 53, no. 3. 2017: pp. 735 – 750. ISSN 0306-4573. doi:https://doi.org/10.1016/j.ipm.2017.02.002.

[145] Zeng, J.; Flanagan, B.; Hirokawa, S.; et al.: A Web Page Segmentation Approach Using Visual Semantics. *IEICE Transactions on Information and Systems*. vol. E97-D, no. 2. February 2014: pp. 223–230. ISSN 1745-1361.

[146] Zhai, Y.; Liu, B.: Web Data Extraction Based on Partial Tree Alignment. In *Proceedings of the 14th International Conference on World Wide Web*. WWW '05. New York, NY, USA: Association for Computing Machinery. 2005. ISBN 1595930469. pp. 76–85. doi:10.1145/1060745.1060761.

[147] Zhang, Q.; Hu, G.; Yue, L.: Chinese Organization Entity Recognition and Association on Web Pages. In *Business Information Systems*, edited by W. Abramowicz; D. Fensel. Berlin, Heidelberg: Springer Berlin Heidelberg. 2008. ISBN 978-3-540-79396-0. pp. 12–23.

[148] Zheng, S.; Song, R.; Wen, J.-R.: Template-Independent News Extraction Based on Visual Consistency. In *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2*. AAAI'07. AAAI Press. 2007. ISBN 9781577353232. pp. 1507–1512.

[149] Zheng, X.; Gu, Y.; Li, Y.: Data Extraction from Web Pages Based on Structural-semantic Entropy. In *Proceedings of the 21st International Conference on World Wide Web*. WWW '12 Companion. New York, NY, USA: ACM. 2012. ISBN 978-1-4503-1230-1. pp. 93–102. doi:10.1145/2187980.2187991.

[150] Zhu, W.; Dai, S.; Song, Y.; et al.: Extracting news content with visual unit of web pages. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015 16th IEEE/ACIS International Conference on*. June 2015. pp. 1–5.

# Appendix A

# Web Page Segmentation and Document Modeling

## A.1 Layout Structure Detection

Burget, R.: Layout Based Information Extraction from HTML Documents. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2. Sep. 2007. ISSN 2379-2140. pp. 624–628. doi:10.1109/ICDAR.2007.4376990.

# Layout Based Information Extraction from HTML Documents

Radek Burget
Brno University of Technology
Faculty of Information Technology
Bozetechova 2, 612 66 Brno, Czech Republic
burgetr@fit.vutbr.cz

## Abstract

*We propose a method of information extraction from HTML documents based on modelling the visual information in the document. A page segmentation algorithm is used for detecting the document layout and subsequently, the extraction process is based on the analysis of mutual positions of the detected blocks and their visual features. This approach is more robust that the traditional DOM-based methods and it opens new possibilities for the extraction task specification.*

## 1 Introduction

Regarding the growing number of documents available in the on-line repositories such as the World Wide Web or being exchanged via e-mail, the task of automatic processing of the contained data becomes important. Since most of these documents are created for human readers and they contain no or very poor explicit structure description, it is necessary to develop methods for extracting the useful information from the documents in order to allow its integration to the existing databases or information systems.

In case of HTML documents, most current approaches to information extraction work with a document representation, which directly comes from the underlying document code. The information extraction process is then based on some assumptions on the HTML usage that include the names of the tags appearing in the neighborhood of the extracted content [8] or certain shapes of the document code tree [7]. However, considering the variability of the HTML language especially when used together with the Cascading Style Sheet technology, these assumptions generally cannot be ensured. This is the main cause of the low robustness of the traditional information extraction methods from HTML documents.

In this paper, we deal with an information extraction approach, which is based on the analysis of the rendered document instead of analyzing the document code. We propose a general model of document that describes the resulting document layout and the important visual properties of the content in a form suitable for information extraction. This model is obtained by page rendering and its subsequent segmentation. Once this model is created, the information extraction process can be based on more general rules that consider the visual appearance and organization of the contents rather than the properties of the underlying code. This way, we ensure that the information extraction process is independent on the document implementation.

The proposed approach focuses on HTML documents. However, with some minor modifications, it can be applied to other document formats that explicitly contain the document text such as the PDF, PostScript or MS Word documents. On the other hand, we don't consider hand written documents and the documents that require any sort of character recognition for reconstructing the contained text content.

## 2 Information Extraction Approach

The general overview of the proposed approach is shown in Figure 1. The extraction process contains of two preprocessing phases that aim to obtaining an abstract of the processed document. The last step of the information extraction is then performed on this model.

First, the document is rendered in order to obtain its final appearance. The document code is interpreted and the rendered document is represented as a set of *boxes* according to the CSS visual formatting model [1], where a box represents a unit of the document content placed on a particular place on the resulting page. Since the boxes can be nested, we generally obtain a tree of boxes. This operation is provided by a rendering engine for the particular document format. Some output formats such as PDF or PostScript directly contain the positions of the individual boxes and therefore, no sophisticated rendering is required. On the other hand, for the HTML/CSS documents, the rendering may present

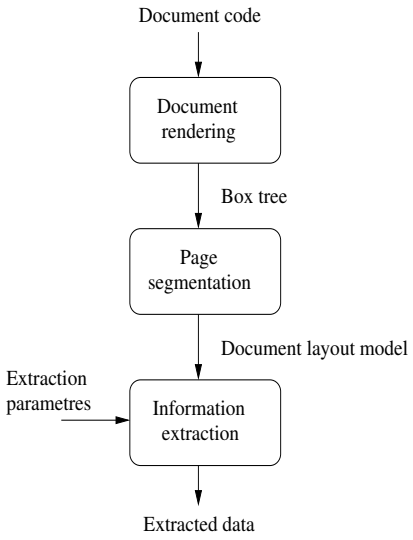



**Figure 1. General overview of the information extraction approach**

a complex task. However, several rendering engines are already available.

As the second step, we use a *segmentation algorithm* for detecting the visual organization of the document. In this phase, the visual areas such as paragraphs or columns are detected in the document including their mutual positions. This information is then represented by a document layout model together with the document content and its visual properties. We have proposed a hierarchical document layout model, which is described in section 4. In section 5, we propose the page segmentation algorithm used for this task.

As the last step, the information extraction is performed. The extraction task is described based on the mutual positions of the content and its other visual properties. This step is described in section 6.

It may seem that the way of creating the document layout model by page rendering and segmentation is unnecessarily complex since the visual areas detected by the page segmentation should directly correspond to certain elements in the HTML code. From this point of view, using a DOM tree for the same purpose seems more logical and simple. However, in practical use, the relation between the code tree and the visual appearance is uncertain and it depends on many block features that are computed as late as in time of page rendering. Therefore, we believe that analyzing the rendered document is the only approach that allows considering all the information contained in the code without any simplifying assumptions.

## 3 Related Work

The information extraction from HTML documents has been investigated for a relatively long time. Most of the work focuses on generating wrappers that identify the appropriate data directly in the HTML code. Such wrappers can be generated automatically based on various machine learning techniques [5, 8]. Modern approaches are usually based on the analysis of the document DOM tree [4].

The investigation of the document layout analysis and page segmentation methods is usually not directly related to information extraction. For HTML documents, the DOM tree analysis can be used as well [3, 7]. The most sophisticated approaches work directly with the visual representation of the rendered document [6, 2]. The advantage of this approach is that it is more general, more robust and it is not limited to the HTML documents only. Therefore, our page segmentation method proposed in section 5 is mostly based on this approach.

The page segmentation has been often investigated in the context of document transformation to a structured format (mainly the PDF documents or OCR) where the XY-Cut approach is usually used [9, 10].

## 4 Document Layout Model

The model of a document layout should describe all the information contained in the document, that can be used for information extraction. In our approach, we require that the model contains following information:

- The document content – text and images contained directly in the document code or referenced from the code
- Visual style of each part of the content such as font properties, size or color – produced by the rendering engine
- Visual organization of the document – produced by the page segmentation algorithm

### 4.1 Document Content and its Visual Properties

For representing the document content, we define the notion of *content element* $e$ as the smallest unit of the content. We distinguish two types of the content elements:

**Text element** – an atomic part of the document text with visual attributes. We define a text element as a tuple

$$e_t = (text, family, size, weight, style, variant, decoration, color)$$

| Attribute | Meaning | Values |
|---|---|---|
| $family$ | Font family | serif, sans-serif, monospace |
| $size$ | Font size | size in points (pt) |
| $weight$ | Font weight | normal, bold |
| $style$ | Font style | normal, italic |
| $variant$ | Font variant | normal, small-caps |
| $decoration$ | Text decoration | none, underline, overline, line-through |
| $color$ | Text color | Color in RGB model |

**Table 1. Visual attributes of a text element**

where $text$ is the text string that forms the element and the remaining components correspond to the visual attributes as defined in Table 1.

**Image element** – represents images in the content (for example photographs). The image element is defined as a tuple $e_i = (width, height, data)$ where the $width$ and $height$ correspond to the image size and $data$ represents the image data in some image format.

Both the text and image elements are created from the appropriate text and image boxes obtained during the page rendering. The visual attributes of the text elements are taken from the computed style of the corresponding box. From the rendered document, we obtain a set $E_t$ of text elements and a set $E_i$ of image elements in the document. Then, a continuous part of the document content can be represented as string

$$t = e_1 e_2 e_3 \dots e_n \tag{1}$$

where $e_i \in E_t \cup E_i; \forall i \in \langle 1, n \rangle$.

### 4.2 Visual Organization

The document visual organization is obtained by a page segmentation described further in section 5. During this process, a set $A$ of visual areas is detected. A *visual area* $a \in A$ represents a rectangular region in the page that is visually separated from the remaining content by any mean (for example by a frame around or by a different background). Each area may be further divided in sub-areas, i.e. the areas may be nested. Therefore, we represent the visual organization of the document by a tree of visual areas

$$T_v = (A, E) \tag{2}$$

where $E \subset A \times A$ is the set of the tree edges.

We can say that $A = A_l \cup A_n$ where $A_l$ is a set of leaf areas in the tree and $A_n$ is a set of non-leaf areas. The *non-leaf* visual areas may only contain nested visual areas. The *leaf* visual areas directly contain a part of the document content.

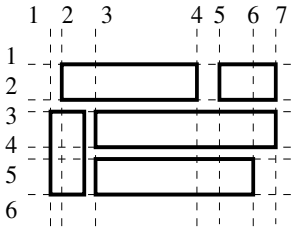



**Figure 2. Representing the mutual positions of the area in a grid**

In case of non-leaf areas, it is necessary to represent the mutual positions of the nested areas. For each non-leaf area we define a topographical grid $g$ that represents the positions of the nested areas as shown in Figure 2. The grid defines the division of the visual area to $m$ columns and $n$ rows and the width and height of each row and column. The position of each nested area is then defined by its top-left row and column in the grid of the parent area and the number of rows and columns it occupies. Therefore, a non-leaf area $a_n \in A_n$ can be defined as

$$a_n = (x, y, w, h, g) \tag{3}$$

where $x$ and $y$ is the position of the area in the parent area grid, $w$ and $h$ is the number of columns and rows the area occupies in the grid and $g$ is the new grid defined for the area itself.

The leaf areas have no grid defined since they only contain a part of the document content. We define a leaf area $a_l \in A_l$ as

$$a_l = (x, y, w, h, t) \tag{4}$$

where $x$, $y$, $w$ and $h$ have the same meaning as in (3) and $t$ is a part of the document content contained in the visual area as defined in (1).

The resulting tree of areas (2) describes all the required information about the document content – a hierarchical structure of visual areas, mutual positions of the areas and the document content together with its visual attributes.

## 5 Page Segmentation Algorithm

The purpose of the page segmentation is to discover the visual areas contained in the document and their hierarchy. We propose a segmentation algorithm based on two levels of box clustering (bottom-up approach).

The input of the algorithm is a set $B$ of boxes produced by the rendering engine. Each box is characterized by its

position and size on the resulting page. We consider a rectangular coordinate system on the page with the origin in the top left corner of the page. The position of a box $b \in B$ can be defined as $p_b = (x_b, y_b, w_b, h_b)$ where $x_b$ and $y_b$ is the position of the top-left corner of the box and $w_b$ and $h_b$ are the width and height of the box. This allows that some box $b_1$ is placed within the area of another box $b_2$. In this case, we say that $b_1$ is enclosed in $b_2$.

The segmentation algorithm works in following steps:

1. We create a tree $T_b = (B, E_b)$ from the set $B$ which represents the box nesting, that means $\forall b_1, b_2 \in B : (b_1, b_2) \in E_b$ iff $b_2$ is enclosed in $b_1$. As a result, we obtain a tree of boxes where each parent box encloses all its descendant boxes.

2. We create a tree $T_a$ of basic areas. A *basic area* is always formed by a single box from $B$ that is visually separated by one of the following means:

   - It directly contains a text or an image (leaf boxes in $T_b$)
   - It has some background color defined that is different from the background color of its parent box
   - It is separated by a visible frame at least at one side

   The tree $T_a$ is created recursively: if a box $b$ from $T_b$ is visually separated, we add a new corresponding area $a_n$ to $T_a$. Then, we recursively apply the same algorithm to the child boxes of $b$ and the eventual new areas are added to $T_a$ as the child areas of $a_n$. For each non-leaf visual area of the resulting tree, the grid positions according to (3) are computed.

3. First box clustering phase – we detect all the visual areas that are placed in the adjoining cells of the area grid and they are not visually separated from each other by the above means. Such areas are joined into a single area. This step corresponds to the detection of content blocks (for example text paragraphs) that consist of several boxes.

4. Second box clustering phase – we look for areas that are not separated but they are delimited with the visually separated areas around. We make a post-order traversal through $T_a$. For each area in $T_a$, we try to cover its children by several new areas. Let's consider an area $a_p$ from $T_a$ and its child areas $a_{p1}, a_{p2}, \ldots, a_{pn}$. We create a covering area $a_c$ and we start with $a_c = a_{p1}$. Then, we try to expand $a_c$ to all four directions in the grid of $a_p$ until we reach some visually separated areas in the grid in all the directions or until we reach the grid border. We make $a_c$ a new child of $a_p$ and we move all children of $a_p$ covered by $a_c$ to $a_c$. These steps are repeated until no further covering areas can be detected in $a_p$.

After the last step, the resulting tree corresponds to $T_v$ defined in (2).

## 6 Information Extraction

The information extraction process consists of identifying the appropriate content elements in the obtained document layout tree. As it results from (4), the content elements are only contained in the leaf nodes of the tree $T_v$ (2) of visual areas. Thus, for the identification of the appropriate content element, we can use the characteristics of the containing area and the properties of the content element itself.

Regarding the visual areas, we can consider the hierarchical relations among the areas that result from the tree model and the mutual positions of the areas at the same tree level expressed by a area grid.

In case of the content element properties, we can distinguish the content type (image or text) and in case of text elements, we can consider both the visual properties and the format of the text string – that means if it contains numbers, letters, etc.

For each piece of the extracted information, an extraction rule must be defined. In simple cases, the rules may be specified manually. In this case, the rule usually consists of a simple condition specifying the required values of some of the above mentioned properties of the content box. Since the tree model of the document layout is sufficiently general, such rules may apply for a large set of documents. We include a simple example in section 7. However, for more complex tasks, machine learning techniques can be adopted for inferring the identification rules from a set of sample documents, as usual in the information extraction area [5, 8]. Then, the rules may include probabilistic values assigned to various properties.

## 7 Experimental Testing

We have created a prototype implementation of the proposed method, which includes our own HTML/CSS rendering engine and the page segmentation. Further, we have created a simple information extraction task for extracting the image descriptions from the documents.

First, we find all the image boxes in the area tree. For simplicity, we assume that the image description is placed below the image or at the right side as common in news portals. We allow certain whitespace between the image and the description. Therefore, we try to find a visual area containing the description in the area grid in the distance of

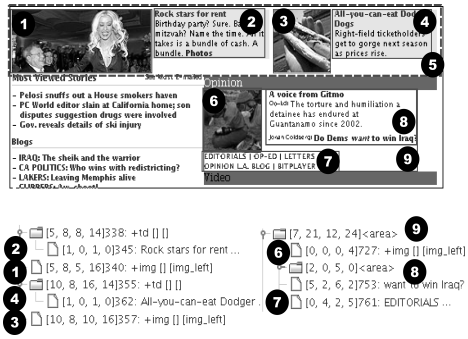



**Figure 3. Detected visual areas in the document and the corresponding parts of the area tree**

one or two cells below the image. When no appropriate area is found, we look the same way on the right of the image.

Figure 3 shows an example of a news page and the corresponding parts of the visual area tree. The important detected areas are marked with numbers. The grid positions are in the square brackets in the form $[x_1, y_1, x_2, y_2]$, where $x_1$ and $y_1$ denote the grid position of the top left corner and $x_2$ and $y_2$ the position of the bottom right corner.

The areas (1), (2), (3) and (4) share a common parent area (5). When we focus on the first image (1) $[5, 8, 5, 16]$, we are looking for a description area in the grid of its parent area. There is no area in the direction down (the parent area ends just below the image). On the right side, we find a grey rectangular $[5, 8, 8, 14]$ area that partly overlaps the image. The content of this area is assumed to be the image description and it is formed by a single sub-area (2). The same situation occurs for the image (3) and the appropriate situation (4). However, in case of the last image (6), we the menu (7) is incorrectly taken for the description instead of the better choice (8).

We have run the above extraction task on 20 web news portals such as `cnn.com`, `nytimes.com`, and similar. We have taken 10 documents from each of them. Due to the amount of the information presented, the news portal pages are quite complex with many visual sections. We have run the extraction task and compared the extraction results with the manual description identification. Although the extraction rules are extremely simple, we have reached 73% recall (available descriptions detected) and 90% precision (correct descriptions) in the description extraction.

This simple extraction task shows that the proposed information extraction method allows using general extraction rules that are not dependent on the document code. Moreover, the grid representation of the layout is suitable for specifying the relative distances between the layout blocks and their mutual positions without introducing any exact distance values that may be document-specific.

## 8 Conclusions

We have proposed a method of information extraction from HTML documents based on using the page segmentation for document preprocessing. Since the information extraction is performed on an abstract model of the document obtained by the page segmentation, it can be based on general rules that consider the document content, its layout and visual presentation. This makes the extraction method independent on the document format and more robust in comparison to the methods based on the direct analysis of the document code.

*This research was supported by the Research Plan No. MSM 0021630528 – Security-Oriented Research in Information Technology.*

## References

[1] B. Bos, H. W. Lie, C. Lilley, and I. Jacobs. *Cascading Style Sheets, level 2, CSS2 Specification*. The World Wide Web Consortium, 1998.

[2] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. *VIPS: a Vision-based Page Segmentation Algorithm*. Microsoft Research, 2003.

[3] J. Chen, B. Zhou, J. Shi, H. Zhang, and Q. Fengwu. Function-based object model towards website adaptation. In *Proceedings of the 10th International Wold Wide Web Converence*, 2001.

[4] C. Y. Chung, M. Gertz, and N. Sundaresan. Reverse engineering for web data: From visual to semantic structures. In *18th International Conference on Data Engineering*. IEEE Computer Society, 2002.

[5] D. Freitag. Information extraction from HTML: Application of a general machine learning approach. In *AAAI/IAAI*, pages 517–523, 1998.

[6] X.-D. Gu, J. Chen, W.-Y. Ma, and G.-L. Chen. Visual based content understanding towards web adaptation. In *Proc. Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 164–173, 2002.

[7] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm. Dom-based content extraction of html documents. In *WWW2003 proceedings of the 12 Web Conference*, pages 207–214, 2003.

[8] T. W. Hong and K. L. Clark. Using grammatical inference to automate information extraction from the Web. *Lecture Notes in Computer Science*, 2168:216+, 2001.

[9] Y. Ishitani. Document transformation system from papers to xml data based on pivot xml document method. *ICDAR*, 01:250, 2003.

[10] J.-L. Meunier. Optimized xy-cut for determining a page reading order. *ICDAR*, 0:347–351, 2005.

## A.2 Box Clustering Segmentation

Zeleny, J.; Burget, R.; Zendulka, J.: Box clustering segmentation: A new method for vision-based web page preprocessing. *Information Processing & Management*. vol. 53, no. 3. 2017: pp. 735 – 750. ISSN 0306-4573. doi:https://doi.org/10.1016/j.ipm.2017.02.002.

Information Processing and Management 53 (2017) 735–750

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

# Box clustering segmentation: A new method for vision-based web page preprocessing

Jan Zeleny*, Radek Burget, Jaroslav Zendulka

*Brno University of Technology, Faculty of Information Technology, Centre of Excellence IT4Innovations, Bozetechova 2, 61266 Brno, Czech Republic*

ARTICLE INFO

*Article history:*
Received 5 May 2016
Revised 1 December 2016
Accepted 2 February 2017
Available online 16 February 2017

*Keywords:*
Clustering
Segmentation
Vision-based page segmentation
VIPS

ABSTRACT

This paper presents a novel approach to web page segmentation, which is one of substantial preprocessing steps when mining data from web documents. Most of the current segmentation methods are based on algorithms that work on a tree representation of web pages (DOM tree or a hierarchical rendering model) and produce another tree structure as an output.

In contrast, our method uses a rendering engine to get an image of the web page, takes the smallest rendered elements of that image, performs clustering using a custom algorithm and produces a flat set of segments of a given granularity. For the clustering metrics, we use purely visual properties only: the distance of elements and their visual similarity.

We experimentally evaluate the properties of our algorithm by processing 2400 web pages. On this set of web pages, we prove that our algorithm is almost 90% faster than the reference algorithm. We also show that our algorithm accuracy is between 47% and 133% of the reference algorithm accuracy with indirect correlation of our algorithm's accuracy to the depth of inspected page structure. In our experiments, we also demonstrate the advantages of producing a flat segmentation structure instead of an hierarchy.



## 1. Introduction

Web page segmentation presents one of substantial preprocessing steps for data mining from web documents. There has been a lot of development in the area of web page partitioning. While some of the designed methods are targeted at specific problems like cleaning the noise from the web page, others, including page segmentation, are more generic in terms of possible utilization of their results. The problem with most of the segmentation algorithms is that they are quite slow, they depend on implementation details of the documents they process and they produce a hierarchical output that is difficult to process.

The objective this paper pursues is the development of a new web page segmentation method that is purely vision-based, independent of any HTML-related heuristics and implementation details of the processed documents. This requirement is present to make out method resilient to potential future changes in technologies used on the web. Moreover, the method should produce a flat model of the segmented page consisting of a list of visual segments with a consistent granularity level.

* Corresponding author.
*E-mail addresses:* izeleny@fit.vutbr.cz (J. Zeleny), burgetr@fit.vutbr.cz (R. Burget), zendulka@fit.vutbr.cz (J. Zendulka).

http://dx.doi.org/10.1016/j.ipm.2017.02.002
0306-4573/

And finally, the method must be unsupervised. The quality criteria against which this new method is evaluated include both speed and precision of the algorithm.

### 1.1. Background

Even though from the technological point of view, the web pages are considered as atomic carriers of information in the World Wide Web, in some research areas, it has been clear for some time (Cai, Yu, Wen, & Ma, 2003) that this granularity is too coarse for processing the contained information. Most web pages are logically split in smaller pieces. From the data mining point of view, some of these pieces can be thrown away as their informational value is negligible. Others can be then used for various purposes by different data mining techniques.

Page segmentation usually presents a preprocessing step in a more complex document processing task. From this point of view, we may find several application domains of page segmentation. Information retrieval and content classification techniques use page segmentation to improve both precision and performance by eliminating those parts of web pages that don't contain useful content Win and Thwin (2014). Separation of multiple topics in one web page is used for example in content classification. This important process can also use segmentation to gain precision Yu, Cai, Wen, and Ma (2003). In the adaptive view transformation (Aguado, 2015; Coondu, Chattopadhyay, Chattopadhyay, & Chowdhury, 2014), segmentation is used to identify coherent parts of the web page that should be kept undivided. Finally, in the information extraction area, page segmentation may be used for the identification of the data-intensive document sections Weng, Hong, and Bell (2011) or even the individual data fields Milička and Burget (2015).

Depending on the target application, different segmentation granularity may be required. The granularity corresponds to visual consistency of segments identified in the page. For the typical applications mentioned above, the following granularity levels may be considered:

- Informative content blocks level – for the page cleaning tasks in the information retrieval and document cleaning areas, the page segmentation is required to discover the basic blocks in the page such as the main content area, header, footer, etc. (Alassi & Alhajj, 2013; Uzun, Agun, & Yerlikaya, 2013; Win & Thwin, 2014; Wu, 2016).
- Paragraph level – for some applications such as vision-based classification of logical parts of the published information Burget (2010); Weng, Hong, and Bell (2014), a finer granularity is required that corresponds to the individual logical parts of the content such as headings, paragraphs, list items, etc.
- Data field level – the finest granularity level is required usually in the information extraction area when the individual data fields have to be identified and extracted Milička and Burget (2015).

Current page segmentation methods such as VIPS and its successors (described in detail in Section 2) produce a hierarchical model of the segmented page that is created by a recursive division (in case of the top-down approaches) or grouping (for the bottom-up approaches) of the detected visual blocks. The required granularity level then corresponds to the size of the leaf nodes of the produced hierarchy and for most segmentation methods, it can be adjusted by setting different parameters of the particular segmentation method such as the *degree of coherence* parameter in VIPS. However, for most of the above mentioned applications, the leaf nodes of the hierarchy are actually the most important ones. The content classification or information extraction methods examine the visual segments of the required granularity and actually do not use the complete hierarchy produced. Therefore, for several applications we have investigated recently (Burget, 2010; Milička & Burget, 2015), we found it more efficient to directly obtain a list of visual segments of the required granularity instead a hierarchical model.

In this paper, we propose the Box Clustering Segmentation (BCS) method that meets the requirements presented in the beginning of this section. Our method is built from ground up and it has virtually nothing in common with existing tree-based algorithms. We embrace a different, so far very marginally explored approach to the page segmentation problem. It is based on processing the rendered page using only very general visual cues. In contrast to the most of current methods, our algorithm does not produce a hierarchy of areas; instead, it aims to put together tiles on the same level of hierarchy. If detected correctly, the tile representation is a much more accurate representation of a web page in terms of user perception and it is more suitable for many application as discussed above. In contrast to most of the existing methods, we don't use any tree-processing approach. Instead, we rely on clustering techniques with a proper distance model in place. The simplified tile representation also allows to achieve a significantly faster segmentation which is traditionally an important issue in case of the vision-based methods.

The rest of our paper is organized as follows: Section 2 introduces the state of the art in the area of web page segmentation. Section 3 then introduces the main concept of the Box Clustering Segmentation. Sections 4, 5 and 7 explain the individual parts of the algorithm in detail. Section 6 covers the metrics we use for the clustering algorithm. Section 8 presents the results of our algorithm and compares them to the reference algorithm and finally, Sections 9 and 10 sum up the achieved results.

## 2. Related work

Our research deals with the issue of splitting up a web page into smaller segments. There has been a lot of research in this area in recent years and several types of algorithms exist to address this problem. Note that we don't compare

our method to any algorithms used for segmentation of scanned documents, as the nature of the input data is completely different.

Template detection algorithms belong to the first type. Their goal is to identify and filter out those parts of a web page that repeatedly occur on similar pages. The assumption is that these parts together constitute a template, sort of a skeleton of the page that can be dropped without loosing the relevant content (Alarte, Insa, Silva, & Tamarit, 2015; Barua, Patel, & Agrawal, 2014; Gao & Fan, 2014). Template detection methods (Kulkarni & Patil, 2014; Kulkarni & Kulkarni, 2015; Lundgren, Papapetrou, & Asker, 2015) are usually quite fast when compared with other methods for page pre-processing and they scale well. Another positive aspect is that they are usually unsupervised. The problem is that they usually need more than one web page to even work and their precision is often highly dependent on higher number of inspected pages. When considering our applications, their other problem is that they typically distinguish only between the useful content and the template (Barua et al., 2014) and they do not offer finer granularity levels.

Compared to template detection, *wrapper generation* is more similar to page segmentation. It is a procedure of creating wrappers – programs that can be later used for extracting particular areas from the given web page. From our perspective, each wrapper can be perceived as a descriptor of a particular segment on the page. However, compared to web page segmentation, literature concerned with wrappers mostly focuses only on information extraction tasks (Dalvi, Kumar, & Soliman, 2011; Ferrez, Groc, & Couto, 2013; Xiang, Yu, & Kang, 2015). The focus on information extraction is logical considering that each wrapper extracts a segment of the page.

The area of web page segmentation itself has also been researched extensively. In general, we may say that page segmentation gives the best results in terms of combination of granularity (superior to template detection) and generality of their subsequent usage (superior to both template detection and wrappers). Various methods belong to this group of algorithms. They can be divided into partially or fully supervised (Bing, Guo, Lam, Niu, & Wang, 2014; Bu, Zhang, Xia, & Wang, 2014; Fragkou, 2013) and unsupervised (Burget, 2007; Cai et al., 2003; Hong, Siew, & Egerton, 2010; Liu, Meng, & Meng, 2010; Shi, Liu, Shen, Yuan, & Huang, 2015) methods. In this work, considering the goals and applications we formulated in the introduction, we consider only the unsupervised page segmentation methods.

Depending on the underlying model used for the representation of the source documents, the web page segmentation methods are usually divided in up to four categories (Eldirdiery & Ahmed, 2015a): DOM-based approaches, text-based approaches, vision-based approaches and hybrid approaches. The DOM-based approaches (Hong et al., 2010; Jiang & Yang, 2015; Shi et al., 2015) operate on an object representation of the HTML code (Document Object Model) that represents the individual HTML elements contained in the code and their nesting. Some related information extraction (Uzun, Agun, & Yerlikaya, 2012) and document cleaning (Uzun et al., 2013; Wu, 2016) approaches share the same concept too. Because the information available in the DOM is very limited and it does not include the visual features of the individual elements, the DOM-based approaches include many heuristics that are used for an approximate estimation of the purpose of the individual HTML elements in order to identify those that form the page segments based on their typical usage in web design. Similarly, the text-based approaches (Bu et al., 2014; Eldirdiery & Ahmed, 2015b; Kohlschütter, Fankhauser, & Nejdl, 2010) focus on the properties of the text content, such as density, to detect the content segments. The segmentation methods based on both the DOM-based and text-based approaches are typically very fast because no complex document preprocessing (such as style analysis or rendering) is required. On the other hand, they operate on a simple approximation of the document based on its code and/or text content and therefore, the accuracy of the segmentation[1] greatly depends on the code properties and the used heuristics.

The vision-based approaches focus on the analysis of visual features of the document contents as they are perceived by a human reader. In case of HTML documents, obtaining the necessary visual information requires processing the document by an HTML rendering engine in order to compute the style and layout of the individual elements. Considering the visual information allows to achieve a higher segmentation accuracy in comparison to the DOM-based approaches. On the other hand, the necessity of page rendering and more complex document models processed typically in multiple steps make the vision-based approaches significantly slower and less scalable than the DOM-based ones.

VIPS (Cai et al., 2003) is probably one of the first and most popular vision-based algorithms. Even some template detection algorithms use VIPS instead of the usual DOM-based approach for detecting the visual structure of the page that is further analyzed in order to distinguish the template from the content (Alassi & Alhajj, 2013; Krishna & Dattatraya, 2015). The VIPS algorithm operates in the following steps: first, the page is divided into visual blocks; then, visual separators are discovered in the page and finally, the resulting page structure is constructed. Although the visual features of the individual elements (such as font sizes and colors) and their positions in the rendered page are taken into account, mainly the first visual block extraction step depends on a number of heuristic rules that are based on the underlying DOM and the HTML elements. Therefore, the VIPS method is sometimes considered as a DOM-based method with visual cues (Zeng, Flanagan, Hirokawa, & Ito, 2014). Since the heuristic rules strongly depend on particular usage of certain HTML elements, the VIPS method is no more directly usable for current web pages due to the evolution of web design techniques and the HTML language itself. Therefore, many extensions have been proposed in order to overcome these limitations: Akpinar and Yesilada (2012, 2013) extend the set of heuristic rules in order to cover the new tags introduced in modern versions of the HTML language and the new web design techniques. Li, Zhou, Fang, Liu, and Wu (2014) employ text statistics in order to recognize

[1] The accuracy is usually defined as a consistency between the result and human perception of the page.

similar visual blocks and Zhu, Dai, Song, and Lu (2015) use text features instead of the particular HTML tags in some VIPS heuristics. The latest approaches avoid the usage of the heuristics completely by using alternative ways of block detections. Burget (2007) uses a bottom-up grouping of visually aligned blocks, Alcic and Conrad (2011) use a clustering technique based on several distance metrics such as DOM-based, geometric and semantic distance. Liu, Lin, and Tian (2011) construct a graph of spatial relationships among the rendered elements that is later partitioned with a Gomory-Hu clustering algorithm and Zeng et al. (2014) compute a seam degree and content similarity of the individual blocks in order to divide larger visual blocks to smaller parts. Finally, Xu and Miller (2015) apply the Gestalt laws of grouping on the extracted visual blocks.

In contrast to VIPS and its successors, other vision-based approaches use entirely graphical representation of the input document that allows to abstract from the HTML-related implementation details. Cormier, Moffatt, Cohen, and Mann (2016) use an edge detection algorithm for detecting the visual separators between the content blocks. Similarly, Wei, Lu, Li, and Liu (2015) use Hough transform for the same purpose and Kong et al. (2012) recognize atomic objects using image processing methods and perform their grouping by using a spatial graph grammar.

The hybrid approaches combine the DOM-based and vision-based ones in order to obtain higher segmentation accuracy or for specific applications. Sanoja and Ganarski (2014) and Manabe and Tajima (2015) both combine the content structure (DOM) with the visual information obtained from a web browser in order to increase the accuracy in comparison to the VIPS algorithm. Safi, Maurel, Routoure, Beust, and Dias (2014) process the input document in two steps: first, a visual information analysis is performed and in the second step, DOM tree filtering is performed based on the analysis results with the aim of supporting visually impaired users. Finally, Fumarola, Weninger, Barber, Malerba, and Han (2011) combine the DOM with a visual information model in order to extract visually presented lists from the input documents. More generic content extraction use case is presented by Song, Sun, and Liao (2015)

In our Box Clustering Segmentation method, we strictly avoid using DOM and the HTML-based heuristics. We use a purely visual representation of the documents which makes our method closer to other methods based on the graphical document representation (Cormier et al., 2016; Wei et al., 2015). On the other hand, we don't detect the visual separators explicitly and the clustering approach is closer to the Web Content Clustering by Alcic and Conrad (2011).

Comparing our Box Clustering Segmentation to the VIPS and the Web Content Clustering, we find two major advantages of our approach. First, strictly utilizing just the visual information gives our algorithm an advantage of being more robust, as it doesn't depend on features of DOM model or HTML language which are subject to change. In the context of web page segmentation, this feature is especially visible on highly dynamic web pages where the DOM tree may actually be quite misleading because there is a lot of relative and absolute positioning utilized on these web pages. That makes the resulting positions and the relations between the visual areas quite different from the relations between their respective DOM nodes. Out of the context of web page segmentation, the advantage of our method it that it is applicable on other documents than web pages; it can be used on any document where we can retrieve information about position, size and color of all the elements with some content (e.g. images and text bounding boxes), which is true for example for most PDF documents available on the web. Second, we find our flat area model much more comprehensible and convenient for further processing than the tree model produced by VIPS. This is further discussed in Section 9.

## 3. Box clustering segmentation

The Box Clustering Segmentation (BCS) is designed to be a pure vision-based method. Also, in contrast to other segmentation methods, BCS is designed to give flat results. That means, it produces a tiled arrangement of segments rather than their hierarchy, which is the usual layout of segmentation results.

The entire process of the BCS is outlined in Fig. 1. Each box in the figure represents a state of the data being processed. Each transition between the states then represents an action that is taken. The first box marked *Web page* represents the input of the entire algorithm – a web page in a form of the HTML code or the corresponding DOM tree. The *rendering* step is done outside of the BCS and is therefore partially independent; any rendering engine can be used for this action. In our BCS implementation, we use the CSSBox rendering engine[2], as it offers the most convenient application interface for accessing the rendered page model. The rendering result is represented as a *rendering tree* that describes the final appearance of the rendered page as described in Section 4.

The Box Clustering Segmentation itself consists of three steps. The first one, called *box extraction*, takes the rendering tree and filters out those parts of the tree that are not useful for the subsequent clustering. The distances between the remaining boxes are computed in the second step based on the criteria described below. Several functions are created as a product of the distance computation. The *clustering* step finally processes the entities and identifies segments of the web page by clustering boxes belonging to the same segment.

## 4. Box extraction

The box extraction is the first step of the Box Clustering Segmentation. It is possible to think of it as a preprocessing step. Before we explain in detail what takes place during the preprocessing, let us inspect the rendering step and its output.

[2] http://cssbox.sourceforge.net/.

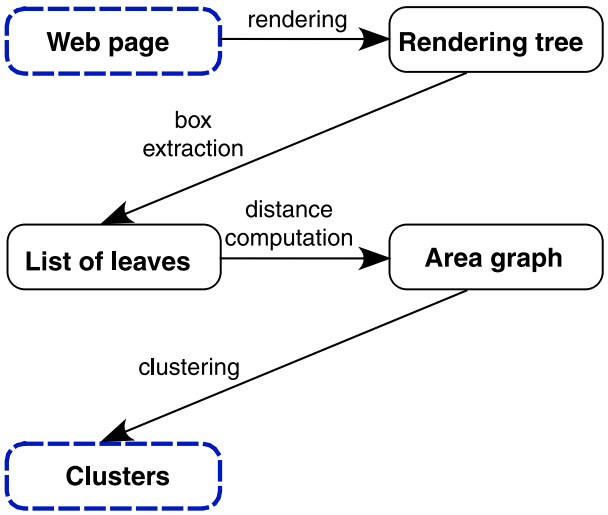



**Fig. 1.** Architecture of Box Clustering Segmentation.

A rendering engine generally transforms the input document represented as a DOM tree accompanied by Cascading Style Sheet (CSS) definitions and other additional data to a visual formatting model that is suitable for displaying the resulting page. The visual formatting model itself and the way how it is created is defined by the Cascading Style Sheet Specification (Bos, Celik, Hickson, & Lie, 2011). It is basically a tree of *boxes* where each box represents a rectangular area in the rendered page. We call this tree a *rendering tree* in this paper. Each box in the rendering tree corresponds to a particular node in the input DOM tree or its part; i.e. it is always possible to identify the source DOM node which generated that particular box. The leaf nodes of the rendering tree are elementary visual boxes that represent atomic units of the content, for example lines of text. Non-leaf nodes correspond to elements in the source DOM and they serve as either wrappers or groups of the visual boxes, for example paragraphs.

The box extraction algorithm performs a pre-order traversal of the rendering tree during which it selects the boxes that will be used in the next steps of the BCS. Among other things, each box contains some basic information that is necessary for computing the box similarity in the next clustering step. This information includes:

- The color of the box
- Box position in the page
- The size and shape of the box (derived from its width and height)

The idea of the box selection is to consider only those boxes that are actually visually rendered in the page. This is usually true for the leaf nodes of the rendering tree, as they represent the actual content of the page. However, several exceptions from this rule exist. The following list provides an explanation of the box extraction process:

1. **Text nodes** are always leaf nodes. They contain a line or its part. Graphically, each text box is a minimal bounding box of the text contained. These nodes are always selected.
2. **Image nodes** are always leaf nodes. They represent a particular image and therefore, they share its properties like position and size. These nodes are always selected.
3. **Childless boxes** that don't fall into previous categories are omitted.
4. **One-child boxes** that don't fall into previous categories are viewed as subtrees rooted at the child box. These subtrees are then inspected for branches and if no branches exist, the smallest box in the subtree with a non-transparent background is selected. If there is no such box, the leaf box is selected.

After the traversal is completed, it is remotely possible that some extracted boxes will visually contain other boxes. If this happens, all such cases are identified and the larger boxes are deselected.

## 5. Connecting the boxes

After all useful boxes are selected, we virtually connect them by detecting their adjacency. The first step to do that is to detect their semi-alignment. This is one of the important elements in the design of Box Clustering Segmentation. The following definitions describe box structure and semi-alignment of two boxes.

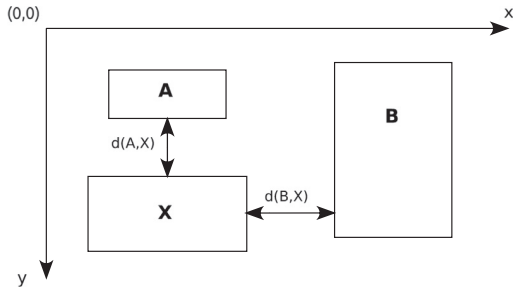



**Fig. 2.** Absolute distance measurement between boxes.

**Definition 1** (Box structure)**.** Let the box be defined as seven-tuple $m = (left,\ right,\ top,\ bottom,\ width,\ height,\ color)$ where *left, right, top* and *bottom* are integer values that represent positions of respective edges of the box; $width = right - left$; $height = bottom - top$ and *color* represents dominating color of the box in RGB format.

**Definition 2** (Projected overlap and semi-alignment)**.** Let $m$ and $n$ be two boxes on a web page. The projected overlap of boxes $m$ and $n$ is defined as a function $pov$: $(m,\ n) \rightarrow \{x,\ y,\ o\}$ where $x$ and $y$ indicate projected overlap on the respective coordinate axes of the web page and $o$ designates "no projected overlap". The following rules apply:

$$pov(m,n) = \begin{cases} x & \text{if} \quad m.right \geq n.left \wedge m.left \leq n.right \\ y & \text{if} \quad m.bottom \geq n.top \wedge m.top \leq n.bottom \\ o & \quad \text{otherwise} \end{cases} \tag{1}$$

The two boxes $m$ and $n$ are in semi-alignment if $pov(m,\ n) \neq o$.

Mutual position is a finer grained version of the projected overlap that is used when determining the distance between boxes.

**Definition 3** (Mutual position of two boxes)**.** Let a set of possible positions be $P = \{a, b, l, r, o\}$ where *a, b, l, r* designate position above, below, left and right respectively and $o$ designates "other position". The position of box $m$ relative to box $n$ is defined as a function $pos$: $(m,\ n) \rightarrow P$ where the following rules apply:

$$pos(m,n) = \begin{cases} a & \text{if} \quad m.bottom \leq n.top \wedge pov(m,n) = x \\ b & \text{if} \quad m.top \geq n.bottom \wedge pov(m,n) = x \\ l & \text{if} \quad m.right \leq n.left \wedge pov(m,n) = y \\ r & \text{if} \quad m.left \geq n.right \wedge pov(m,n) = y \\ o & \quad \text{otherwise} \end{cases} \tag{2}$$

Besides being used for defining the neighborhood of each box, semi-alignment is also used for limiting the domain of the similarity function as described in Section 6. There are two reasons for limiting both the domain of the similarity function and the number of boxes included in the neighborhood detection. The first one is purely practical – it is easier for the clustering algorithm to extract the neighboring boxes when the number of candidates is limited. The same applies for calculating the similarity – it helps to reduce the number of similarity calculations. The second reason is based on our observation that boxes that are visually related are always organized this way (placed right next to each other or right below each other).

Now, for the adjacency itself. In this paper, we use a term *direct neighborhood* to express a set of boxes adjacent to a specific box. To understand the direct neighborhood, it's important to know how absolute distances between boxes are calculated. These distances, graphically outlined in Fig. 2, are formally expressed by the function *abs*() that is included in Definition 4.

**Definition 4** (Absolute Distance, Direct Neighborhood of a Box)**.** Let $B$ be a set of boxes on a web page and let $m,\ n \in B$. Absolute distance between two boxes is a function $abs : B \times B \rightarrow \mathbb{R}$:

$$abs(m,n) = \begin{cases} n.top - m.bottom & \text{if } pos(m,n) = a \\ m.top - n.bottom & \text{if } pos(m,n) = b \\ n.left - m.right & \text{if } pos(m,n) = l \\ m.left - n.right & \text{if } pos(m,n) = r \\ \infty & \text{otherwise} \end{cases} \tag{3}$$

Direct neighborhood of a box $m$ is defined as $N_m = \{n | n \in B \wedge pos(m,n) \neq o \wedge \nexists k \in B : (pos(m,k) = pos(m,n) \wedge abs(m,k) < abs(m,n))\}$

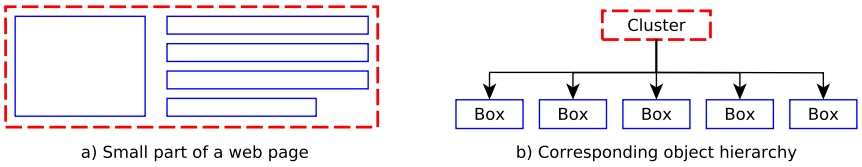



**Fig. 3.** Hierarchy of clusters and the corresponding boxes.

Direct neighborhoods of all the boxes in a web page essentially create virtual connections that are being used in further calculations and, subsequently, the clustering algorithm itself.

## 6. Similarity model

A proper model for evaluating the similarity of elements in a web page is a core piece of any segmentation algorithm. In this paper, we use a compound similarity model that consists of two parts. To understand both of them, it's necessary to understand the organization of elements in a web page as we represent it.

Boxes have already been described in Section 4. A *cluster* is the second element type that we use in BCS. Together, the two types of elements form a two-level hierarchy as outlined in Fig. 3.

As stated before, we use two different similarity metrics. The first one, called *base similarity*, is based on the visual features of boxes. The second one, called *cluster similarity*, is then used to express the similarity between two elements where at least one of them is a cluster.

### 6.1. Base similarity

The base similarity can be calculated for any pair of boxes. However, for the reasons explained in Section 5 we calculate it only for those pairs that are semi-aligned. We have chosen a simple similarity model based on several visual properties of the compared boxes. The reason for this choice was to make the algorithm both transferrable to other types of documents and resilient to any potential HTML, CSS or DOM changes in the future.

The base similarity is essentially an arithmetic mean of three components that are described further: *distance, shape similarity* and *color similarity*:

$$bsim(m,n) = \begin{cases} 0 & \text{if } distance(m,n) = 0 \\ 1 & \text{if } distance(m,n) = 1 \\ \dfrac{\begin{pmatrix} distance(m,n) + \\ sim\_shape(m,n) + \\ sim\_color(m,n) \end{pmatrix}}{3} & \text{otherwise} \end{cases} \tag{4}$$

#### 6.1.1. Distance

Distance between elements of a web page is the primary way how web designers separate the visual and semantic blocks in the web page. As such, it is also often used as the primary indicator of the separation in segmentation algorithms, including VIPS Cai et al. (2003). Our distance model is based on some ideas that come from our prior work Burget (2007). In the current model, we use relative distances that are computed in two steps. The first step – detecting direct neighborhood – was already covered in Section 5. Using the direct neighborhood, we then transform the absolute distances to relative distances, expressed by the *distance*() function.

**Definition 5** (Relative distance)**.** Let $B$ be a set of all boxes on a web page. For each box $m \in B$ and its direct neighborhood $N_m$, there is a maximal neighborhood distance $maxd(m) = abs(m,k)$ where $k \in N_m \wedge \nexists l \in N_m$: $abs(m,l) > abs(m,k)$. For each $n \in N_m$ the relative distance, designated $distance(m,n)$, is calculated as:

$$rel_m(m,n) = \frac{abs(m,n)}{maxd(m)} \tag{5}$$

$$rel_n(m,n) = \frac{abs(m,n)}{maxd(n)} \tag{6}$$

$$distance(m,n) = \frac{rel_m + rel_n}{2} \tag{7}$$

**Table 1**
Colors that are assigned to basic box types.

| Box type | Assigned color |
|---|---|
| Images | Color tone of the image |
| Text | Font color of the text |
| Other leaf boxes | Background color of the box |

The relative distance expresses how far the two boxes are from each other in context of their direct neighborhoods. The function as formulated in Definition 5 can obviously assume values in the range $< 0, 1 >$ with direct correlation between the distance and the value $rel(m, n)$.

*6.1.2. Shape*

When comparing the shapes of two boxes, we base our calculation on a premise that the boxes that look similar are likely to belong to the same cluster. This metric was included after our observation that the shape similarity is often what visually binds together blocks of text or items in menus. On the other hand, some other cases exist in which we don't necessarily want to group the boxes that are similar in some way. To distinguish between the different types of similarity, we came up with a system utilizing the aspect ratio and the size of the two compared boxes.

Let's have two boxes $m, n \in B$. For the aspect ratio comparison of the boxes, we use the following formulas:

$$r_m = \frac{m.width}{m.height} \tag{8}$$

$$r_n = \frac{n.width}{n.height} \tag{9}$$

$$ratio(m,n) = \frac{max\{r_m, r_n\} - min\{r_m, r_n\}}{\frac{max\{r_m, r_n\}^2 - 1}{max\{r_m, r_n\}}} \tag{10}$$

The second part of shape similarity measurement is the size comparison. We use the following formulas for evaluating the size similarity of two boxes $m, n \in V$:

$$s_m = m.width * m.height \tag{11}$$

$$s_n = n.width * n.height \tag{12}$$

$$size(m,n) = 1 - \frac{min\{s_m, s_n\}}{max\{s_m, s_n\}} \tag{13}$$

The final shape similarity is simply calculated as a mean value of *ratio* and *size*:

$$sim\_shape(m,n) = \frac{ratio(m,n) + size(m,n)}{2} \tag{14}$$

*6.1.3. Color*

Color difference presents another method that both web developers and segmentation algorithms use to separate visual segments of web. The boxes in a web page may contain many colors such as the text (foreground) color, background color, borders or even more colors in case of images. For computing the color distance of two boxes, we assign each box a single color depending on its type as shown in Table 1.

The color distance itself is a metric used to quantify the difference between two colors. It is commonly denoted as $\Delta E$ and there exist many formulas to calculate it. The actual choice of the most suitable formula depends on the application (Sharma, 2004).

The International Commission on Illumination came up with several formulas that work on *Lab* and *LCH* color spaces. They are all based on the fact that the human eye is more sensitive to changes in chroma than to changes in lightness (Sharma, 2004). As opposed to *RGB* color space, both *Lab* and *LCH* color spaces allow a separate calculation for lightness and chroma.

In our application, we have experimented with both *Lab* and *LCH* based color distances; however, we have evaluated a simple euclidean *RGB*-based difference as the one with the best results. Our observations showed that the reason is most likely that the web designers most often use different hue rather than chroma to distinguish between the components that don't belong to each other.

Because the results of the color distance have to match all the other components of the box comparison, we have normalized the euclidean distance by dividing it by the maximal diagonal distance in the *RGB* color space. In our color representation, we use the standard *RGB* model where each color channel can assume values in the range $< 0, 1 >$ and the maximal diagonal distance is $\sqrt{3}$.

**Definition 6** (Color similarity)**.** Let *m, n* be two boxes and let $m.color = (R_m, G_m, B_m)$ and $n.color = (R_n, G_n, B_n)$ be their color representations. The color similarity *sim_color* is defined as

$$sim\_color = \frac{\sqrt{(R_n - R_m)^2 + (G_n - G_m)^2 + (B_n - B_m)^2}}{\sqrt{3}} \tag{15}$$

### 6.2. Cluster similarity

When grouping single boxes into clusters, it is necessary to extend the similarity model to accommodate the clusters; that means, we need to evaluate the similarity of two clusters or a cluster and a box. Unfortunately, the characteristics of clusters cannot be simply inherited from the characteristics of the individual boxes constituting the clusters, mainly as it is difficult to interpret the contribution of individual boxes to the whole cluster.

Therefore, we use a model that is based on clusters' inner similarity indicators. This model builds on base similarity and follows the idea of Degree of Coherence in VIPS and box clustering in Burget (2007). The inner similarity is basically a mean value of base similarity that is calculated using the boxes within a cluster. As a prerequisite of this representation, the definition of direct neighborhood must be extended so it can accommodate both clusters and boxes. The model derives direct neighborhood of each cluster from direct neighborhoods of all the boxes contained in that cluster.

**Definition 7** (Unclustered Boxes, Cluster Direct Neighborhood)**.** Let $B$ and $C$ respectively be sets of boxes and clusters on a web page. Furthermore, let $B_c \in C$ be a set of boxes constituting cluster $c$ and let $N_m$ designate direct neighborhood of box $m$. A set of unclustered boxes on the page is defined as $B_U = \{b | b \in B; \nexists B_c \in C : (b \in B_c)\}$.

For a cluster $c$, its direct neighborhood is defined as $N_c = \{m | m \in B_U \wedge \exists n \in B_c : n \in N_m\} \cup \{B_d | B_d \in C \wedge \exists m \in B_c : \exists n \in B_d : n \in N_m\}$.

Corresponding to the previous definition, the value of similarity between a cluster and any entity in its direct neighborhood represents the mean value of similarities between that entity and all the boxes contained in the cluster. The previous text implies only connections between pairs of boxes that are adjacent where each such pair has a corresponding similarity value. However, the concepts of connection cardinality and cumulative similarity introduce additional functions *card*() and *cumul*() which are defined as follows:

**Definition 8** (Connection cardinality and cumulative similarity)**.** Let $B$, $B_U$ and $C$ respectively be sets of boxes, unclustered boxes and clusters on a web page. Also, let $N_e$ designate direct neighborhood of entity $e$. Functions $card : C \times C \cup B_U \rightarrow \mathbb{N}$ and $cumul : C \times C \cup B_U \rightarrow \mathbb{R}$ respectively represent connection cardinality and cumulative similarity. Note that the cumulative similarity uses the function $s()$ which is defined in Section 6.3. Both functions are defined as follows:

$$card(c, e) = |\{m | m \in B_c \wedge e \in N_m\}| \tag{16}$$

$$cumul(c, e) = \sum_{\forall m \in B_c} s(m, e) \tag{17}$$

When a cluster is being created, all the unclustered boxes are scanned and those ones that are about to become the cluster neighbors are selected for processing. For each of these future neighbors, the value of the cumulative similarity and the cardinality is calculated and the final similarity *csim* is then calculated as their quotient:

$$csim(c, b) = \frac{cumul(c, b)}{card(c, b)} \tag{18}$$

### 6.3. Entity similarity

Now when both compounds of the similarity model are described, their combined usage is straightforward:

**Definition 9** (Entity similarity)**.** Let $B$ and $C$ be sets of boxes and clusters on a web page respectively and let $e_1, e_2 \in B \cup C$ be two entities. The entity similarity $s$ is defined as:

$$s(e_1, e_2) = \begin{cases} bsim(e_1, e_2) & \text{if } e_1 \in B \wedge e_2 \in B \\ csim(e_1, e_2) & \text{if } e_1 \in C \\ csim(e_2, e_1) & \text{if } e_1 \notin C \wedge e_2 \in C \end{cases} \tag{19}$$

## 7. Clustering

The set of boxes $B$ and the value of a *Clustering Threshold*, designated $CT$ and described further, are the only inputs of the clustering algorithm. It outputs a set of identified clusters $C$. The clustering algorithm has four main parts that will be closely described in this section:

1. Creation of cluster seeds
2. Entity selection for merging – creation of candidate clusters
3. Overlap handling
4. Cluster verification and commission

The main algorithm loop and its initialization are shown in Fig. 1. The main loop itself covers the first two parts of the entire algorithm – the creation of the cluster seeds and the selection of entities for further merging. They are almost equivalent; the only difference is in the type of their input entities. The idea is to find the most similar couples of boxes and then, to select them for merging. If at least one of the entities is a cluster, a new candidate cluster is created. If both entities are boxes, a new cluster seed is created instead. However, even the seed should be considered just as a candidate seed at first. After it is committed, it becomes a valid cluster seed.

The selection of the two entities for merging is performed on line 6 of the Algorithm 1. The selected relation is then removed from the set to avoid infinite loops. With the two entities selected, we have to check if the similarity between them is within an acceptable range before the candidate is created. This is simple for two boxes; however, for clusters, we have to employ the formula from Section 6.2 to get the real value of similarity. Picking the right similarity value is abstracted by function *sim*(). If the difference between the selected entities is too big, the candidate is not created at all.

**Algorithm 1** The clustering algorithm.

```
1:  function BCS(IN: CT, IN OUT: G, OUT: C)
2:      loop
3:          if |B_U| < 2 then
4:              return
5:          end if
6:          m, n ← m, n ∈ B_U ∪ C : ∄x, y ∈ B_U ∪ C : (s(x, y) < s(m, n))
7:          if s(m, n) > CT then
8:              return
9:          end if
10:         create cluster candidate cc
11:         if ∃c ∈ C : c overlaps cc then
12:             continue
13:         end if
14:         if ∃b ∈ B : b overlaps cc then
15:             MERGEOVERLAPS(B, cc)
16:         end if
17:         COMMIT(cc, G, C)
18:     end loop
19: end function
```

The Clustering Threshold $CT$, used on line 7, is a static real number that can assume values in the range of $< 0, 1 >$. The Clustering Threshold corresponds to the Permitted Degree of Coherence (PDoC) used in VIPS algorithm. It has to be set in advance and it remains constant for the entire page. Picking the right value of $CT$ is a difficult task and every web page has a different optimal value. If it's too low, many boxes will end up unclustered. On the other hand, if picked too high, some clusters that should be separate are merged instead. Compared to VIPS, there is also another consequence: The results can also look completely different with different values of $CT$. That is caused by the overlap merging phase. Selecting the right value of $CT$ for the web page is out of scope of this paper, much like VIPS doesn't cover selection of the PDoC. In practical applications, we assume several approaches, for example an iterative or bisective approach with the number of unclustered boxes being used as an indicator when to stop. This solution is feasible with the support of our Cluster-based page segmentation (Zeleny & Burget, 2013) that can record the optimal value for one page and re-use it on other web pages that are similar.

Some tests are performed on the new candidate after it is created. These tests are the reason we create just a candidate – failing the tests prevents the cluster creation from being verified and in such cases it is easier to dispose of the cluster candidate than to undo the merging step.

1. An overlap with another cluster. Such overlaps are not allowed by definition in our method and therefore, if the candidate overlaps with another cluster, it is marked as invalid and it is removed.

2. An overlap with other boxes. This step is one of the most important ones. Both creating the cluster seed and extending the cluster by merging it with a nearby box can make the cluster overlap with other boxes. The overlap with a box can eventually end up being an overlap with another cluster. Since this is not acceptable, we have to consider the overlapped box or boxes for merging into the cluster right at the moment when the overlap is created. If the candidate boxes don't cause any new overlaps, they are accepted and merged into the cluster. Otherwise, the cluster candidate is removed as invalid. This is represented by the `mergeOverlaps()` function.

The entire cluster creation is then encapsulated in the function `commit()`. If the cluster candidate passes all the tests and it is not marked as invalid, it is marked as verified and committed to be a valid cluster. Several partial actions have to be carried out during this operation:

1. All new boxes in the candidate cluster are marked as members of the final cluster.
2. The inner similarity indicators of the cluster are re-calculated.
3. If the cluster candidate was created by merging other clusters, these clusters are deleted from the cluster set $C$ and removed.
4. The new cluster is added to the cluster set $C$.

After the processing of all the boxes is finished, some boxes that don't belong to any group may still remain. These boxes are ignored in the result as they are likely not important in the web page in terms of information retrieval. If the usage context of the algorithm is content filtering, these boxes can be safely dropped.

## 8. Experimental evaluation

In order to verify our design, we have created a reference implementation. We use Java as a platform for the implementation for the reasons explained further. As a rendering engine, we use CSSBox that is written in Java. CSSBox is the most capable rendering engine in terms of access to internals and the platform independence.

The goal of the experimental evaluation is to compare our implementation with the existing algorithms. We use VIPS algorithm as a baseline. To make the comparison as accurate as possible, we use Java implementation of VIPS[3], which is based on the original paper (Cai et al., 2003). For achieving an accurate comparison, both segmentation programs are written in Java and both use the same CSSBox rendering engine. Therefore, the performance comparison is not influenced by the differences that can be caused by a different platform or rendering engine. Also the comparison of accuracy is more precise due to the same rendering engine being used.

In the comparison, we watch the following three criteria:

- Time the algorithm spent segmenting the web page.
- How accurate the results are.
- How stable across web pages the results are.

With respect to the watched criteria, the testing of every web page was performed as follows:

1. Pick a web page.
2. Let a user create reference segmentation of the web page.
3. Render box representation of the web page.
4. Run each algorithm multiple times, each time with different value of the Clustering Threshold and Permitted Degree of Coherence respectively.
5. Compute the mean run time of each algorithm and select the CT/PDoC that leads to the most accurate segmentation result.
6. Compare the run times and the accuracy of results.

The first step in the process is to pick a web page on which the segmentation is performed. To test the robustness of both compared algorithms, it was necessary to identify as wide variety of page layouts as possible and test at least one web page for every layout identified. There are several layout types of web pages we consider:

- *Complex index pages* – pages like news indexes or listings are characterized by a high degree of structure, as there are multiple topic areas covered on a single page, every area being represented by only a handful of boxes. E-commerce systems are also good representatives of this category of pages. The main difference is that the e-commerce systems usually have stronger structure in terms of similarity between individual elements.
- *Articles* – pages like these contain one main block of text, usually consisting of multiple paragraphs and image elements. Besides this one big block, there are some smaller areas that are usually related to navigation and some small pieces of generic information (like contact or news on company web sites).
- *Simple web pages* – this is a good example of some minimalistic web pages, usually educational ones. The main characteristics of web pages like these is a minimal amount of elements (and subsequently also visual areas) other than the main content.

[3] https://github.com/tpopela/vips_java.

**Table 2**
Algorithm run time comparison.

| page | VIPS time | BCS time |
|---|---|---|
| businessinsider.com (article) | 522 ms | 20 ms |
| idnes.cz (index) | 1079 ms | 39 ms |
| idnes.cz (article) | 723 ms | 53 ms |
| novinky.cz (index) | 28126 ms | 699 ms |
| novinky.cz (article) | 390 ms | 18 ms |
| reuters.com (index) | 475 ms | 15 ms |
| reuters.com (article) | 442 ms | 37 ms |
| yahoo news (article) | 342 ms | 21 ms |

We have created an evaluation dataset of real web pages containing all the mentioned page types as we describe further.

### 8.1. Evaluation dataset preparation

For creating the evaluation data set, we have identified 8 different types of pages from 5 news web sites that are listed in Table 2. Note that the *businessinsider.com* and *yahoo news* sites don't use paging and we were therefore unable to statistically process their respective index pages. We have collected a set of 100 pages for every type, which means 800 pages in total. Then, we asked three volunteers to independently create a reference segmentation for every page from the set.

To facilitate the work of the volunteers and to ensure consistent annotation of all pages of each of the eight types by a single volunteer, we have used a semi-automatic annotation approach. This approach is based on the fact that all the pages of the same type share the same template that is used for generatig their HTML code. We have created a graphical tool that allows the volunteer to interactively mark the visual clusters in one sample page of every type. Then, the tool maps the manually created segments to a DOM model of the sample page and subsequently, it automatically creates equal segments in the remaining 99 pages of the same type. Finally, the volunteer is able to browse the results of the automatic annotation graphically in order to verify its correctness. The annotation results are stored as text files containing the positions of the annotated segments for every page as well as PNG images showing the annotated segments graphically for later verification.

As a result, we have obtained 2400 annotated pages in total from our three volunteers.

### 8.2. Performance evaluation

Table 2 demonstrates the first part of the algorithm evaluation – the mean run times of both algorithms on the evaluation data set. As the Table 2 demonstrates, our algorithm is superior to the VIPS in terms of time required to process a web page. This difference gets bigger with decreasing complexity of evaluated web page.

### 8.3. Accuracy and stability evaluation

Evaluating the accuracy is a more complex task. In the area of page segmentation, there is no commonly used method for evaluating the accuracy. In statistical analysis in general, the F-score is a common way how to evaluate the accuracy. In Kreuzer, Hage, and Feelders (2013), the F-score is used, even though the underlying method for matching segments is rather crude. There is, however, an alternative that we can use. As this paper proposes, the page segmentation task, regardless of how it's performed, is basically a clustering task – each segment being a cluster of page elements. In data clustering, Adjusted Rand Index (ARI) (Hubert & Arabie, 1985) is being used to measure similarity between two clusterings. In this paper, we compare BCS and VIPS using both methods.

For ARI, each segmented web page is viewed as a clustering and every segment in that web page is a cluster of boxes rendered on that page. In case of F-score, we take a similar approach. We create pairs of *automatically detected areas* and *manually annotated areas* which share at least one rendered box. For each such pair, we calculate the precision and recall of that pair. If there are any manually selected areas that do not share boxes with any automatically detected areas, we set the recall value for each of them to 0. The resulting F-score is calculated using average values of precision and recall for the entire page. In both cases, we measure the accuracy using rendered boxes and their pertinence to visual areas in the reference segmentation. There are several rules when creating the reference segmentation:

- Each box is assigned to at most one visual area.
- There are no empty visual areas in the web pages –i.e. those that would contain no boxes.
- There are no overlapping visual areas in the page.
- Every visual area has to meet the *semantic condition*: The boxes in the area have to constitute one unit of content that is coherent visually, semantically or (preferably) both.

Evaluating one reference web page of a given type from a given site might be misleading, as there is no guarantee or even indication that segmenting other pages would generate any kind of corresponding results. That's why we performed statistical evaluation on a large set of pages.

**Table 3**
Algorithm accuracy comparison using the ARI and F-score metrics.

| page | BCS ARI | VIPS ARI | BCS F | VIPS F |
|---|---|---|---|---|
| businessinsider.com (a) | 0,5704 | 0,7010 | 0,6345 | 0,7394 |
| idnes.cz (article) | 0,6629 | 0,7240 | 0,5570 | 0,5720 |
| idnes.cz (index) | 0,5954 | 0,7926 | 0,5522 | 0,7259 |
| novinky.cz (article) | 0,7670 | 0,7877 | 0,6446 | 0,7191 |
| novinky.cz (index) | 0,5303 | 0,9121 | 0,4265 | 0,9043 |
| reuters.com (article) | 0,6123 | 0,6786 | 0,5914 | 0,6943 |
| reuters.com (index) | 0,5832 | 0,8160 | 0,5145 | 0,7569 |
| yahoo news (article) | 0,7556 | 0,5626 | 0,7102 | 0,5446 |

**Table 4**
Algorithm stability comparison: Standard deviation of the results in the dataset.

| page | BCS ARI | VIPS ARI | BCS F | VIPS F |
|---|---|---|---|---|
| businessinsider.com (a) | 0,1275 | 0,1740 | 0,0358 | 0,0721 |
| idnes.cz (article) | 0,0646 | 0,0766 | 0,0424 | 0,0794 |
| idnes.cz (index) | 0,0733 | 0,0078 | 0,0108 | 0,0070 |
| novinky.cz (article) | 0,1274 | 0,1406 | 0,0404 | 0,0731 |
| novinky.cz (index) | 0,0529 | 0,0140 | 0,0558 | 0,0219 |
| reuters.com (article) | 0,1316 | 0,1687 | 0,0301 | 0,0847 |
| reuters.com (index) | 0,0328 | 0,0516 | 0,0203 | 0,0336 |
| yahoo news (article) | 0,2089 | 0,1658 | 0,1000 | 0,0539 |

One problem remains in the evaluation system presented above and that is the hierarchy of visual area presented by VIPS. To eliminate the ambiguity that the hierarchy presents, only the leaf areas of the hierarchy will be used for the evaluation.

The Table 3 shows the comparison of accuracy of both BCS and VIPS. The F-score value is a real number between 0 and 1, where higher values are better. ARI score is between $-1$ and 1, higher values are better.

The results show that the accuracy of VIPS is slightly better, especially when processing structured pages. The reason is that BCS is too aggressive when creating clusters, thus effectively overlooking the structure. VIPS on the other hand does much better job in finding repeating patterns in the web page. When processing pages with less structure, the accuracy of BCS and VIPS is comparable, in some cases BCS is even better than VIPS.

Stability of both algorithms can be also calculated using the same data that was used to populate Table 3. We calculated stability in each data set, specifically as standard deviation of results in the set. The results are displayed in Table 4.

Again, both algorithms are comparable. In some cases the stability of BCS is almost three times better than that of VIPS, in others, it's exactly the opposite. Not looking at the degree of superiority, the stability of BCS is better in 5 data sets, i.e. 62.5% of measurements.template but across the templates as well

## 9. Discussion

Looking at the results in Section 8, the most important practical implication of the results emerges. Significantly increasing performance of vision-base page segmentation algorithms while not losing their level of accuracy opens them way to practical application, as speed is very important in modern data mining systems. Using just generic visual cues supports that attractivity, as one algorithm can be used for processing multiple document types and it is resilient to possible future changes in technologies like HTML.

The flat structure BCS produces is as important for the practical application as performance. Being able to easily consume the output of segmentation algorithm significantly lowers the barrier for using it. Note that this was even proven in Section 8 where extra measures had to be taken to make the results of VIPS comparable to the results of BCS.

To better demonstrate the advantage of the flat output of the Box Clustering Segmentation (BCS) in the evaluation process, the difference between the two output models is displayed in Fig. 4. The BCS flat model is quite straightforward – it is just a set of groups, each of which can be further processed right away. The VIPS tree model on the other hand is not that simple. Fig. 4 visualizes the different levels of the output tree with different shades of gray and the internal consistency level by numbers in the leaf areas. It may be understandable for a human observer; however, in context of an automatic processing, one needs to performs a subsequent deep analysis of the segmented result to select the right area set, as we don't necessarily want to always pick the leaf nodes of the tree. Even though the tree model offers some bright sides like the possibility to compensate for some potential defects in the output, we don't find them that significant. Therefore, we consider the flat model to be the most distinct advantage of BCS when compared to VIPS and other hierarchy-producing algorithms.

Section 8 briefly describes a brute force approach to selecting the best values of PDoC and CT. That highlights another aspect that is important in practical application – selecting the right value of the target granularity parameter. As the original

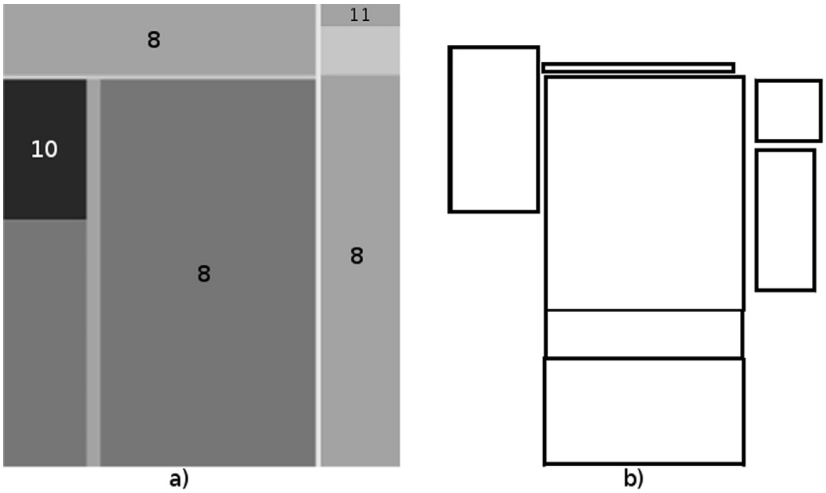



**Fig. 4.** Output model comparison: (a) VIPS tree model and (b) BCS flat model.

VIPS paper points out, different applications might need different output granularity and set the PDoC accordingly. That is our perception of CT as well. For that reason, any deeper investigation of the PDoC/CT selection process does not belong to this paper. But, since we are comparing BCS and VIPS, let's compare practical use of their respective granularity parameters. In both cases, the parameters are limited to fixed range of values and their change significantly influences the results of algorithms they are used in. We see the flexibility of CT being a real number to be a great advantage over PDoC which is integer. On the other hand, PDoC and how VIPS behaves when it changes is a great advantage over CT, as the results of changed PDoC are more predictable than the results of BCS when its CT changes.

In the field of theory, our paper opens new research area: new group of vision-based document segmentation can be explored. This area has a lot of potential, further research can improve both the accuracy and performance of vision-based clustering segmentation techniques. The overall potential of the Box Clustering Segmentation may be even greater considering it currently uses very low amount of information to perform the segmentation.

## 10. Conclusion

In this paper, we have presented a new web page segmentation method called Box Clustering Segmentation. We showed that its precision is comparable to VIPS in some cases and slightly worse in others. We have also shown that its performance is superior to the VIPS performance and we have presented two major advantages the Box Clustering Segmentation has over the existing algorithms. First of them is the strict usage of visual information only which makes our method transferrable to other document types. It also makes it resilient to changes in HTML, DOM and other technologies used on the web. The second advantage is the output structure that is more comprehensive and convenient for further processing.

The assumed applications of the proposed page segmentation method include the document cleaning, automatic adaptation of web pages for small screen devices, page preprocessing for information retrieval and document classification, logical structure discovery and information extraction tasks. By setting the appropriate values of the input parameters, the segmentation may be performed on any granularity level depending on the particular task.

## Acknowledgment

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science – LQ1602.

## References

Aguado, J. (2015). Emerging perspectives on the mobile content evolution. *Advances in multimedia and interactive technologies: IGI global*. URL https://books.google.cz/books?id=Omm2CgAAQBAJ

Akpinar, E., & Yesilada, Y. (2012). Vision based page segmentation: Extended and improved algorithm. Tech. Rep. eMINE Technical Report Deliverable 2 (D2), Middle East Technical University, Ankara, Turkey.

Akpinar, M. E., & Yesilada, Y. (2013). Vision based page segmentation algorithm: Extended and perceived success. In *Revised selected papers of the ICWE 2013 international workshops on current trends in web engineering - Vol. 8295* (pp. 238–252). New York, NY, USA: Springer-Verlag New York, inc.

Alarte, J., Insa, D., Silva, J., & Tamarit, S. (2015). Temex: The web template extractor. In *Proceedings of the 24th international conference on world wide web, WWW '15 companion* (pp. 155–158). New York, NY, USA: ACM. doi:10.1145/2740908.2742835.

Alassi, D., & Alhajj, R. (2013). Effectiveness of template detection on noise reduction and websites summarization. *Information Sciences, 219*, 41–72.

Alcic, S., & Conrad, S. (2011). Page segmentation by web content clustering. In *Proceedings of the international conference on web intelligence, mining and semantics, WIMS '11, ACM, new york, NY, USA* (pp. 24:1–24:9). doi:10.1145/1988688.1988717.

Barua, J., Patel, D., & Agrawal, A. K. (2014). Removing noise content from online news articles. In *Proceedings of the 20th international conference on management of data, COMAD '14, computer society of india, mumbai, india, india* (pp. 113–116). URL http://dl.acm.org/citation.cfm?id=2726970.2726988

Bing, L., Guo, R., Lam, W., Niu, Z.-Y., & Wang, H. (2014). Web page segmentation with structured prediction and its application in web page classification. In *Proceedings of the 37th international ACM SIGIR conference on research & development in information retrieval, SIGIR '14* (pp. 767–776). New York, NY, USA: ACM.

Bos, B., Celik, T., Hickson, I., & Lie, H. W. (2011). Cascading style sheets level 2 revision 1 (CSS 2.1) specification. *W3c recommendation*.

Bu, Z., Zhang, C., Xia, Z., & Wang, J. (2014). An far-sw based approach for webpage information extraction. *Information Systems Frontiers, 16*(5), 771–785.

Burget, R. (2007). Layout based information extraction from HTML documents. In *Proceedings of the ninth international conference on document analysis and recognition - Vol. 02, ICDAR '07, IEEE computer society, Washington, DC, USA* (pp. 624–628).

Burget, R. (2010). Visual area classification for article identification in web documents. In *Proceedings of the 2010 workshops on database and expert systems applications, DEXA '10, IEEE Computer Society, Washington, DC, USA* (pp. 171–175). doi:10.1109/DEXA.2010.49.

Cai, D., Yu, S., Wen, J. r., & Ma, W. y. (2003). VIPS: A vision-based page segmentation algorithm. *Microsoft technical report MSR-TR-2003-79*.

Coondu, S., Chattopadhyay, S., Chattopadhyay, M., & Chowdhury, S. R. (2014). Mobile-enabled content adaptation system for e-learning websites using segmentation algorithm. In *Software, knowledge, information management and applications (SKIMA), 2014 8th international conference on* (pp. 1–8). doi:10.1109/SKIMA.2014.7083570.

Cormier, M., Moffatt, K., Cohen, R., & Mann, R. (2016). Purely vision-based segmentation of web pages for assistive technology. *Computer vision and image understanding*.

Dalvi, N., Kumar, R., & Soliman, M. (2011). Automatic wrappers for large scale web extraction. *VLDB Endowment*, 230.

Eldirdiery, H. F., & Ahmed, A. H. (2015a). Article: Web document segmentation for better extraction of information: A review. *International Journal of Computer Applications, 110*(3), 24–28.

Eldirdiery, H. F., & Ahmed, A. H. (2015b). Detecting and removing noisy data on web document using text density approach. *International Journal of Computer Applications, 112*(5), 32–36.

Ferrez, R., Groc, C., & Couto, J. (2013). Mining product features from the web: A self-supervised approach. In J. Cordeiro, & K.-H. Krempels (Eds.), *Web information systems and technologies, Vol. 140 of lecture notes in business information processing, Springer Berlin Heidelberg* (pp. 296–311).

Fragkou, P. (2013). Information extraction versus text segmentation for web content mining. *International Journal of Software Engineering and Knowledge Engineering, 23*(08), 1109–1137.

Fumarola, F., Weninger, T., Barber, R., Malerba, D., & Han, J. (2011). Extracting general lists from web documents: A hybrid approach. In *Proceedings of the 24th international conference on industrial engineering and other applications of applied intelligent systems conference on modern approaches in applied intelligence - Vol. Part I, IEA/AIE'11, Springer-Verlag, Berlin, Heidelberg* (pp. 285–294).

Gao, B., & Fan, Q. (2014). Multiple template detection based on segments. In *Advances in data mining. applications and theoretical aspects, Springer* (pp. 24–38).

Hong, J. L., Siew, E.-G., & Egerton, S. (2010). Information extraction for search engines using fast heuristic techniques. *Data & Ledge Engineering, 69*(2), 169–196.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification, 2*(1), 193–218. doi:10.1007/BF01908075.

Jiang, K., & Yang, Y. (2015). Noise reduction of web pages via feature analysis. In *Information science and control engineering (ICISCE), 2015 2nd international conference on* (pp. 345–348).

Kohlschütter, C., Fankhauser, P., & Nejdl, W. (2010). Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on web search and data mining, WSDM '10, ACM, New York, NY, USA* (pp. 441–450).

Kong, J., Barkol, O., Bergman, R., Pnueli, A., Schein, S., Zhang, K., & Zhao, C. (2012). Web interface interpretation using graph grammars. *IEEE Transactions on Systems, Man, and Cybernetics. Part C (Applications and Reviews), 42*(4), 590–602.

Kreuzer, R., Hage, J., & Feelders, A. (2013). *A quantitative comparison of semantic web page segmentation algorithms*. Universiteit Utrecht, Faculty of Science Master's thesis.

Krishna, S. S., & Dattatraya, J. S. (2015). Schema inference and data extraction from templatized web pages. In *Pervasive computing (ICPC), 2015 international conference on* (pp. 1–6).

Kulkarni, A., & Patil, B. (2014). Template extraction from heterogeneous web pages with cosine similarity. *International Journal of Computer Applications, 87*(3), 5.

Kulkarni, H. H., & Kulkarni, M. K. (2015). Template extraction from heterogeneous web pages. *International Journal of Electrical, Electronics and Computer Engineering, 4*(1), 125.

Li, L., Zhou, A. M., Fang, Y., Liu, L., & Wu, Q. (2014). An improved VIPS-based algorithm of extracting web content. In *Material science, civil engineering and architecture science, mechanical engineering and manufacturing technology II, Vol. 651 of applied mechanics and materials, Trans Tech Publications* (pp. 1806–1810).

Liu, W., Meng, X., & Meng, W. (2010). ViDE: A vision-based approach for deep web data extraction. *IEEE Transactions on Knowledge and Data Engineering, 22*(3), 447–460.

Liu, X., Lin, H., & Tian, Y. (2011). Segmenting webpage with gomory-hu tree based clustering. *Journal of Software, 6*(12), 2421–2425.

Lundgren, E., Papapetrou, P., & Asker, L. (2015). Extracting news text from web pages: An application for the visually impaired. In *Proceedings of the 8th ACM international conference on PErvasive technologies related to assistive environments, PETRA '15, ACM, New York, NY, USA* (pp. 68:1–68:4).

Manabe, T., & Tajima, K. (2015). Extracting logical hierarchical structure of html documents based on headings. *Proceedings of the VLDB Endowment, 8*(12), 1606–1617.

Milička, M., & Burget, R. (2015). Information extraction from web sources based on multi-aspect content analysis. In *Semantic web evaluation challenges, semwebeval 2015 at ESWC 2015, Vol. 2015 of communications in computer and information science, Springer International Publishing* (pp. 81–92).

Safi, W., Maurel, F., Routoure, J.-M., Beust, P., & Dias, G. (2014). A hybrid segmentation of web pages for vibro-tactile access on touch-screen devices. *3rd workshop on vision and language (VL 2014) associated to 25th international conference on computational linguistics (COLING 2014), Dublin, Ireland*. 95–02

Sanoja, A., & Ganarski, S. (2014). Block-o-matic: A web page segmentation framework. In *Multimedia computing and systems (ICMCS), 2014 international conference on* (pp. 595–600). doi:10.1109/ICMCS.2014.6911249.

Sharma, A. (2004). *Understanding color management*. Graphic Design/Interactive Media Series, Thomson/Delmar Learning.

Shi, S., Liu, C., Shen, Y., Yuan, C., & Huang, Y. (2015). Autorm: An effective approach for automatic web data record mining. *Knowledge-Based Systems, 89*, 314–331.

Song, D., Sun, F., & Liao, L. (2015). A hybrid approach for content extraction with text density and visual importance of dom nodes. *Knowledge and Information Systems, 42*(1), 75–96. doi:10.1007/s10115-013-0687-x.

Uzun, E., Agun, H. V., & Yerlikaya, T. (2012). Web content extraction by using decision tree learning. In *2012 20th signal processing and communications applications conference (SIU)* (pp. 1–4). doi:10.1109/SIU.2012.6204476.

Uzun, E., Agun, H. V., & Yerlikaya, T. (2013). A hybrid approach for extracting informative content from web pages. *Information Processing & Management, 49*(4), 928–944.

Wei, T., Lu, Y., Li, X., & Liu, J. (2015). Web page segmentation based on the hough transform and vision cues. In *2015 asia-pacific signal and information processing association annual summit and conference (APSIPA), IEEE* (pp. 865–872).

Weng, D., Hong, J., & Bell, D. A. (2011). Extracting data records from query result pages based on visual features. In *Advances in databases: 28th British national conference on databases, BNCOD 28, Manchester, UK, July 12–14, 2011, revised selected papers, Springer, Berlin, Heidelberg* (pp. 140–153).

Weng, D., Hong, J., & Bell, D. A. (2014). Automatically annotating structured web data using a svm-based multiclass classifier. In *Web information systems engineering – WISE 2014: 15th international conference, thessaloniki, greece, october 12–14, 2014, proceedings, part I, Springer International Publishing, Cham* (pp. 115–124).

Win, C. S., & Thwin, M. M. S. (2014). Web page segmentation and informative content extraction for effective information retrieval. *International Journal of Computer & Communication Engineering Research, 2*(2), 35–45.
Wu, Y.-C. (2016). Language independent web news extraction system based on text detection framework. *Information Sciences, 342*, 132–149.
Xiang, Z. L., Yu, X. R., & Kang, D. K. (2015). Wrapper induction of news information for feeding to social networking service on smartphone. In *2015 17th international conference on advanced communication technology (ICACT)* (pp. 292–295). doi:10.1109/ICACT.2015.7224806.
Xu, Z., & Miller, J. (2015). Identifying semantic blocks in web pages using gestalt laws of grouping. *World Wide Web*, 1–22.
Yu, S., Cai, D., Wen, J.-R., & Ma, W.-Y. (2003). Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *Proceedings of the 12th international conference on world wide web, WWW '03, ACM, New York, NY, USA* (pp. 11–18). doi:10.1145/775152.775155.
Zeleny, J., & Burget, R. (2013). Cluster-based page segmentation – A fast and precise method for web page pre-processing. In *Proceedings of the 3rd international conference on web intelligence, mining and semantics, WIMS '13, ACM, New York, NY, USA*.
Zeng, J., Flanagan, B., Hirokawa, S., & Ito, E. (2014). A web page segmentation approach using visual semantics. *IEICE Transactions on Information and Systems, E97-D*(2), 223–230.
Zhu, W., Dai, S., Song, Y., & Lu, Z. (2015). Extracting news content with visual unit of web pages. In *Software engineering, artificial intelligence, networking and parallel/distributed computing (SNPD), 2015 16th IEEE/ACIS international conference on* (pp. 1–5).

## A.3 Accelerating the Process of Web Page Segmentation via Template Clustering

Zeleny, J.; Burget, R.: Accelerating the Process of Web Page Segmentation via Template Clustering. *Int. J. Intell. Inf. Database Syst.*. vol. 9, no. 2. March 2016: pp. 134–154. ISSN 1751-5858.

# Accelerating the process of web page segmentation via template clustering


## Jan Zeleny

Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic E-mail: izeleny@fit.vutbr.cz

## Radek Burget

Faculty of Information Technology
Brno University of Technology
IT4Innovations Centre of Excellence
Brno, Czech Republic E-mail: burgetr@fit.vutbr.cz



**Abstract:** Segmenting a web page is often one of the initial steps when performing some data mining on that page. We acknowledge that there is a lot of research in the area of segmentation based on visual perception of the web page. In this paper we propose a method how to improve the efficiency of virtually all vision-based segmentation algorithms. Our method, called Cluster-based Page Segmentation, takes the widely spread concept of web templates and utilizes it to improve the efficiency of vision-based page segmentation by clustering web pages and performing the segmentation on the cluster instead of on each page in that cluster. To prove the efficiency of our algorithm we offer experimental results gathered using three different vision-based segmentation algorithms.



**Keywords:** VIPS, vision-based page segmentation, clustering, template, template detection



**Biographical notes:** Jan Zeleny is a PhD student at the Brno University of Technology, Czech Republic. His reserach interests are in the area of Web Data Mining and Information Retrieval in general. He is also interested in the area of electronic privacy in context of these two topics.

Radek Burget received his PhD in Information Technology in 2004 from the Brno University of Technology. He is an assistant professor at the Faculty of Information Technology, Brno University of Technology. His research interests include data mining methods, semi-structured data modeling, knowledge engineering and the semantic web.


## 1 Introduction

In recent years, the World Wide Web has become perhaps the most important source of information in the world. A family of algorithms to process that information grows with it. There are several tasks that we might want to perform on the data on the Web. The largest

group of tasks falls into the area of data mining. These are tasks like information retrieval, content extraction and classification and others. Another big group of tasks targets web page restructuring for mobile devices.

All algorithms working with the data on the web require one common step – initial preprocessing in a form of web page segmentation. The segmentation step works with a simple premise – that the web page is not one coherent block of information. Instead, it contains multiple such blocks, each one containing different type of content [1]. The goal of segmentation algorithms is to identify these blocks so they can be processed separately. In case of the data mining tasks, the segmentation is often complemented by classification with the intended result being identification of blocks that are either relevant in the context of the web page or that are important for the subsequent algorithm. In case of restructuring for mobile devices, the subsequent goal is to rearrange the web page, maybe even remove some blocks that are not considered important.

There is a variety of ways how to perform the segmentation. They differ in all possible aspects, starting with the requirements on the input side and ending with granularity of the output.

Template detection (further referred to as TD) methods are a good example of that. In the literature it is considered being just remotely related but we see some common parts. For example TD identifies page segments as well, even though it just distinguishes two – useful content and the rest. The results of segmentation methods are more generic. It's a set of various blocks, each block being internally consistent in some way, most often either visually or logically.

In our work we focus on unsupervised page segmentation. Although this area has been extensively researched, there are just two usual ways how to approach the task. The first one usually utilizes DOM tree of a web page or its textual content and focuses on performance with the result being potentially inaccurate [2, 3, 4, 5]. The source of this inaccuracy is that large portions of information from the web page, such as computed CSS styles, are dropped. The second one on the other hand strongly prefers accurate results even at the expense of the result being produced more slowly [6, 7, 8, 9]. The most distinct feature of these algorithms is that they work on every web page independently. This feature is both their benefit (no need to have more pages on the input) and problem (scaling).

The motivation of our research is to remove the negative aspect of the independent processing of single web pages by eliminating the need to perform the segmentation on every inspected web page. This significantly boosts performance and scaling of vision-based page segmentation. To achieve the goal, we use clustering algorithm that is based on methods used in template detection. This combination of template detection and vision-based page segmentation addresses the greatest issue of vision-based page segmentation (time complexity) while keeping its level of accuracy. It also keeps the positive aspect of the independent page processing (only a single page is required for the segmentation itself). The method, called *Cluster-based Page Segmentation* (further referred to as CPS), is in fact a complementary algorithm. Its potential usage is not limited to any particular algorithm, it is not even bound to the area of vision-based page segmentation if the complemented algorithm in question and its implementation keep some basic rules.

In this paper we describe structures and algorithms that are comprised in CPS. We then use the theoretical and practical description and demonstrate the efficiency of our algorithm by selecting three vision-based segmentation methods and running them with and without the CPS.

## 2 Related work

Our research belongs into the extended area of web page segmentation. It is extended because we are not strictly focused on page segmentation, as explained above. While the core of our research is in the wider area because we examine possible utilization of template detection, the state of the art is more important in the area of web page segmentation itself.

To understand the area of web page segmentation, it is first important to properly classify possible motivations. Reorganizing a web page for devices with small display is a motivation that is only marginally relevant to our research, therefore it won't be analyzed further.

The most important motivation to segment web pages is preprocessing for data mining techniques. This needs to be done because web pages are semi-structured documents that contain other elements among the useful content, for example HTML tags, various descriptors and other metadata. The most simple option how to perform the preprocessing would be to strip all the metadata. However that approach is not entirely accurate because not all the metadata is easily detectable. A significant portion of what in fact belongs to metadata or is strictly speaking just a noise is presented as a data to user who looks at the page [10]. Back to the motivation, the most obvious is to clean up the metadata and the noise. That can be done by virtually every segmentation method but the template detection methods focus solely on this goal [11, 12]. Built on top of that, the next motivation is to identify semantically distinguished blocks that the page contains [8].

The goal of page segmentation is to find blocks that are internally consistent more than the page itself. The consistency can be either logical or visual, based on input parameters and used segmentation method. Both types of consistency often overlap.

The logical consistency is targeted by DOM-based and text-based segmentation methods [2, 3, 4, 5]. These analyze only textual representation of a web page, either in a form of source code of the page or a DOM tree which is just a model of this text form. The distinctive feature of these methods is that they don't need any complex transformations to perform their task, they completely depend on heuristics applied on some form of the textual representation of the analyzed page. The set of heuristics is also the only factor defining the quality of segmentation results. The array of heuristics can vary from pure text evaluation [2] to complex algorithms taking a wide variety of properties into account [5]. However because these methods don't perform any complex transformation of the textual representation of the page, they always fail to take one very important aspect into account and that is the real layout of the inspected page. Even the text-based DOM model doesn't accurately describe the real relations of individual blocks in consideration of their visual appearance [6]. When we consider CSS rules in all their complexity, the DOM tree and the corresponding tree representing the visual appearance can be very different. As an example, figure 1 demonstrates how individual parts of a web page can be moved or transformed by CSS. The example is very simple and yet the CSS changed the appearance of the page completely.

Contrary to the DOM processing algorithms, vision-based page segmentation methods are based on a simple concept with quite large computing demands. They address the issue of real layout by calculating real values of attributes defining the layout. In its complexity this calculation corresponds to real rendering of a web page, thus it will be referred to as such. The process of rendering is very complex due to complexity of both HTML and CSS specifications. That implies quite high demands both for computational power and time to process one page and that's even before the segmentation itself. Rendered page is often segmented in several iterations [6] which is also very demanding. The most commonly

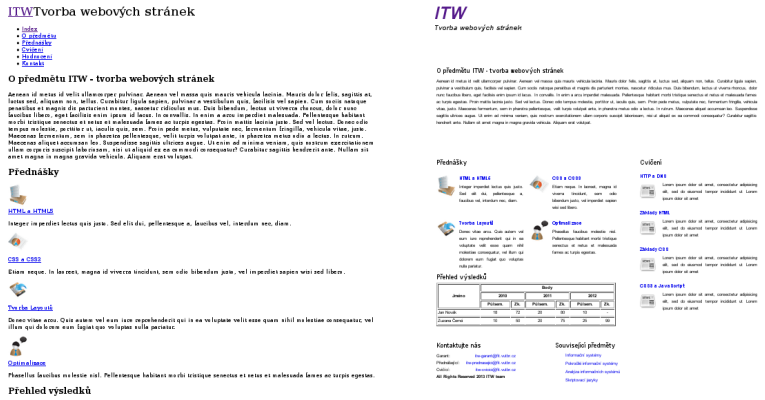



**Figure 1** Influence of CSS: a) is a slice of raw page, b) is the same page with CSS applied

used algorithm in the area of vision base segmentation is VIPS [7] and algorithms using it as a black box and improving its results [8, 9]. Another approach, partially derived from the original VIPS specification, has been offered by Burget [6]. In this paper we will demonstrate results of Cluster-based Page Segmentation working on top of several known implementations of vision-based page segmentation, including VIPS as an industry standard.

Template detection can be perceived as a special type of page segmentation. Template detection methods [11, 12, 13, 14, 15] are used to identify the noise on the processed web page so it can be removed afterwards. Algorithms in the template detection group are based on the concept of page templates [16]. This concept comes from the area of modern web design where a relatively small number of templates is used to dynamically build all the content on a web site. Each template defines core structure of a large set of pages within that site. Physically, each template is a pre-defined code which creates a frame with marked spaces in it. Based on user input, a data set is fetched from the underlying data source and a web page is created by filling these fetched data into the frame. While being a great help to web designers and content authors, templates pose a problem for information retrieval algorithms. A substantial part of every page contains information unrelated to the core topic of the page – for example navigation, advertisement and other noise. In the context of templates and template detection, all this unrelated content is considered to be a part of the template. The job of template detection methods is to correctly separate the aforementioned blank spaces from the pre-defined code.

The focus of template detection methods implies one of the disadvantages these methods have. They can be only used to distinguish between the template and the content, no finer granularity is available. Because of their design, they also usually lack the ability to perform inspection on a single web page. On the other hand they are usually designed to be as fast as possible and they scale very well.

One subset of the template detection methods makes them closely related to our research. Its a subset containing methods that are based on DOM tree comparison [11, 13, 15, 17]. Their core functionality can be used to determine a degree of similarity between two web

pages. This can be represented for example by a simple probability-based equality indicator. If the value of such indicator goes over a pre-defined threshold, the two pages are considered to be significantly similar. In the context of template detection, the significant similarity translates to both pages being based on the same template.

## 3 Cluster-based Segmentation

In our work we consider the following scenario where page segmentation is used as the key part: we have a crawling algorithm that wants to index as many documents as possible. For the initial description of the use case we can define a constraint that only a specific web site will be scanned. Every web page on that site is supposed to be segmented and the result further processed. At this point it doesn't matter what is the subsequent processing going to be, it might be anything starting from information retrieval and ending with semantic classification.

With the standard segmentation approach, every scanned web page is going to be segmented. For some large servers like world-wide news servers this means performing the segmentation task hundreds of thousands of times. Obviously this doesn't scale at all and the time required to process mid- to large-size web site is unacceptably long.

In our research we propose a new way how to deal with scaling and performance problems when processing large sets of web pages. One of the advantages is that the bigger the set of web pages is, the greater optimization this approach achieves. Another advantage is that our algorithm doesn't require all the pages to be processed at once, partial data set can be retrieved, the process interrupted and continued any time later. Our algorithm can be summarized in the following proposition.

**Proposition 1:** *When processing more pages within the same site, it is possible to increase the performance of a segmentation algorithm by performing the actual segmentation only for a limited number of pages and transform these pages to represent their respective template-based clusters. When a page can be matched to an existing cluster, an isomorphic mapping between the page and the structure representing the corresponding cluster can be used to get the results of page segmentation without performing it.*

Most often, the clustering is performed on a complete set of values. However in our case it is not convenient to store all pages and perform the clustering on the entire set of pages. One of the reasons is that it is not possible to estimate upfront how many pages will be in the set. Also there is the issue of continuous content generation. Big web sites like news servers keep publishing new content, therefore the set of web pages will never be completed. All these reasons lead to the conclusion that some form of stream clustering algorithm is required.

This aligns with another preferred feature – to store as few data as possible in order to optimize memory and disk space consumption in practical application. To achieve this goal, we don't store processed web pages at all, we just store minimal structures representing the individual cluster. These structures will be further called Cluster Representatives. Note that while we aim for the structure to be minimal, the basic requirement is to store all the data for situations where the Cluster Representative is used. These will be analyzed further in the following sections.

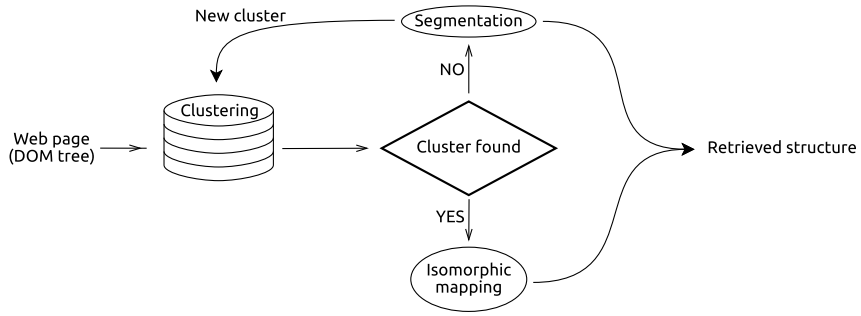



**Figure 2** Block schema of the entire approach

The high-level overview of Cluster-based Page Segmentation is outlined on figure 2. Our design starts at the *clustering* step where we try to identify which cluster the page belongs to. If such cluster is identified, an isomorphic mapping between the Cluster Representative and the page is performed and, by extension of that action, all interesting parts of the web page identified without performing the segmentation. If there is no cluster corresponding to a page on the input, we have to create a new cluster and use the input web page as an initial representative of this cluster. Before we create the representative, we need to segment the page first, as the segmentation result is an important part of each Cluster Representative. The *segmentation* process is considered to be a black box taking DOM tree as an input and returning a set or hierarchy of visual areas on the output. Any algorithm acting in conformity with these requirements can be used for the segmentation. Section 7 covers experiment with three different segmentation algorithms.

## 4 Segmentation methods

Before going more deeply into the structures and algorithms of Cluster-based Page Segmentation, we need to take a closer look at the segmentation methods this algorithm is primarily meant for. This section will be divided into two parts. The first part will cover some currently used segmentation methods that are described in the literature. The second part will be oriented towards an alternative segmentation method that we have designed. In both cases we will outline how these methods work and we will especially focus on their output format. That will be important for the subsequent design of data structures we use in the Cluster-based Page Segmentation.

### *4.1 Hierarchical segmentation methods*

There are two methods that will be described here as representatives of hierarchical segmentation methods. Their common feature is the output format. While the specifics are different, the main characteristic is the same for both – the output format is structured as a tree.

The first representative of this algorithm group is *VIPS* [7]. It segments the page in three steps. The first step is a top-down analysis of the DOM tree where the algorithm identifies visual blocks on the web page using various heuristics. For each identified block a decision is

made whether or not will the block be recursively split further – that creates a tree of blocks. In the second step the algorithm identifies separators between previously identified blocks. The approach is top-down again – at first the entire web page is considered a separator and it is recursively split to smaller ones. The last step of VIPS is content structure construction. In involves merging some boxes to achieve the right granularity.

After being processed by VIPS, the web page is represented by a set of blocks $O$, a set of separators $\Phi$ and a relation between blocks $\delta$(two blocks are in relation if they are adjacent). The most important feature of blocks is that they are not overlapping. Each block in the set is recursively segmented and then represented by another set of blocks, separators and relation. Formally, it is designated as follows [7]: $\Omega = (O, \Phi, \delta)$ where $O = \Omega_1, \Omega_2, \ldots \Omega_n$ and every $\Omega_i$ is defined in the same way as $\Omega$. This definition implies the tree structure of the returned result.

For each block the information about its position and size is absolutely essential, as well as its internal coherence. Also the alignment with its parent is used [9]. For separators it is important to store their visual impact, which can be in form of width or visibility.

Compared to VIPS, Burget takes the opposite approach in his work [6]. First of all his algorithm doesn't work on a DOM tree itself. The web page is rendered first and the algorithm then processes another structure that is called *render tree*. This tree basically represents a hierarchy of visual boxes as they are really placed on the web page.

After the render tree is created, the analysis goes bottom-up. In the next step a new tree is created – a tree of visual areas. Each visual area corresponds to exactly one box in the render tree. Only those boxes that are visually distinct from adjacent boxes are considered. After the initial tree of visual areas is created, it is modified by merging similar visual areas (for example adjacent paragraphs of a text). Finally the significant areas are identified using some slightly modified techniques used by VIPS.

The result of Burget's algorithms is different from the result of VIPS in several ways. First of all, the tree produced by Burget's algorithm contains two node types: *visual areas* and content nodes called *boxes*. All visual areas contain information about the position and dimensions of the area but the system is completely different. Burget assigns a special topographical grid to each non-leaf visual area and puts all child areas on the grid. A position of each area is represented by the grid cell in which the top-left corner of the area is. Area dimensions are represented by the number of rows and columns the area takes. This grid representation provides the possibility to disregard real distances between boxes while keeping the information about their mutual position. Every non-leaf visual area in the tree can contain only other visual areas. Each leaf visual area contains just a list of contained boxes representing either *images* or *text*. Each of these types contains different attributes describing its appearance.

### *4.2 Box Clustering Segmentation*

The Box Clustering Segmentation is included in this paper for two reasons: 1) it is designed specifically to remove the hierarchy from the result of page segmentation and 2) it is much faster than both of the methods above, therefore by including it in the result set we demonstrate that the variety of algorithms that can be improved by Cluster-based Page Segmentation is very wide.

First it is important to outline how the algorithm works and then what its results look like. Unlike the previous two, our algorithm is based on building everything from basic elements on the web page – elementary boxes. To get these boxes, the first step is to render

the web page, similarly to Burget's original approach [6]. After we have the tree of boxes, it is filtered so only the smallest visible boxes remain. These are often leaf nodes in the render tree output.

On this set of boxes we create a graph structure $G = (B, E, sim)$ where $B$ is a set of boxes on the web page, $E$ is a set of edges and $sim$ is a similarity function. To get more performance from the segmentation algorithm we optimize the graph creation by creating edges only between boxes that share at least one pixel column or row on the rendered page.

This graph structure is then used for the next step – clustering. Each cluster is started and then grown by merging boxes with other boxes or clusters. The process it iterative – in every iteration we take the most similar entities in the graph and if nothing blocks it, we merge them. The merging step includes detection of possible overlaps of the new cluster with other boxes and their subsequent inclusion in the cluster. The clustering stops when a threshold value of similarity is reached. This threshold must be set upfront and its optimal value differs for every processed page.

The mutual similarity if given by our specially designed similarity two-fold model. For simple boxes, the similarity is calculated based on similarity of their shape, size and color and by their mutual position. For clusters we then remember the mean similarity between the boxes in each cluster and use a set of special formulas to calculate the resulting cluster similarity.

There are two entities playing role in the output of Box Clustering Segmentation: boxes and clusters. Clusters represent the identified visual areas after the algorithm run is complete. For every cluster we disregard the boxes it contain. However some boxes might remain unclustered after the algorithm finishes. Those are somewhat important but they can be safely dropped in the Cluster-based Page Segmentation process because they most likely don't contain any content that might be worth storing for future retrieval.

Taking that into account, only the clusters remain useful for the Cluster-based Page Segmentation. Because we disregard the boxes they contain and there is not tree structure involved in the result of the segmentation process, they are just simple boxes represented by their position and dimensions, both measured in pixels. This demonstrates the simplicity of the result compared to the methods described in section 4.1.

## 5 Cluster set data structures

We will describe the maximally optimized cluster set as introduced in section 3. That means each cluster consists only of the Cluster Representative. There are three parts we have to consider:

- templates represented by DOM trees
- trees or sets of visual areas
- mapping between the previous two

Before going deeper, let's formally define the cluster set in general:

**Definition 1:** *Let the cluster set be defined as a set of Cluster Representatives* $S = \{C_i | 0 < i < n\}$ *where* $n$ *is the number of elements in the set and each Cluster Representative is defined as an n-tuple* $C = (D, V, M_{VD})$. $V$ *represents a result of segmentation performed*

*on the Cluster Representative, $D$ represents its DOM tree and $M_{VD}$ represents mapping between $V$ and $D$.*

Each of these parts will be explained in detail in the following parts of the paper.

### *5.1 DOM tree*

In our work we use a pruned DOM tree to represent the web page. The following definitions therefore define the pruned version of the DOM tree, as used in each Cluster Representative.

**Definition 2:** *Let a DOM tree be defined as a three-tuple $D = (V_D, v_r, P_D)$ where $V_D$ is a set of vertices, $v_r$ is a root node and $P_D$ is a set of paths. Each vertex in the set $V_D$ represents a node of the DOM tree, i.e. DOM node. A structure of the tree is encapsulated within these DOM nodes.*

Some general features of the DOM tree which should be considered when storing it follow [18, 19]. There are four basic data types in the DOM tree: string, timestamp represented by an integer number, user data blob and object. The last one represents a reference to any other DOM node. Each DOM node can have, depending on its type, $0..N$ child nodes. Similarly, a DOM node can have $0..N$ attributes. These attributes can be represented either by one of basic data types as element properties (deprecated) or by child nodes of `Attr` type.

**Definition 3:** *Let a DOM node be defined as an n-tuple $v = (p_v, C_v, A_v)$, where $p_v \in V_D$ is a parent of the node, $C_v$ represents an ordered set of its child nodes and $A_v$ is a set of its attributes.*

As it was stated above, tags in HTML follow one another in a specific order and this order usually matters for rendering. That means the same order has to be preserved in the DOM tree. The following definition specifies the relation on top of elements of $C_v$ which makes the set ordered.

**Definition 4:** *Let $C_v$ be an ordered set of child nodes of node $v$: $C_v = \{u_1, u_2, \ldots, u_n\}; \forall i : u_i \in V_D$. The element order preservation in $C_v$ can be expressed as: let $u_i, u_j \in C_v; i \neq j$, then the following condition must be obliged $i < j \leftrightarrow u_i \; precedes \; u_j \; in \; the \; HTML \; code$.*

The unordered set of attributes $A_v$ can contain virtually any HTML attribute as well as style definition. All these attributes can be later used for both more accurate matching of DOM trees and more accurate mapping of DOM nodes. In this paper only one attribute is important and that is the `id` attribute. All other attributes can be dropped from the set $A_v$. The formal definition of set $A_v$ follows.

**Definition 5:** *Let the set of node attributes of node $v$ be defined as $A_v = \{(k, v)\}$, that is a set of key-value pairs, where $k$ designates a name of the attribute and $v$ its value.*

The algorithm for matching DOM trees works with path sets. It is highly inconvenient to retrieve paths by traversing the tree every time we need to match a Cluster Representative

to new page. Thus the best option is to create the set and store it as a part of the DOM tree itself. The path set is designated $P_D$ and is defined as follows.

**Definition 6:** *Let a path in the tree be defined as an n-tuple* $p = (v_r, v_1, \ldots, v_{n-2}, v_l)$ *with the following conditions met:*
$v_1 \in C_{v_r}; v_l \in C_{v_{n-2}}; \forall 0 < i < n-2 : v_{i+1} \in C_{v_i}, C_{v_n} = \emptyset$.
*Path set is an unordered set of paths* $P_D = \{p\}$.

Note that the path set is not a multiset. The DOM tree itself can contain multiple identical paths that lead to different nodes of the tree. However the intended application of the path sets doesn't require this property to be preserved so only one instance of each element is kept in the set.

Each piece of text in the web page is always considered to be a special node in the DOM tree. Text on a web page is split into these pieces by any occurrence of an HTML element. As it will be explained in section 6, DOM nodes representing text are excluded from paths in the path set, however we still need them for mapping DOM nodes between the Cluster Representative and inspected page. Therefore we keep them in the DOM tree but just as bare DOM nodes, the content itself won't be included. The same applies for images – only the DOM node will be included, not their content. In this context it's important to note that some properties of both the image and the text might be stored in our representation, as they might be important for some steps following the Cluster-base Page Segmentation. These will be then stored in the set of attributes $A_v$.

### *5.2 Visual areas*

Section 4 demonstrates that the representation of segmentation results varies significantly and each method has some specifics. However we need just a single and generic enough representation that will unify the results.

We need to start the definitions from the building blocks – visual areas:

**Definition 7:** *Let the visual node* $v \in V_V$ *be defined as* $v = (A_v, C_v, D_v)$*, where* $A_v = \{(k, v)\}$ *is an implementation specific set of area attributes, defined as key-value pairs.* $C_v$ *is a set of child nodes and* $D_v$ *is a set of corresponding DOM nodes.*

This representation of visual area allows both the tree structure of Burget's algorithm and VIPS and also for the flat structure of Box Clustering Segmentation. In the last case, the set of child nodes will be empty for all visual areas.

The set of attributes can contain attributes like position, visual features, size and others that might be required in further processing. What is important about the attribute $C_v$ is that unlike in case of DOM tree, this time the order of children doesn't matter. The same situation applies to $D_v$, therefore both these attributes are plain sets.

The attribute set $A_v$ is a solution of the requirement for the design to be generic for any type of tree of visual areas, as it can be simply ignored in generic implementation. Attributes $C_v$ and $D_v$ reflect common properties of outputs of all the segmentation algorithms described in section 4.

Similarly to the visual area, we need to define generic enough structure that will hold all the visual areas. We call this structure *tree of visual areas*, even though the tree might have just one level of nodes in some cases.

**Definition 8:** *Let the tree of visual areas be defined as a two-tuple $V = (V_V, v_r)$ where $V_V$ is a set of visual vertices and $v_r$ is a root node of the tree. Each vertex in the set $V_V$ represents a node in the tree of visual areas, i.e. a visual area. A structure of the tree is encapsulated within these visual areas.*

When using the tree to contain the results of Box Clustering Segmentation, root node $v_r$ will be set to $nil$ value and it won't be further used. The tree of visual areas is derived from the DOM tree, the main difference between the two is just in their nodes.

### *5.3 Tree mapping*

The description in sections 5.1 and 5.2 leads to a conclusion that the cluster set can be viewed as a forest of $D$ and $V$. If references from nodes of $V$ to nodes in $D$ are not omitted, it can be also viewed as one big tree rooted at node $v_r \in V$.

References from $V$ to $D$ are important and only their basic version has been described. Because these connections are utilized in some algorithms working on top of vision-based page segmentation [8, 9], the mapping between both trees should be well defined:

**Definition 9:** *Let the relationship $A \supset B, A \in V_V, B \in V_D$ be defined as $A$ visually contains entire $B$. Let the mapping $M_{VD}$ between $V$ and $D$ be defined as a set of two-tuples: $M_{VD} = \{(v, d) | v \in V_V, d \in V_D, v \supset d, \nexists v_1 (v_1 \supset d, v_1 \in C_v)\}$.*

In order to be consistent with previous sections, we amend the definition of a visual area by the following definition:

**Definition 10:** *Let $D_v$ be a set of DOM nodes corresponding to a visual area $v$. This set is defined as $D_v = \{d | (v, d) \in M_{VD}\}$.*

## 6 CPS Algorithms

Now that all structures related to cluster set and Cluster Representatives have been described, algorithms working on top of them can be defined. Algorithm 1 displays the Cluster-based Page Segmentation algorithm, utilizing structures defined in section 5. The Algorithm 1 is more formal expression of figure 2. All the following algorithms are written in pseudo-code based on Python syntax.

The algorithm is fully automatic, no human intervention is needed. Also no learning phase is necessary, it "learns" new templates while processing the web site. Based on the algorithm outline, it is possible to identify three distinct non-trivial parts. Their description follows.

### *Creating a Cluster Representative*

This task consists of series of small transformations of DOM tree of the original page and the corresponding set or tree of visual areas with the result being the initial form of the Cluster Representative. This transformation leads to the Cluster Representative's n-tuple $C = (V, D, M_{VD})$.

Before the structure $V$ is built, the simplified representation of DOM tree $D$ has to be created by cleaning the original DOM tree of redundant nodes, according to definition

**Algorithm 1** Segmentation using the CPS algorithm

```
def segment_cps(page, cluster_set):
  dom = parse_page(page)
  representatives = cluster_set.get_all()
  for representative in representatives:
    if dom.matches(representative):
      return dom.visual_tree()
  visual = segment_page(dom)
  cluster_set.store(dom, visual)
  return visual
```

in section 5.1. This step is straightforward – a simple recursive, post-order tree traversing algorithm can be used. Creating the graph structure $D$ is trivial, it can be done as a part of the traversing algorithm.

Since $V$ is very similar to the output returned by segmenting algorithm, the only thing remaining to build $V$ is to ensure that each node has a set of DOM nodes it visually contains. This is done during the creation of $M_{VD}$. For purpose of this paper, the assumption is that this is handled by the segmenting algorithm in use, since the algorithm is the only component that has this information. With this assumption in consideration, we have $D_v$ for every node $v$ at the moment $V$ is created and we just have to extract all the information into $M_{VD}$ and then verify that $M_{VD}$ is correct by checking its conformity with definitions 9 and 10. After this step, we have a valid Cluster Representative which can be added into the cluster set.

### *Matching DOM tree to the cluster set*

After everything is stored, the trivial approach would be to segment another page. But with the Cluster-based Page Segmentation we can utilize having the cluster set and try to find a cluster which the new page belongs to. A comparison with all loaded Cluster Representatives for the site (more specifically with their $D$ elements) has to be performed first. As it was outlined in section 3, a simple iteration over the set $D$ containing all clusters and matching one by one can be performed. Because there is only a small number of clusters for each site, performing a simple iteration is sufficient to gain considerable performance boost against the plain page segmentation. The only condition that has to be met is for the DOM-to-DOM matching algorithm to be fast.

For this DOM-to-DOM matching we use modification of Common Paths Distance measuring algorithm [13] as it has been proven significantly faster than tree-edit-distance algorithms while still being precise enough to match the DOM tree to the correct template. However in our practical evaluation we needed to adjust the original algorithm for better precision. Our modifications are as follows:

- Filter out all nodes that are not representing particular HTML elements (e.g. attribute nodes, text nodes, etc.)
- If any element has an id, don't use the plain element name in the path, use it in combination with the id

These simple modifications improved results of the matching algorithm significantly and enabled higher level of result granularity. That means more clusters are created, thus less false-positives for cluster matches are encountered.

When a web page is matched to a $D$ belonging to some Cluster Representative, it is possible to use the corresponding structure $V$ containing visual areas that associated with the DOM tree. If no matching Cluster Representative is found, we consider the page to be based on a template that the algorithm hasn't encountered yet. In such case the segmentation process has to be performed on it and the result has to be used to create a new Cluster Representative.

### *Mapping nodes of both DOM trees*

Mapping nodes of the processed page to those stored in the matched Cluster Representative is the last step for the Cluster-based Page Segmentation to return useful results. We need this step so the actual content of visual areas can be retrieved, as that is the next process that is likely to be performed on the result of the Cluster-based Page Segmentation. The mapping procedure is trivial for Cluster Representatives themselves, as the mapping is already a part of its stored structure.

However in case of any other page in the cluster the process is more difficult. Again, we have a set of visual blocks of the Cluster Representative in which we are interested and the corresponding set of Cluster Representative's DOM nodes. Now we need to find the corresponding DOM node of the input page for every DOM node of the Cluster Representative. If we designate the DOM tree on input $D_I$, we are looking for a *Tree mapping between $D_I$ and $D$*. The tree-mapping problem for two DOM trees is quite complex in general, as many examples demonstrate [11, 17]. However, as we explain in our previous work [20], our scenario is very specific so we can afford some simplifications.

We are not looking for a mapping of each node, we are looking for a subtree rooted at corresponding node. Therefore the assumption is that once we find a root, all descendant nodes will correspond in both trees. For finding the root we use a *distinguished path* to a node. In its simplest version, this path is defined below. We acknowledge that this representation is rather crude but it serves our purpose well. Possible improvements are covered in our previous work [20].

**Definition 11:** *Let the distinguished path $p_N$ from root of given DOM tree to a node in that DOM tree $N$ be defined as n-tuple of two-tuples:*
$p_N = ((p_1, c_1), (p_2, c_2), \ldots, (p_k, c_k))$ *where $p_i$ is a position of a node withing its siblings and $c_i$ is a total count of siblings including that node. $i$ is an index designating a level of DOM tree from its root, i.e. how many nodes deep in the structure is the selected node and its siblings.*

This indexing approach is based on the premise stated in section 5.1 that order of DOM nodes within DOM tree has to be preserved to preserve the content layout on the web page. Therefore if a certain node in Cluster Representative was on position $3/5$ within its siblings, its corresponding DOM nodes from other pages in that cluster will again be positioned as $3/5$. This condition will be always true when traversing the part of DOM tree that corresponds to template. Once out of the template scope, thus in a particular data region, the condition might not be true but the assumption is that interesting visual areas don't have root outside of the template scope. To add at least some level of error detection

in case DOM nodes of Cluster Representative and inspected page don't correspond, there is the $c_i$ parameter which is used as a simple failsafe mechanism – if $c_i$ differs for Cluster Representative and inspected web page, two things could have caused this. Either the page has been incorrectly matched to the Cluster Representative or the node identified on level $i$ is already outside the template and within one of its data regions. In any case, the algorithm stops at that moment, as the result would be wrong anyway.

With the indexing approach described above, the algorithm using it will need to store these distinguished paths somewhere or they have to be constructed on the fly. If they are stored as part of DOM nodes or visual areas of Cluster Representatives, the algorithm for finding the DOM node within inspected page can be used directly.

**Algorithm 2** Finding a node within given DOM tree

```
def find_dom_node(distinguished_path, root_node):
  node = root_node
  for (position, count) in distinguished_path:
    if node == None or count != len(node.C):
      return None
    node = node.C[position]
  return node
```

If distinguished paths are not stored as a part of Cluster Representatives, we need to use algorithm 3 for their construction first. After construction of each path, algorithm 2 can be be used again for finding corresponding DOM node within inspected web page using the path.

**Algorithm 3** Construction of the path from root to the given node

```
def get_path(node):
  if node.p == None:
    path = ()      # empty tuple
  else:
    path = get_path(node.p)
    sibling_count = len(node.p.C)
    node_pos = 0
    for n in node.p.C:
      if n == node:
        break
      node_pos += 1
    path.prepend((node_pos, sibling_count))
  return path
```

## 7 Experimental evaluation

An experimental implementation has been designed and realized as a proof of concept. We used our Java implementation of all three algorithms in question. The Box Clustering segmentation and Burget's segmentaion algorithm are both original. Our VIPS implementation gives comparable results as the original with slightly worse performance. We use this implementation so the all three algorithms are based on the same rendering core and are thus comparable.

### *Implemented scenario*

The following scenario is considered: we have a simple crawler program, which is given a web site to process in a form of starting URL. It is processing the site page by page and extracting an interesting content from that page. If the page contains multiple areas with the interesting content, the algorithm extracts all of them.

For each algorithm we selected a different content type to be designated as interesting. The selection was based on capabilities of the algorithm – because the content classification is not our primary concern, we selected what was easiest for the algorithm to detect. For Box Clustering segmentation it was a body of an article - basically all the areas significantly bigger than the mean size of areas on the page. Within results of VIPS and Burget's algorithm we detect headlines – areas with font size significantly bigger than then average size on the page. When common web page design is considered, there is usually one heading for one article body, thus the number of selected areas should be similar for all the algorithms.

The crawling algorithm takes a very simple breadth-first-search approach. We need a global list of all links which are planned to be inspected. The crawler always takes the first URL in this list that has not yet been visited and loads the page on this URL for further processing. The second list contains visited links so we can quickly filter out already visited pages and not visit them again. Not performing this would cause deviations in our measurement. The last list contains links that lead out of the site(usually recognized by the same second-level domain) that is currently processed.

If a link leads to different site than the one that is currently inspected, instead of processing it is just stored for later usage. The crawling mechanism doesn't start parsing new site before the entire site it is currently parsing is processed. At that point, we stop our inspection but the program has capability to continue to the other site by clearing the *Extracted list*, adding the first link from *Outgoing list* into it and continue crawling on the new site.

The reason for the program to process one site at a time is simple: there is only a very limited number of templates for each site. Our measurements show us that the number of templates on a single site is quite low (see table 1 for details). Since the number of Cluster Representatives is equal to the number of templates used on the web site, having this number small makes it simple to store all the Cluster Representatives in memory while inspecting the web site they belong to. Such approach is highly convenient because we want fast access to these Cluster Representatives when processing the site. However storing them only in memory would mean that all Cluster Representatives of the site are dropped once inspection of that site finishes. That's not something we want because it would mean longer processing of the site the next time. Rather than that we store all Cluster Representatives in persistent storage. Once a site is selected for processing, all its Cluster Representatives are loaded from database and after the processing finished, the database is updated if necessary.

| site | clusters | hit ratio |
|---|---|---|
| iDnes.cz | 42 | 91.6% |
| e15.cz | 27 | 94.6% |
| telegraph.co.uk | 32 | 93.6% |
| slashdot.org | 18 | 96.4% |
| businessinsider.com | 16 | 96.8% |
| gizmodo.com | 11 | 97.8% |

**Table 1** Cluster counts for different sites

### *Implementation details*

We used in-memory database in form of a simple list of Cluster Representatives. This list is retrieved from persistent storage managed by OrientDB object-oriented database engine before processing any pages on the site. When matching DOM trees, we made one modification in the implementation for the sake of simplicity – section 6 suggests that the algorithm is working on the tree structure $D$, however the implementation works with string representation of all paths. A set of paths of a DOM tree is constructed when the DOM tree is prepared for matching and in case the DOM tree is used for a Cluster Representative, this set of paths is used along with it. The algorithm just iterates over all Cluster Representatives from the web site and compares the path set of the input DOM tree with their respective path sets one-by-one. After the DOM tree is matched, its segmented counterpart is fetched and desired content is extracted.

### *Testing and measurements*

The implementation has been tested on several Czech and world-wide web sites. Three of them are quite extensive news sites (iDnes.cz, novinky.cz, telegraph.co.uk) and two of them are highly focused CMS-based portals. Each testing set contained 500 pages recursively fetched from index page of that site. Test were performed on following hardware: Intel Core2Duo P8700 2.53GHz, 4GB RAM, HDD 5400RPM.

Table 1 illustrates how many clusters were detected per site. The column *clusters* gives the actual number of clusters on the site, while *hit ratio* illustrates how many percent of pages were matched when 500 pages were inspected. Note that matched page doesn't have to be segmented therefore the higher the hit ratio is the more time saving is achieved. One important observation has been made during the testing and that is that during the first inspection of a site, the majority of clusters is usually detected early in the process. If the number of total processed web pages has been doubled, the number of detected templates has risen for at most 33%. This demonstrates logarithmic growth of cluster set size, leading to confirmation that the number of clusters is low compared to number of the web pages on the site. Figure 3 demonstrates graphically that the number of templates on a single site converges fast to a relatively small number.

Tables 2, 3 and 4 demonstrate time difference between standard segmentation methods and Cluster-based Page Segmentation for each web site. The *plain* column tells the time necessary to segment inspected 500 pages within the site. The time includes only segmentation and retrieval of all desired data. The *CPS* column contains the time necessary to retrieve the same data with Cluster-based Page Segmentation. The time includes segmentation of pages when creating a new cluster, comparison of incoming pages against

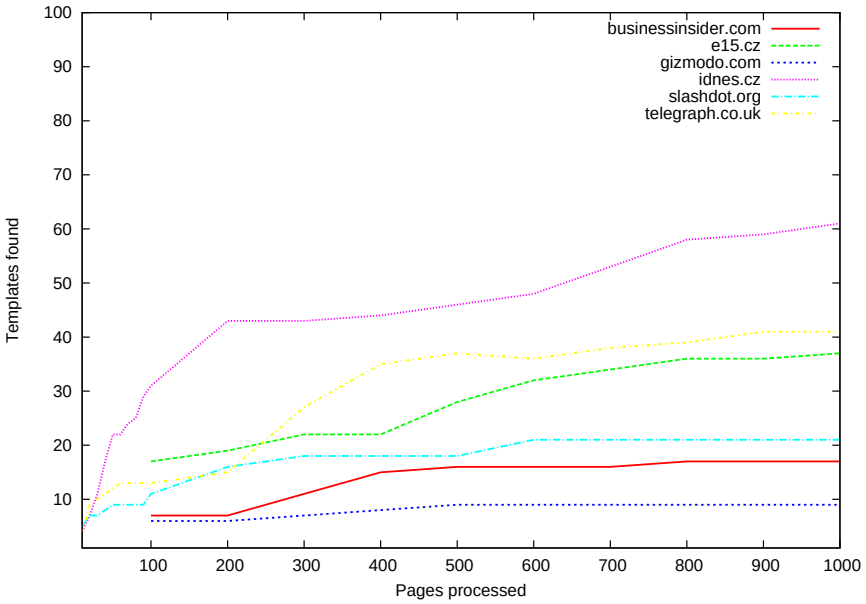



**Figure 3** Dependency of cluster count on page count

existing Cluster Representatives and retrieval of desired content from the page through node mapping. The time represents the worst case scenario for CPS, i.e. the first processing of the site. This scenario is the worst case because if the site is already processed and the corresponding Cluster Representatives are retrieved from database, the time to do this is orders of magnitude better than the time necessary to create Cluster Representatives by segmentation, even if the cluster set to create is very small. The *time saved* intuitively demonstrates how many percent of the time necessary for each segmentation algorithm to process the page set is saved when using Cluster-based Page segmentation. Both plain and CPS times are measured as a sum of times necessary to retrieve the data from all 500 processed pages.

| site | plain | CPS | time saved |
|---|---|---|---|
| iDnes.cz | 1 158 s | 145 s | 87.5% |
| e15.cz | 661 s | 55 s | 91.7% |
| telegraph.co.uk | 2 719 s | 841 s | 69.1% |
| slashdot.org | 1 925 s | 89 s | 95.4% |
| businessinsider.com | 766 s | 38 s | 95.0% |
| gizmodo.com | 560 s | 39 s | 93.0% |

**Table 2** Performance measurements of the VIPS

| site | plain | CPS | time saved |
|---|---|---|---|
| iDnes.cz | 4 019 s | 282 s | 93.0% |
| e15.cz | 420 s | 34 s | 91.9% |
| telegraph.co.uk | 1 569 s | 97 s | 93.8% |
| slashdot.org | 946 s | 30 s | 96.8% |
| businessinsider.com | 587 s | 29 s | 95.1% |
| gizmodo.com | 832 s | 38 s | 95.4% |

**Table 3** Performance measurements of the Burget's algorithm

| site | plain | CPS | time saved |
|---|---|---|---|
| iDnes.cz | 1 423 s | 195 s | 86.3% |
| e15.cz | 411 s | 42 s | 89.7% |
| telegraph.co.uk | 1 521 s | 603 s | 60.3% |
| slashdot.org | 597 s | 50 s | 91.6% |
| businessinsider.com | 530 s | 34 s | 93.6% |
| gizmodo.com | 771 s | 49 s | 93.6% |

**Table 4** Performance measurements of the BCS

Our results clearly prove that Cluster-based Page Segmentation offers high performance boost. This is confirmed by another result not visible in table 3. The time necessary for retrieving data from page belonging to existing cluster is lower by one to three orders of magnitude compared to the time necessary for retrieving the data from page not belonging to any cluster. The accuracy of the Cluster-based Page Segmentation is the same as accuracy of the used algorithm because that is the element performing the segmentation itself, no further modifications of returned results are performed. Moreover it is not purpose of this paper to evaluate accuracy, as it is depending solely on the used segmentation algorithm and selection of the algorithm is virtually not limited.

## 8 Conclusion

In this paper we presented a new way how to deal with performance shortcomings of vision-base page segmentation algorithms. Templates, one of fundamental concepts of modern web, have been used for significant performance boost of these algorithms. By combining precision of vision-based segmentation algorithms with performance superiority of template detection algorithms, it is possible to create an algorithm both precise and fast while keeping its universality.

We showed that Cluster-based Page Segmentation significantly improves performance of vision-based page segmentation when used on large sites and it therefore compensates for the greatest disadvantage of plain vision-based page segmentation algorithms.

We have also shown an alternative segmentation method called Box Clustering Segmentation and its possible usage in the Cluster-based Page Segmentation. When combined to perform the segmentation on entire web sites, these two algorithms vastly outperform any known segmentation method.

This research laid down solid base for future research. That might include improved adaptation of the Cluster-based Page Segmentation to potential post-processing algorithms and further integration with the Box Clustering Segmentation.

## Acknowledgement

This paper is a revised and expanded version of a paper entitled Cluster-based Page Segmentation - a fast and precise method for web page pre-processing presented at WIMS '13, June 12-14 2013, Madrid, Spain. This work was supported by the BUT FIT grant FIT-S-11-2 and the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070.

## References

[1] Shipeng Yu, Deng Cai, Ji-Rong Wen, and Wei-Ying Ma. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 11–18, New York, NY, USA, 2003. ACM.

[2] Eduardo Sany Laber, Críston Pereira de Souza, Iam Vita Jabour, Evelin Carvalho Freire de Amorim, Eduardo Teixeira Cardoso, Raúl Pierre Rentería, Lúcio Cunha Tinoco, and Caio Dias Valentim. A fast and simple method for extracting relevant content from news webpages. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1685–1688, New York, NY, USA, 2009. ACM.

[3] Jer Lang Hong, Eu-Gene Siew, and Simon Egerton. Information extraction for search engines using fast heuristic techniques. *Data Knowl. Eng.*, 69(2):169–196, February 2010.

[4] Bing Liu, Robert Grossman, and Yanhong Zhai. Mining data records in web pages. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 601–606, New York, NY, USA, 2003. ACM.

[5] Miroslav Spousta, Michal Marek, and Pavel Pecina. Victor: the Web-Page Cleaning Tool. In *Proceedings of the 4th Web as Corpus Workshop, LREC*, 2008.

[6] R. Burget. Layout based information extraction from HTML documents. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02*, ICDAR '07, pages 624–628, Washington, DC, USA, 2007. IEEE Computer Society.

[7] Deng Cai, Shipeng Yu, Ji rong Wen, and Wei ying Ma. VIPS: a vision-based page segmentation algorithm. Microsoft technical report MSR-TR-2003-79, November 2003.

[8] Petasis G., P. Fragkou, A. Theodorakos, V. Karkaletsis, and C. D. Spyropoulos. Segmenting HTML pages using visual and semantic information. In *Proceedings of the 4th Web as a Corpus Workshop, 6th Language Resources and Evaluation Conference.*, LREC 2008, pages 18–25, June 2008.

[9] Wei Liu, Xiaofeng Meng, and Weiyi Meng. ViDE: A vision-based approach for deep web data extraction. *IEEE Trans. on Knowl. and Data Eng.*, 22(3):447–460, March 2010.

[10] David Gibson, Kunal Punera, and Andrew Tomkins. The volume and evolution of web page templates. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, WWW '05, pages 830–839, New York, NY, USA, 2005. ACM.

[11] D. C. Reis, P. B. Golgher, A. S. Silva, and A. F. Laender. Automatic web news extraction using tree edit distance. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 502–511, New York, NY, USA, 2004. ACM.

[12] Lan Yi, Bing Liu, and Xiaoli Li. Eliminating noisy information in web pages for data mining. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 296–305, New York, NY, USA, 2003. ACM.

[13] Thomas Gottron. Bridging the gap: from multi document template detection to single document content extraction. In *Proceedings of the IASTED International Conference on Internet and Multimedia Systems and Applications*, EuroIMSA '08, pages 66–71, Anaheim, CA, USA, 2008. ACTA Press.

[14] Karane Vieira, André Luiz Costa Carvalho, Klessius Berlt, Edleno S. Moura, Altigran S. Silva, and Juliana Freire. On finding templates on web collections. *World Wide Web*, 12(2):171–211, June 2009.

[15] Karane Vieira, Altigran S. da Silva, Nick Pinto, Edleno S. de Moura, João M. B. Cavalcanti, and Juliana Freire. A fast and robust method for web page template detection and removal. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, pages 258–267, New York, NY, USA, 2006. ACM.

[16] Ziv Bar-Yossef and Sridhar Rajagopalan. Template detection via data mining and its applications. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pages 580–591, New York, NY, USA, 2002. ACM.

[17] Gabriel Valiente. An efficient bottom-up distance between trees. In *Proceedings of the 8th International Symposium of String Processing and Information Retrieval*, pages 212–219. Press, 2001.

[18] Arnaud Le Hors, Philippe Le Hegaret, Lauren Wood, Gavin Nicol, Jonathan Robie, Mike Champion, and Steve Byrne. Document object model (DOM) level 3 document object model core. W3C Recommendation, April 2004.

[19] Johnny Stenback, Philippe Le Hegaret, and Arnaud Le Hors. Document object model (DOM) level 2 document object model html. W3C Recommendation, January 2003.

[20] Jan Zeleny and Radek Burget. Isomorphic mapping of dom trees for cluster-based page segmentation. In *Proceedings of the Twelfth International Conference on Informatics INFORMATICS'2013*, INFORMATICS '13, 2013.

# Appendix B

# Entity Classification and Semantic Object Extraction

## B.1 Visual Area Classification

Burget, R.: Visual Area Classification for Article Identification in Web Documents. In *21st International Workshop on Databases and Expert Systems Applications*. IEEE Computer Society. 2010. ISBN 978-0-7695-4174-7. pp. 171–175.

2010 Workshops on Database and Expert Systems Applications

# Visual Area Classification for Article Identification in Web Documents

Radek Burget
*Faculty of Information Technology*
*Brno University of Technology*
*Bezetechova 2, 612 66 Brno, Czech Republic*
*Email: burgetr@fit.vutbr.cz*

***Abstract*—In the World Wide Web, the news and other articles are usually published in complex HTML documents containing many types of additional information that is not explicitly marked. In this paper, we propose a visual information analysis approach to the article discovery in complex HTML documents. We use a classification approach for the identification the important parts of the article within the page and we propose an algorithm for the detection of the article bounds within the page. Finally, we provide the results of an experimental evaluation.**

***Keywords*-article extraction; document cleaning; page segmentation; visual analysis**

## I. Introduction

The articles available in web news portals, online magazines, weblogs and other online sources are usually published in HTML documents with quite a complex structure. These documents often contain many other types of additional contents such as navigation or advertisement that are not explicitly annotated. This complicates accessing the article contents for both the human readers (for example using portable devices) and the automatic agents used for the content indexing, retrieval or other automatic processing [1]. The solution may be the methods of article extraction from documents (often called web page cleaning).

Most of current approaches to this problem are based on processing the HTML code of the documents. This approach brings two significant drawbacks: First, it ignores the information provided outside of the HTML code (e.g. style sheets). And second, it depends on a particular way of HTML use that may apply for a limited set of documents only.

In this paper, we propose a different approach based on the analysis of the visual aspects of the page that is presented to the reader. It is based on an assumption that there exist some common visual features that are shared among the content parts with the same meaning across the web sites. We use a classification approach for identifying important parts of the document based on their visual appearance and the page layout and we propose an algorithm for the detection of the article bounds based on this information.

## II. Related Work

Our work belongs to the area of information block discovery in documents. Moreover, it is closely related to the areas of page segmentation and document content classification.

Most page segmentation methods in the area of HTML document processing are based on DOM tree analysis [2]. However, our work is more related to the visual information based methods such as VIPS [3].

The area of information block detection (or document cleaning) is overlapping with page segmentation to some extent. From the point of view of the used document model, most of existing approaches are based on the analysis of the HTML code tree as well [2], [4], [5], [1]. Some approaches work with a visual page representation obtained by page segmentation. The work published by Yu et al. [6] is based on a visual page segmentation using the VIPS algorithm. Similarly, the work of Song et al. [7] is based on the visual block classification.

Regarding the method used for the information block discovery, most the existing approaches are based on some predefined heuristic rules used for recognizing block purpose [2], [4], [1]. Other methods are based on some classification algorithm [7], [5] applied to different features of parts of the document. This allows obtaining better flexibility of the block discovery. In [7], the RBF networks and Support Vector Machines are used whereas in [5], the Conditional Random Fields method is used.

Our method presented in this paper is based on a visual page model obtained by a page segmentation algorithm and the application of classification methods on the visual features of the detected areas. It combines and extends the results of our previously published work: In [8], we have proposed an algorithm for the article detection based on pre-defined heuristic rules and in [9], we have presented an approach to visual area classification based on selected visual features. Unlike the DOM-based methods, our method does not depend on actual structure of the document code and therefore, it should be suitable for processing sets of documents from heterogeneous sources.

## III. Method Overview

Our article identification method works with a visual representation of the web page. From the HTML code of the document and the eventual Cascading Style Sheet definitions, we create a model of the page layout that describes all the visual features of each piece of the document

1529-4188/10 $26.00 © 2010 IEEE
DOI 10.1109/DEXA.2010.49

contents. This model is used for detecting standalone visual blocks in the page. To each block, we assign its estimated purpose within the assumed article based on the values of its different visual attributes. From the positions of its estimated individual parts, we finally determine the bounds of the whole article in the page. The whole process can be summarized in following steps:

1) Page rendering – obtaining the positions, sizes and other visual features of the individual pieces of the document content from the HTML code and the related CSS specifications.
2) Page segmentation – detection of the basic visually consistent blocks on the page.
3) Visual block classification – assigning each detected block a class that corresponds to its purpose in the article such as "heading" or "paragraph". The classification is based on the values of various visual features that are computed for each block.
4) Article identification – determining the exact bounds of an area on the page that most probably contains the published article.

For rendering the web page, we use our own layout engine *CSSBox*[1]. After processing a web page, this engine provides a model of the resulting page layout. For each piece of the page content (for each HTML element), the model contains a complete information about the values of its individual visual properties. These properties include an exact position and size on the resulting page, font properties, color properties (text color, background), border information (CSS allows to create a border around any HTML element) and the text content itself. These properties are then further processed in the next step described below.

## IV. Detection of the Basic Visually Consistent Blocks using Page Segmentation

The information obtained from the rendering engine describes the resulting page layout at the finest possible granularity level – it works with the smallest identifiable pieces of the content. These pieces usually correspond to displayed atomic text strings or other atomic content (such as images) and we will call them the *content elements*. Each content element can be assigned a set of its visual properties. The next step in our method is discovering the basic *visually consistent blocks* formed by sets of content elements that potentially correspond to some part of the article with a particular purpose.

For the recognition of such blocks, we use our previously published page segmentation algorithm [9]. Basically, the algorithm consists of one top-down and two bottom-up segmentation steps:

1) Basic visual area detection (top-down step) – at start, we consider the whole page to be a single *visual area*. We detect the separators in this area that may be formed by rivers of whitespace, border lines contained in the contents or special separating objects. We split the page to sub-areas according to the detected separators. Then, we repeat the same operation recursively on the detected areas until no more visual separators are detected. As the result of this step, we obtain a *hierarchy of visual areas* in the document. The smallest visual areas in this hierarchy (the leaf areas) are the individual content elements. In the following two bottom-up steps, we detect specific groups of the visual areas for each parent area in the hierarchy.
2) Line detection – we detect the leaf areas that are vertically aligned to form a single line of text and we join them to a single area. That means, we make a text line the smallest identifiable element.
3) Visually consistent block detection – for each parent area, we create groups of its adjoining child areas that have consistent text style. We consider two content elements to have a consistent text style if the difference in their font size, weight and style is under a certain pre-defined threshold. During this process, we also consider the previously detected area separators; we do not group the explicitly separated visual areas together. We will call the groups detected in this step the *visually consistent blocks*.

The result of the page segmentation phase is a hierarchy of visual areas detected in the document and a set of visually consistent blocks detected in these areas.

[1] http://cssbox.sourceforge.net/

Table I
THE CLASSES ASSIGNED TO THE INDIVIDUAL PARTS OF THE ARTICLES

| | |
|---|---|
| `h1` | main article heading |
| `h2` | second-level heading in the article |
| `subtitle` | the article subtitle |
| `perex` | the leading paragraph of the article |
| `paragraph` | an ordinary paragraph |
| `date` | publication date |
| `author` | author name |
| `authordate` | author and the date in a single block |
| `none` | remaining areas that do not belong to the article |

## V. Visual Block Classification

Some of the detected visually consistent blocks potentially form specific parts of the published article. Based on various visual features, we assign a class to each consistent block. For this purpose, we have identified a set of block classes summarized in Table I that are commonly present in the articles published on the web. The last class `none` corresponds to any other area in the document that does not form part of the published article.

For the classification of documents, we use all the visual features available in the segmented documents. Since the

Table II
FONT FEATURES

| | |
|---|---|
| *fontsize* | Average font size in percent where the average font size of the whole document is considered to be 100 %. |
| *weight* | Average font weight from the range 0..1. The regular font characters are assigned the weight of 0, bold characters have the weight of 1. |
| *style* | Average font style from the range 0..1 (normal or italic style) computed analogically to the average weight. |

Table III
SPATIAL FEATURES

| | |
|---|---|
| *aabove*, *abelow*, *aleft*, *aright* | Numbers of areas that are placed above, below, on the left and on the right of the area within its parent area. |
| *relx*, *rely* | Relative position of the area within the whole page. 0 means the left edge or top of the page respectively, 1 means the right edge or bottom. |
| *depth* | Height of the visual area hierarchy subtree with the root in this area. |

Table IV
TEXT FEATURES

| | |
|---|---|
| *nlines* | Number of text lines in the area. |
| *ncols* | Number of columns in the area, i.e. number of child areas placed at different horizontal positions. |
| *tlength* | Total length of the contained text. |
| *pdigits*, *plower*, *pupper*, *pspaces*, *ppunct* | Percentages of digits, lowercase letters, uppercase letters, whitespaces and punctuation characters in the contained text. |

documents come from different sources, it is important that the chosen visual features be comparable for the whole set. Therefore, we have avoided using absolute values such as absolute font size and we prefer relative values of all features. Following visual features are computed for all the visual areas of all the processed documents:

- **Font features** (Table II) are computed as average values for the whole visual area weighted by the number of characters with a particular value of that feature.
- **Spatial features** (Table III) describe the position of the given visual area in the page and the relations to other area.
- **Text features** (Table IV) describe the text content properties. The text content of an area is a text string obtained as a concatenation of all the text boxes covered with the area.
- **Color features** (Table V) include the color properties of the area background and the average color properties of the text. In order to obtain some comparable values, we have used the luminosity and contrast rates according to the WCAG recommendation ([10]).

The values of the individual features are computed for each area and they are used as input for the area classification. First, the classifier must be trained using a set of documents where the classes have been assigned manually to the individual document parts. Then, the classifier is used to assign classes to all the visual ares in new, previously unseen documents. The particular data sets used for our experiments are described in section VII.

Table V
COLOR FEATURES

| | |
|---|---|
| *tlum* | Average text luminosity. |
| *bglum* | Background luminosity. If the given area has a transparent background, we consider the background color of its parent area. |
| *contrast* | Average color contrast computed from the text and background luminosities. |
| *cperc* | Percentage of text of the same color in the document. This value tells how much this text color is unique or prevalent within the given document. |

As the result, we obtain a document with the contained areas assigned either to one of the article part classes from table I or to the class `none` that indicates no part of the article. The last step is then the article identification itself.

## VI. ARTICLE IDENTIFICATION

For the article identification, we use a segmented document where some areas have been recognized as parts of the article. However, the article identification algorithm must assume somehow limited overall precision of the classification that – according to our experiments – varies from 60% to 98% depending on the properties of the training and testing document sets. In other words, not all the visual areas in the document are correctly classified.

In our approach, we combine the classification result with a few heuristic rules that are used for obtaining the most probable bounds of the article. The rules are based on following observations:

- The article forms a rectangular area in the page.
- It usually starts with a heading that is easily identifiable. As we have shown in our previous work, the headings can be easily recognized in the page using even simple heuristic rules [8] as well as using the classification methods [9].
- It consists mainly of a *consistent text flow* with occasional inserted boxes that may be further structured (for example images or information boxes).
- The article contents is left-aligned (or right-aligned for languages written in the right-to-left direction) or block-aligned.

The consistent flow of text in our visual area model corresponds to a sequence of areas corresponding to text paragraphs placed below each other. These areas have only one column, i.e. the $ncols$ visual property has the value of 1.

Moreover these areas are not further structured, i.e. the value of $depth$ property of the area is low (typically $depth < 3$ if we allow some nested visual areas formed for example by a box drawn around the block).

According to these observations, our algorithm uses headings as the starting points of the article discovery. For each visual area that has been assigned the `h1` class during the classification, the algorithm identifies the rectangular area in the page (that we will call the article *bounds*) as follows:

1) We identify the *parent area* of the given heading area. The whole process is performed within this parent area.
2) At start, the article bounds are equal to the heading bounds within the parent area.
3) We take all the visual areas placed directly below the current article bounds and we compute the probability that these areas belong to the article as explained below.
4) If the probability is sufficient, we expand the article bounds down to cover the new visual areas. This may also include expanding the bounds to the left or to the right if necessary. We repeat with the step 3.
5) If there is no such area or the probability is not sufficient, we have encountered the end of the article and the process ends.

For estimating the probability that a visual area belongs to the article, we assign the examined areas one of following *probability levels*:

- **Level 3** (classified) – the area or its part has been assigned a class different from `none`.
- **Level 2** (probable) – the area corresponds to the simple text flow. There is a single visual area below the current bounds and it has the visual features $ncols = 1$ and $depth < 3$. Even if this area has not been classified as a part of the article, it most probably corresponds to a paragraph of text.
- **Level 1** (possible) – there are multiple areas placed below the current bounds or the area has more complex structure ($ncols > 1$ or $depth \geq 3$). These areas may present a more complex structures within the article (for example an info box) or they might indicate that the article has finished.
- **Level 0** (not possible) – the area requires expanding the article bounds to the left – it breaks the article alignment on the left side.

The visual areas of the levels 0 and 1 possibly indicate the end of the article. For determining the bottom bound of the article more precisely, we use an error counter $errcnt$. In the beginning, we set $errcnt = 0$. If the level 0 or 1 areas are encountered, the $errcnt$ value is increased by some value depending on the area level. If the level 2 area is encountered, $errcnt$ is reset to 0. If the $errcnt$ value exceeds certain threshold, we have encountered the end of the article.

Table VI
THE PERCENTAGES OF ERRONEOUS IDENTIFICATION OF THE INDIVIDUAL ARTICLE BOUNDS FOR THE ARTICLES FROM A SINGLE SOURCE (A) AND FROM MIXED SOURCES (B)

| | **(A)** | **(B)** |
|---|---|---|
| Left bound | 0 % | 6 % |
| Right bound | 4 % | 17 % |
| Top bound | 0 % | 1 % |
| Bottom bound | 19 % | 27 % |
| (incomplete article) | 2 % | 13 % |
| (additional content) | 17 % | 19 % |
| Overall failure rate | 21 % | 29 % |
| without add. content cases | 6 % | 19 % |

In that case, we revert the article bounds to the last value where the visual area level was at least 2 ("probable"). In other words, we allow the "possible" and "not possible" level areas to form part of the article only if there is some higher level area placed in a sufficient distance below them.

At the end this process, we obtain probable rectangular bounds of the article in the page. If there are multiple `h1` headings detected in the page, we obtain the bounds for multiple articles. The article contents correspond to the contents of all the visual areas contained in the obtained bounds. Moreover, the classes assigned to the visual areas inside of the discovered article bounds indicate further article structure.

## VII. EXPERIMENTAL RESULTS

For running the experiments, we have implemented a page segmentation tool that performs the page segmentation and it computes the values of all the visual features for each visual area. For the classification, we have tested several classifiers from the WEKA[2] package that we have integrated to our tool. We have obtained the best overall precision and recall values using the Support Vector Machines algorithm. Therefore, in this section, we present the results obtained using this method.

Our segmentation tool also allows to manually assign classes to the detected visual areas using a graphical user interface. This feature has been used for creating a training set of documents with manually assigned classes.

We have created a list of European and American web news portals containing 20 portals in total. Each of these portals provides an RSS feed that contains the URLs of the latest published articles. We have automatically segmented all the articles in the RSS feeds. From each website, from 5 to 15 documents have been obtained according to the different sizes of the RSS feeds. In total, we have obtained 559 documents and during the segmentation, 198 129 visual areas have been detected in these documents.

[2]http://www.cs.waikato.ac.nz/ml/weka/

We have created a training set of documents by manually annotating the classes for 3 documents from each website. This set was used for training the classifiers. Then, we have applied our article identification algorithm to all remaining pages from all the websites. We have tested the article extraction for two use cases:

1) Extraction from a single source – the training set and testing set of documents contain the documents from the same news portal; however, the sets are disjoint. We have repeated the tests for all the source portals separately.
2) Extraction from multiple sources – the training and testing sets contain mixed documents from all the sources (the sets are disjoint again).

The combination of the training and testing set influences the percentage of correctly identified visual areas in the documents and thus, it also influences the precision of the article identification. The $errcnt$ threshold for the bottom bound detection has been set to 10 in our experiments.

During the evaluation, we have observed the percentages of correct bounds identification separately for the top, bottom, left and right bound of the article. The results for both use cases are shown in Table VI. The top and left bounds are usually correctly determined. The left bound is usually detected from the left visual area alignment; the top bound is always determined from the heading position. The failure in the top bound indicates that the heading has not been found correctly and it implies the failure in the remaining bounds as well. However, this has only occurred for 1 % of the documents.

The discovery of the bottom bound is the least reliable. Therefore, we have included separate failure rates for the cases when an incomplete article has been detected (the bottom bound was placed above the correct position) and the opposite case, where an additional content has been included in the article (the bottom bound was placed below the correct position). However, in most cases, the additional content detected as a part of the article below the text itself includes links to related articles, discussion related to the topic. The bottom bound of the article is then not clear even for a human reader and for many applications, this is an acceptable result of the article extraction. Therefore, in addition to the overall failure rate, we include the failure rate without the cases where some acceptable additional content has been added below the article text in table VI.

For this case, the results show 94 % and 81 % of correctly identified articles depending on the training and testing set combination. Better results might be reached by further experiments with the $errcnt$ computing and threshold. These results are comparable or slightly better than the results of a HTML-based method published in [5]. However, since the task specification and the document sets were different in both cases, the comparison is only informative.

## VIII. Conclusion

In this paper, we have proposed a method for article discovery in web pages based on the analysis of the visual appearance of the individual elements and of the page layout. Unlike the HTML-based methods, our approach does not require any specific way of the HTML usage to be shared among the processed documents or web sites. The experiments indicate that a sufficient precision may be reached.

## Acknowledgment

This work was partially supported by the BUT FIT grant FIT-S-10-2 and the research plan MSM0021630528.

## References

[1] L. Yi, B. Liu, and X. Li, "Eliminating noisy information in web pages for data mining," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2003.

[2] M. Kovacevic, M. Diligenti, M. Gori, and V. Milutinovic, "Recognition of common areas in a web page using visual information: a possible application in a page classification," in *ICDM '02*. Washington, DC, USA: IEEE Computer Society, 2002, p. 250.

[3] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma, *VIPS: a Vision-based Page Segmentation Algorithm*. Microsoft Research, 2003.

[4] S.-H. Lin and J.-M. Ho, "Discovering informative content blocks from web documents," in *KDD '02: Proceedings of the eighth ACM SIGKDD international conference*. New York, NY, USA: ACM, 2002, pp. 588–593.

[5] M. Spousta, M. Marek, and P. Pecina, "Victor: the web-page cleaning tool," in *Proceedings of the 4th Web as Corpus Workshop (WAC4) at the sixth International conference on Language Resources and Evaluation (LREC 2008)*, Marrakech, Morocco, 2008.

[6] S. Yu, D. Cai, J.-R. Wen, and W.-Y. Ma, *Improving Pseudo-Relevance Feedback in Web Information Retrieval Using Web Page Segmentation*. Microsoft Research, 2002.

[7] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma, "Learning block importance models for web pages," in *WWW '04: Proceedings of the 13th international conference on World Wide Web*. New York, NY, USA: ACM, 2004, pp. 203–211.

[8] R. Burget, "Automatic web document restructuring based on visual information analysis," in *Advances in Intelligent Web Mastering - 2, Proceedings of the 6th Atlantic Web Intelligence Conference - AWIC'2009*, ser. Advances in Intelligent and Soft Computing , Vol. 67. Springer Verlag, 2010, pp. 61–70.

[9] R. Burget and I. Rudolfová, "Web page element classification based on visual features," in *1st Asian Conference on Intelligent Information and Database Systems ACIIDS 2009*. IEEE Computer Society, 2009, pp. 67–72.

[10] B. Caldwell, M. Cooper, L. G. Reid, and G. Vanderheiden, *Web Content Accessibility Guidelines 2.0*. The World Wide Web Consortium, 2008.

## B.2 Automatic Annotation of Online Articles

Burget, R.; Burgetová, I.: Automatic Annotation of Online Articles Based on Visual Feature Classification. *International Journal of Intelligent Information and Database Systems.* vol. 5, no. 4. 2011: pp. 338–360. ISSN 1751-5858.

# Automatic annotation of online articles based on visual feature classification

Radek Burget* and Ivana Burgetová

Faculty of Information Technology,
Brno University of Technology,
Bozetechova 2, 612 66 Brno, Czech Republic
E-mail: burgetr@fit.vutbr.cz
E-mail: burgetova@fit.vutbr.cz
*Corresponding author

**Abstract:** When applying the traditional data mining methods to World Wide Web documents, the typical problem is that a normal web page contains a variety of information of different kinds in addition to its main content. This additional information such as navigation, advertisement or copyright notices negatively influences the results of the data mining methods as for example the content classification. In this paper, we present a method of interesting area detection in a web page. This method is inspired by an assumed human reader approach to this task. First, basic visual blocks are detected in the page and subsequently, the purpose of these blocks is guessed based on their visual appearance. We describe a page segmentation method used for the visual block detection, we propose a way of the block classification based on the visual features and finally, we provide an experimental evaluation of the method on real-world data.

**Keywords:** automatic annotation; online articles; page segmentation; document preprocessing; visual features; visual analysis; data mining; classification.

**Reference** to this paper should be made as follows: Burget, R. and Burgetová, I. (2011) 'Automatic annotation of online articles based on visual feature classification', *Int. J. Intelligent Information and Database Systems*, Vol. 5, No. 4, pp.338–360.

**Biographical notes:** Radek Burget received his PhD in Information Technology in 2004 from the Brno University of Technology. He is an Assistant Professor at the Faculty of Information Technology, Brno University of Technology. His research interests include data mining methods, semi-structured data modelling, knowledge engineering and the semantic web.

Ivana Burgetová received her PhD in Computer Science and Engineering in 2009 from the Brno University of Technology. Currently, she is an Assistant Professor at the Faculty of Information Technology, Brno University of Technology. Her research focuses on bioinformatics, data mining and theoretical computer science.

## 1 Introduction

In the last years, almost all the information published in the classical printed media is also available electronically via the World Wide Web. Newspapers, magazines and press agencies maintain their online versions and a large number of electronic magazines, weblogs and other types of online media make the web a vast source of electronically published articles.

It is the specific feature of the web that the articles themselves are usually buried in other content that forms the web page such as the web navigation, related articles, discussion or advertisement. As shown in Yi et al. (2003), this feature of web pages significantly complicates further application of data mining and information retrieval methods on the published articles. When applying the text classification methods on the whole web documents, the additional information in the page can be viewed as a noise that reduces the precision of the classification. Therefore, it is desirable to remove the additional content before further computer processing of the article. This requires the application of a detection algorithm that recognises the main article in the web page and eliminates the remaining information.

Similar situation occurs when automatically processing the article contents. The articles usually consist of several content blocks with a different meaning that include headings, author, publication date, lead paragraph and text paragraphs. The identification of these parts is also important for further article processing. For example, different weights may be assigned to different content blocks when a search index is being built.

Currently used web publication technology provides very limited options for an explicit annotation of the purpose of the individual content blocks and often, even these limited options are not used by the publisher. Therefore, the automatic identification of the article in the web page and the recognition of the purpose of its individual parts require a thorough analysis of the document. One of the possible ways is analysing the visual information that is provided for human readers.

The human readers usually use the visual features of the content for finding the main article in the page and for identifying the purpose of its individual parts. The designers of the web pages help the users by maintaining some style rules commonly accepted for this kind of contents. An article usually forms a visually separated block in the page with a different visual style than the surrounding information. Its main heading can be easily identified in the whole page thanks to its visual features, mainly the font size and style. The internal structure of the article is expressed visually as well. The leading paragraph, publication date, eventual subheadings and other blocks are clearly visually separated from the main text of the article.

Although the visual presentation of the document content is apparently important for human readers, most of the existing approaches to the automatic content block recognition are based on the analysis of the document code usually represented as a DOM tree. On the other hand, very few authors have examined the role of the actual visual features of the content. In this paper, we propose an approach to automatic annotation of important web page elements based on the above observations regarding the visual content presentation. We propose a model that describes the human perception of the page as exactly as possible. In order to use all the visual information that is available to the human reader, we work with a model of the rendered page instead of a more common approach based on modelling the document code. We propose detecting the basic blocks

in the page by an adopted page segmentation algorithm and subsequently, we propose a way of classification of the detected blocks based on their mutual positions and other visual features. As the result, we can assign a class to each detected block that helps to decide if the block forms a part of an article and eventually its meaning in the article. This information is potentially useful for further automatic processing of the documents.

We present the results of our experimental application of the proposed approach on real web documents. According to our experiments, the results can be used for detecting the important areas in web documents and the method can be used as an alternative to the existing DOM-based approaches or eventually, both approaches may be combined.

## 2 Related work

From a general point of view, our paper belongs to a broader area of information extraction from web documents. The research in this area has been running from the late '90s and during this time, many different approaches have been developed. Most of the methods are based on specialised procedures called *wrappers* that identify a particular information in a document based on recognising HTML code patterns that typically surround the desired information in a defined set of documents (Kushmerick, 1997). Since a manual construction of the wrappers is a time-consuming work, many methods have been proposed for automatic wrapper construction based on a set of annotated sample documents. The most important approaches include grammatical or automata inference (Freitag, 1997; Kosala et al., 2002) and relational machine learning (Cohen et al., 2002). The main problem of the wrapper is their sensitivity to changes in the documents (often called *brittleness*) which may cause the wrapper to stop working properly at any time (Kushmerick, 2000). Therefore, some approaches introduce more general models of the processed documents. This allows separating the extraction algorithm from the details of the document code. For example, Burget (2004) proposes a conceptual modelling approach. Some of the existing tools provide a human-assisted visual wrapper definition (Baumgartner et al., 2001; Liu et al., 2001) where the resulting wrappers are described using an abstract language in order to obtain greater flexibility.

The area of web page cleaning and information block detection in web pages can be viewed as a specific application of the general information extraction methods. Most of the existing approaches are based on the analysis of the HTML code of the document which is usually represented by a DOM (e.g., Chakrabarti et al., 2008; Gupta et al., 2003; Kovacevic et al., 2002; Lin and Ho, 2002; Mukherjee et al., 2003; Yi et al., 2003). The advantage of the DOM approach is generally its simplicity and scalability. On the other hand, the DOM itself gives an approximation of the resulting page only. For example, it does not contain any information about the positions of the individual content blocks and their visual style. In order to obtain the complete picture, the actual visual features of the content must be computed according to the attached style sheets, used fonts and other information which actually results in document rendering.

The approaches based on the rendered document processing usually work with a visual page representation obtained by page segmentation. The work published by Yu et al. (2002) is based on a visual page segmentation using the VIPS algorithm Cai et al. (2003). The closest to our approach is the work of Song et al. (2004) that is also based on the visual block classification. It assigns one of the four importance levels to each block based on its visual features. In contrast to this approach, we do not evaluate an overall

importance of the blocks. Instead, we assign more specific classes to the visual blocks in order to distinguish their purpose. Moreover, we have chosen a different set of visual features used for classification that we found more suitable as discussed in Section 5.

Our approach is based on a modification of our previously published page segmentation algorithm Burget (2007). Similarly to Song et al. (2004), we work with visual blocks detected in the rendered documents. However, our approach models the page division with finer granularity and it looks for areas consistent in their visual style. As the next step, we use the classification approach based on the visual features only for recognising the particular areas in the page. In contrast to the algorithms based on the HTML code analysis, there are no additional requirements on the underlying HTML code such as the usage of particular HTML elements. Moreover, the method is independent on the document language since only the visual features are used.

This paper is an extended version of a previously published paper Burget and Rudolfová (2009). In contrast to the original paper, we have introduced a new set of visual features used for the classification that allowed increasing the achieved precision significantly. We have greatly extended the experimental testing part by using larger sets of data and more classification methods. And lastly, we provide a more detailed description of the proposed approach and its evaluation.

## 3 Overview of the approach

Our approach is inspired by the way the human readers read the web pages. When looking for the article in a web page, the readers accomplish the following tasks:

1 identification of the basic visual blocks in the web page

2 classification of these blocks based on various features in order to determine their meaning

3 choosing the appropriate block that is worth reading.

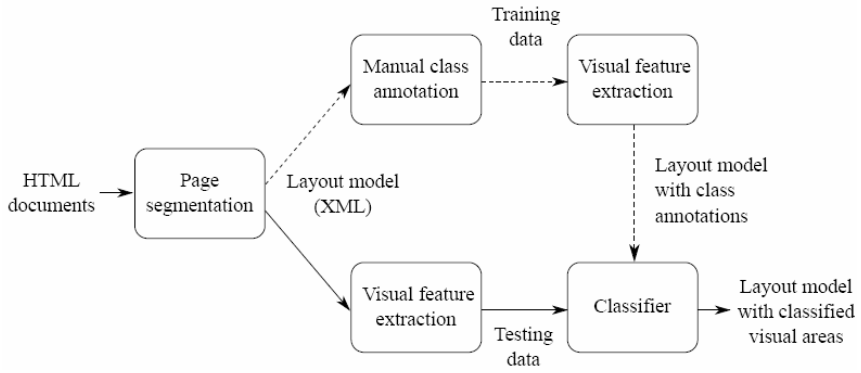
In this paper, we focus on the first two tasks. The detection of the basic visual blocks in the page is accomplished by employing a page segmentation algorithm. For accomplishing the second task, various visual attributes of the detected blocks are extracted and a classification algorithm is used for assigning classes to the individual blocks.

The whole visual area classification process is illustrated in Figure 1. It consists of two phases – a training phase and the classification phase itself.

In both phases, the first step is page segmentation. In this step, the page is rendered and basic visual blocks are detected. The details are described in Section 4. As the result, we obtain a model of the page layout stored in an XML file. The whole page is represented as a tree of visual blocks that describe the page layout from the root block (the whole page) to the smallest detected blocks. For each block, the model contains its position and other visual properties.

For the training phase, we segment a sample set of web pages and we manually annotate the visual blocks that form part of a contained article such as headings, text paragraphs, etc. Then, we extract the visual properties of all the visual blocks in the sample set of web pages and use them as the training examples for the classifier together with the manual annotations.

**Figure 1** A schema of the visual area classification process




Note: Dashed lines show the training phase, solid lines show the classification phase.

In the classification phase, we process new HTML documents that have not been manually annotated. Each document is segmented using the same algorithm and the same visual features of the detected blocks are extracted directly from the obtained layout model. Obtained data forms an input of the previously trained classifier that assigns classes to the individual blocks.

The details of both the training set preparation and the used classification algorithms are provided in Section 5.

## 4 Page segmentation approach

The purpose of the page segmentation is to detect the visual blocks that can be assigned a meaning in context of the article (for example a heading or a paragraph). Therefore, we are looking for separated groups of text lines in the page where the whole group has a consistent visual style.

The segmentation method we have used is based on our previously published method (Burget, 2007) with significant modifications. We have redefined the purpose of the box clustering phases so that the algorithm consists of a *line detection phase* and a subsequent *block detection phase* that includes the visual style comparison. As the result, we obtain a larger number of smaller areas with consistent style from the segmentation process. This result is more suitable for the following step of area classification described in Section 5.

The input of the page segmentation algorithm is an HTML document and the related Cascading style sheets required for rendering the document. The output is a tree of detected areas that models the visual properties of the areas and their eventual nesting. The whole segmentation process consists of the following steps:

1 page rendering – obtaining the information about the positions of the basic document elements on the page and about their visual style.

2 detecting basic visual areas – detecting the elements that form visual areas in the page and creating a tree of areas based on their nesting

3 text line detection – joining the areas on the same line

4 block detection – detecting the larger areas with the same visual style of the blocks

In the following sections, we briefly describe the individual phases.

### *4.1 Page rendering*

For rendering the web page, we have used an experimental *CSSBox* layout engine (http://cssbox.sourceforge.net/) that has been designed mainly for this purpose. The task of the rendering engine is to load the HTML code of a document and all the corresponding CSS style sheets and subsequently, to determine the positions and visual features of all the elements on the resulting page. The way of determining these features is given mainly by the CSS specification (Bos et al., 1998).

The rendering is performed on a virtual canvas with the initial size of 1,000 × 800 pixels which may be adjusted according to the resulting style (computed widths and heights) of the rendered elements. The result of the rendering is (according to the CSSBox terminology) a set of *boxes*. By a box, we understand a rectangular area in the resulting page with a given position and size on the canvas containing an arbitrary part of the document content. Generally, there is a box created for each DOM element (so called *element boxes*), for each connected text string in the document (*text boxes*) and for each inserted object such as an image (so called *replaced boxes*). For each box, the following visual features are computed by the rendering engine:

- size and position on the page – the *x* and *y* coordinates and the *width* and *height*; we will call this the *box bounds*
- background colour (some boxes may have transparent background)
- font properties (font size, weight and style)
- border properties – in CSS, there may be a border (a frame) of an arbitrary width defined around a box
- if the box is formed by a text, we obtain the text string.

For further analysis, each box may be represented as a tuple:

$$b = (type, x, y, width, height, bgcolor, fontsize, weight, style, border, text) \quad (1)$$

where *type* is either the element box, text box or replaced box as defined above and the other members correspond to the above mentioned visual features. All the positions and sizes are measured in pixels. The background colour (*bgcolor*) is represented as a RGB value or a special *transparent* value, if there is no background colour defined for the given box. The *weight* and *style* properties have the value of 0 for normal font and 1 for bold or italic font respectively.

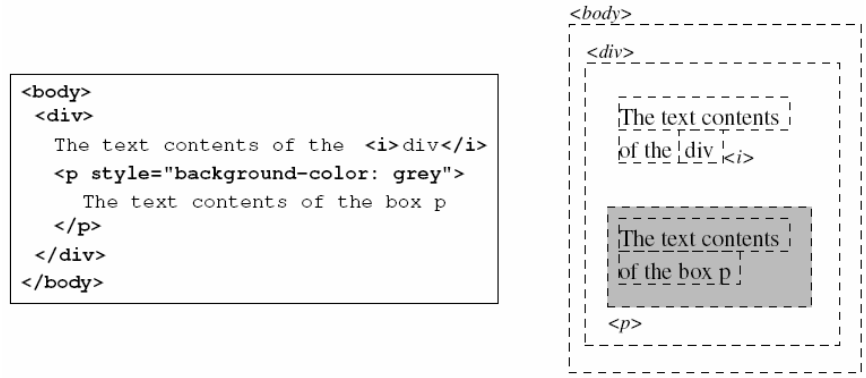
The whole rendered document may be then represented as a set of boxes

$$B = \{b_1, b_2, b_3, ..., b_n\} \quad (2)$$

where *n* is the total number of boxes in the document. Subsequently, this set of boxes is analysed in the next phase of the page segmentation algorithm in order to discover the visual blocks in the page.

Figure 2 shows an example of a simple HTML document and the rendered page. The bounds of the individual boxes are marked with dashed lines. As we may see, more text boxes may be generated for a continuous text because of the automatic line wrapping that is applied according to the CSS specification.

**Figure 2** An example of an HTML document and the boxes obtained by document rendering




### *4.2 Detecting basic visual areas*

In HTML documents, one of the following means may be used for creating visually separated blocks in the page:

- using a box that is visually separated from the remaining content, e.g., by a different background colour or a frame
- creating groups of boxes that are separated from the remaining content by other visual means that include line separators and rivers of white space.

In this phase of the page segmentation, we detect the boxes from the set $B$ that form standalone visual blocks, i.e., the first one of the above mentioned cases. We say that a box is *visually separated* if at least one of the following conditions holds:

- the box is created by the document root element and thus, it represents the whole rendered page
- the box is a text box or a replaced box as defined in Section 4.1. These boxes represent the actual document contents
- the background of the box is not transparent or there is a visible border defined around the box, i.e., the box is apparent on the resulting page.

Each box that meets the above definition is perceived by the user as a basic standalone unit of the page and we will call it a *basic visual area*. The bounds of this area correspond to the bounds of the appropriate box. Let $a(b_i)$ be a basic visual area that corresponds to the box $b_i \in B$. In this step, we create a set of basic visual areas:

$$A = \{a(b_i); b_i \in B \wedge b_i \text{ is visually separated}\} \tag{3}$$

In many cases, the areas are overlapping in the page. A typical example is a colour box in the page that contains several lines of text. Each text line is represented by a text box $b_i \in B$ and it is placed inside of the bounds of an element box $b_j \in B$; $i \neq j$ with a colour background. We say, that the text boxes $b_i$ are fully enclosed in the background box $b_j$. In our example in the Figure 2, this is the case of the paragraph <*p*> and its content boxes. In HTML documents, this is a very frequent situation that is simply created by two or more nested HTML elements with the appropriate visual properties.

Very rarely, two boxes may be detected in the document that overlap only partially; than means, the areas overlap but we cannot say that one of the areas is fully enclosed in the other one. In this case, we consider the box drawing order defined in the CSS specification and we say that the box $b_i$ is enclosed in $b_j$ when $b_i$ is drawn in front of $b_j$ according to the CSS specification.

We model the box overlapping by creating a tree of visual areas:

$$T_A = (A, E) \tag{4}$$

where $A$ is the set of basic visual areas (3) and $E$ is the set of tree edges, where $(a(b_j), a(b_i)) \in E$ if and only if $b_i$ is enclosed in $b_j$.

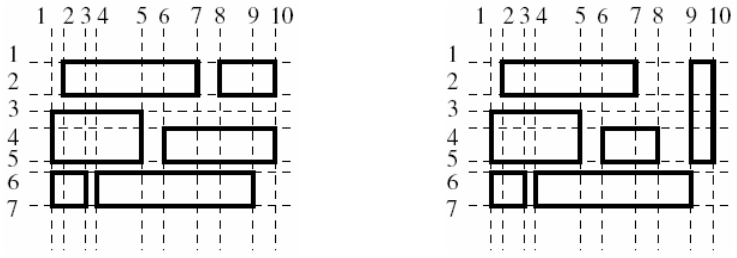
The resulting tree represents the visual area nesting as perceived by the user. Its root area represents the whole page (it corresponds to the document root element) and the leaf areas correspond to the individual pieces of text or other content. The remaining areas correspond to other boxes that are visually separated for example by a colour border or background.

The tree of basic areas obtained from the sample document from the Figure 2 is shown in the Figure 4(a). The <*div*> and <*i*> boxes are not visually separated and thus, they do not create a visual area. The <*p*> box is separated by a colour background and the remaining boxes are the text boxes and the document root.

### *4.2.1 Area grid*

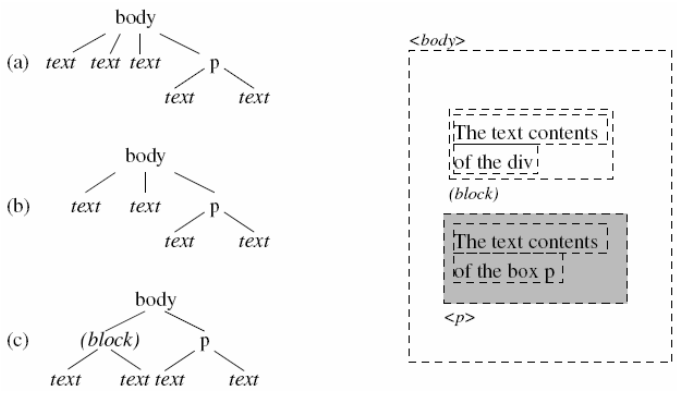
The above defined tree of visual areas represents the area nesting. For the further steps of the page segmentation, it is also necessary to represent the mutual positions of the areas within their parent area. For this purpose, we define an area grid $g_i$ for each area $a(b_i) \in A$ that describes the mutual positions of all the child areas of $a(b_i)$ in $T_A$. Examples of such a grid are shown in the Figure 3.

**Figure 3** Examples of two area grids used for line detection (a) with three lines detected, (b) with a box preventing the first two lines from being detected




(a) (b)

**Figure 4** The basic visual area tree, (a) the resulting tree after the line detection phase (b) the block detection phase (c) the resulting bounds of the detected visual areas




All the child areas of $a(b_i)$ are placed in a grid with variable column widths and row heights where each area occupies an arbitrary number of rows and columns. The position of each area is determined by its starting and ending column and row in the grid. Using this representation, we can quickly determine the mutual positions of an arbitrary pair of child areas. Similar grids are created for all the child areas recursively. These grids are primarily used for locating the directly neighbouring areas in the page and for the line and block detection as described in next chapters.

### *4.3 Text line detection*

The purpose of this phase is to join the areas that form a single text line. In our approach, we consider a text line to be the smallest visual area that is not broken to even smaller pieces. In order to detect the lines, we look for the areas that share the same rows in the appropriate area grid and the bounds of the line do not overlap with any other detected line or existing area. When found, we check if the areas are not visually separated from each other. In this case, we require that the areas have the same background colour and there is no separating border between the areas.

If the areas are not visually separated, we join them into a single area representing the whole line. The areas $a(b_i)$, $a(b_j) \in A$ that share the same parent area in $T_A$ will be replaced by a new area $a(b_i, b_j) \in A$. The bounds of the resulting area then cover all the joined areas and the area corresponds to a set of boxes instead of a single box. The set of child areas of the new area in $T_A$ is the union of the sets of child areas of $a(b_i)$ and $a(b_j)$.

For example, in the Figure 3, we detect three lines in the grid (a): the first line spans for the grid rows (1, 2), the second one is (3, 5) and the third one is (6, 7). In contrast, in the grid (b), the first two lines will not be detected because they would overlap with the box in the column 9. This corresponds to the fact that the areas form a more complicated structure that does not correspond to simple lines of text.

In our example in the Figure 4, the text ‘of the div’ is created by two boxes that correspond to the two elements in the original document. After applying the line detection algorithm, these two boxes are joined to a single one. The resulting tree of visual areas is shown in the Figure 4(b).

### *4.4 Block detection*

The last step of the page segmentation is the detection of visually consistent blocks. In this phase, we create new areas in $T_A$ that group together adjacent areas with the same style. The purpose of this step is to detect the paragraphs of a consistent style (most frequently the headings consisting of multiple lines or the simple paragraphs of text) that can be later classified as standalone areas. Two areas are considered to have a consistent style if they share the same average font size and style of their text contents.

Moreover, we consider the visual separators that may appear in the page and that can be created by other visually separated areas or by rivers of white space. For the detection of these separators, we use the block division algorithm published in Cai et al. (2003).

For each area, we have a set of separators and a set of child areas with the given style. When detecting the covering areas, we find the largest rectangular areas that cover the child areas with the same style. At the same time, we require that the new areas do not cover any of the separators. When such a covering area is found, it is added to the area tree at the appropriate point so that the tree still represents the area nesting.

In our example, the first two lines of text have a consistent style and they are not visually separated by any separator. We create a new area that covers these text lines. The resulting tree of areas is shown in the Figure 4(c) where the newly created area is marked as *block*.

### *4.5 Segmentation result*

As the result of the above described segmentation process, we obtain a tree of areas, where the root area corresponds to the whole page; the leaf areas correspond to the individual text lines and the remaining areas corresponds to some visually distinguished blocks in the page. This model is built based on both the visual properties of the page elements and various kinds of separators. Therefore, the obtained tree represents the visual page organisation as it is perceived by a user.

For the purpose of its further analysis, we serialise the obtained tree of areas to an XML file. This file contains the description of all the detected visual areas including their positions in the page and in the layout grid described in Section 4.2.1. For the leaf visual areas, the information about contained text boxes and their visual properties is included as well. Thus, each XML file contains all information necessary both for displaying the original page contents and for determining the visual properties of each detected area.

## 5 Page element classification

All the non-leaf nodes of the visual area tree obtained from the page segmentation correspond to some visual areas detected in the page. For each of these visual areas, we determine the values of various visual properties that are used for the area classification. The task of the classifier is to assign a class to each area based on these properties.

The classification process consists of two phases. In the learning phase, we train the classifier using a training set of segmented documents where the classes have been manually assigned to the individual areas. Then, in the classification phase, new

documents are segmented and the discovered areas are assigned classes using the previously trained classifier.

### 5.1 Obtaining the visual features of the areas

For the classification of large sets of documents from different sources, it is important that the chosen visual features be comparable for the whole set of previously unknown documents. Therefore, we have avoided using absolute values such as absolute font size and we prefer relative values of all features.

For obtaining the values of all the features, we have implemented a *feature extraction tool*. This software tool reads a set of XML files containing a segmentation result for a set of documents as described in Section 4.5 and it produces a file in the ARFF format that can be used as an input for the WEKA classifier described below. This file contains a list of all the visual areas in the document set where each area is represented by the values of the individual features.

The following visual features are computed for all the visual areas of all the processed documents.

#### 5.1.1 Font features

The font features are computed as the average values for the whole visual area evaluated weighted by the number of characters with a particular value of that feature. The following values are computed:

- *fontsize* – average font size in percent where the average font size of the whole document is considered to be 100%. Thus, the value tells whether the font size for a particular area is below or above average.
- *weight* – an average font weight from the range 0..1. The regular font characters are assigned the weight of 0; bold characters have the weight of 1.
- *style* – an average font style from the range 0..1 (normal or italic style) computed analogically to the average weight.

#### 5.1.2 Spatial features

Spatial features describe the position of the given area in the page and the relations to other areas.

- *aabove*, *abelow*, *aleft*, *aright* – the number of areas that are placed above, below, on the left and on the right of the area within its parent area.
- *relx*, *rely* – the relative position of the area within the whole page. 0 means the left edge or top of the page respectively, 1 means the right edge or bottom.

#### 5.1.3 Text features

The text content of an area is a text string obtained as a concatenation of all the text boxes covered with the area. The following values are computed for each area:

- *nlines* – number of text lines in the area.

- *tlength* – total length of the contained text in characters.
- *pdigits*, *plower*, *pupper*, *pspaces*, *ppunct* – percentages of digits, lowercase letters, uppercase letters, whitespaces and punctuation characters in the text content of the area.

### *5.1.4 Colour features*

Colour features include the colour properties of the area background and the average colour properties of the text. In order to obtain some comparable values, we do not consider the colours themselves but we compute the following values:

- *tlum* – an average text luminosity computed according to the WCAG recommendation (Caldwell et al., 2008). The appropriate formula in this recommendation considers the real properties of the human vision and therefore, it seems to be suitable for this purpose.
- *bglum* – the background colour luminosity computed using the same formula. If the given area has a transparent background, we consider the background colour of its parent area.
- *contrast* – the average colour contrast computed from the text and background luminosities according to the WCAG recommendation.
- *cperc* – the percentage of text of the same colour in the document. This value tells how much this text colour is unique or prevalent within the given document.

## *5.2 Training data preparation*

For preparing the training set of documents, we have implemented a graphical *annotation tool* that allows displaying a segmented page stored in an XML file and assigning a particular class to the displayed visual areas. The assigned class is then stored back to the XML file. The extraction tool can then include the assigned class in the input data for the classifier together with the computed feature values. The classifier input obtained this way is then used as the training dataset. This process corresponds to the training phase illustrated in Figure 1.

Moreover, the graphical annotation tool can be used for displaying the visual feature values of the individual detected areas and after the classification phase is finished, it is able to display the results obtained by the automatic classification and eventually, to compare them graphically with the manually assigned classes.

The overview of the classes used for annotation is in Table 1. These classes correspond to the individual parts of articles that are commonly used in internet sources. A special class *none* is automatically assigned to all remaining areas detected in the page that do not form part of the article contents.

It is a special feature of the dataset prepared this way that the number of areas assigned to the *none* class significantly exceeds the number of areas assigned to the other classes. This corresponds to the fact that in a normal web page, most of the detected areas do not correspond to any standalone part of the articles. Many of them correspond to the single text lines that form only a part of some paragraph, some areas correspond to other type of additional content.

**Table 1** The classes assigned to the individual visual areas

| | |
|---|---|
| `h1` | Main article heading |
| `h2` | Second-level heading in the article |
| `subtitle` | The article subtitle |
| `perex` | The leading paragraph of the article |
| `paragraph` | An ordinary paragraph |
| `date` | Publication date |
| `author` | Author name |
| `authordate` | Author and the date in a single area |
| `none` | Remaining areas that do not belong to the article |

### *5.3 Classification algorithms*

For the classification itself, we have used the University of Waikato WEKA data mining tool (http://www.cs.waikato.ac.nz/~ml/index.html) that offers a variety of implemented classification methods based on various principles. WEKA can read both the training and testing datasets produced by our software tools and it also allows evaluating the results of classification, mainly the precision and recall rates for the individual classes.

We have tested the following classification methods for our data:

- the J48 tree classifier which is an implementation of the Quinlan's (1993) C4.5 algorithm
- Bayesian network classifier
- multilayer perceptron – a neural network with backpropagation
- support vector machines.

The advantage of the J48 algorithm is that it produces a decision tree that is interesting for our further research. Therefore, we have used this algorithm in our previous works and we include it for comparison in this work. Bayesian network presents a classical classification method. We have chosen it for its reliable results in our previous experiments. The neural network classifier and the SVM classifier have been chosen for comparison with the results published by Song et al. (2004).

We have experimentally compared these algorithms on datasets we have created from real web data. All the selected methods produced acceptable results. The detailed data about our experiments are provided in section 6.

## 6 Experimental evaluation

For the experimental evaluation, we have implemented several tools according to the Figure 1 introduced in the overview. The segmentation tool implements the page segmentation algorithm described in Section 4. For each segmented document, the resulting tree of visual areas is serialised to XML and stored to a separate file. Secondly, the graphical annotation tool allows to display the XML files obtained from segmentation and to manually assign classes to the individual areas. And finally, the feature extraction

tool processes a set of segmented pages (either annotated or not annotated), it computes the values of all the visual features for all the visual areas contained in these files and stores the obtained values into an ARFF file that may be used as a training or testing dataset for the WEKA classifier. All our tools have been implemented in the Java environment.

We have tested our classification approach on a set of web documents obtained from the websites of various newspapers worldwide. We segmented all the downloaded documents using our segmentation tool and we manually annotated a subset of documents from each website. Subsequently, we have computed the visual features of all the visual areas detected in all the documents and tested the classification algorithms on various subsets of these data. As the next step, we have tested selected algorithms in situations that correspond to the assumed real use of the method.

**Table 2** The websites used as sources of the testing documents and the corresponding RSS feed URLs

| *Website* | *RSS feed* |
|---|---|
| Aktualne.cz | http://aktualne.centrum.cz/export/rss-hp.phtml |
| Idnes.cz | http://servis.idnes.cz/rss.asp?c=zpravodaj |
| Novinky.cz | http://novinky.cz/rss2/ |
| Lupa.cz | http://rss.lupa.cz/2/clanky/ |
| Root.cz | http://www.root.cz/rss/clanky/ |
| Lidovky.cz | http://www.lidovky.cz/export/rss.asp?r=ln domov |
| El Mundo | http://rss.elmundo.es/rss/descarga.htm?data2=4 |
| El Pais | http://www.elpais.com/rss/feed.html?feedId=1022 |
| La Vanguardia | http://feeds.feedburner.com/lavanguardia/alminuto |
| LA Times | http://feeds.latimes.com/latimes/news?format=xml |
| USA Today | http://rssfeeds.usatoday.com/usatoday-NewsTopStories |
| Zeit Online | http://newsfeed.zeit.de/index |
| Focus Online | http://rss.focus.de/fol/XML/rss folnews.xml |
| Stern.de | http://www.stern.de/feed/standard/all/ |
| Le Figaro | http://rss.lefigaro.fr/lefigaro/laune?format=xml |
| Le Parisien | http://rss.leparisien.fr/leparisien/rss/actualites-a-la-une.xml |
| 20minutes.fr | http://www.20minutes.fr/rss/flux/une.xml |
| Corriere Della Sera | http://www.corriere.it/rss/homepage.xml |
| La Repubblica | http://rss.feedsportal.com/c/32275/f/438637/index.rss |
| Il Sole 24 ore | http://feeds.ilsole24ore.com/c/32276/f/438662/index.rss |

## *6.1 Source data*

We have manually created a list of source websites that is provided in Table 2. Each of these websites provides an RSS feed that contains the URLs of the latest published articles. We have automatically segmented all the articles in the RSS feeds. From each

website, from five to 15 documents have been obtained according to the different sizes of the RSS feeds. In total, we have obtained 559 documents and during the segmentation, 198,129 visual areas have been detected in these documents.

Using the annotation tool, we have manually annotated four documents from each website. In total, we have manually annotated 784 areas in these documents. From the annotated documents, we have created various training and testing datasets for testing the classification method in different situations. The remaining segmented documents have been used for visual evaluation of the classification results using our graphical tool.

We have used the obtained datasets for two basic experiments:

1 We have compared the precision and recall of the selected classification algorithms listed in 5.3 in two different situations regarding the combination of training a testing sources. The aim of these experiments was to check the fitness of the individual algorithms for the given purpose. These experiments are further discussed in the Section 6.2. Based on the obtained values, we have chosen the SVM and BayesNet classifiers for further experiments.

2 With the selected classification methods, we have run thorougher experiments in order to verify the classification results for the expected use cases of our approach. We have tested more different combinations of the training and testing sets in order to minimise the influence of possible specific features of the individual web sources. These experiments are discussed in the Section 6.3.

### *6.2 Classification algorithm evaluation*

According to our previous experiments, the resulting precision and recall of the individual classification algorithms greatly depends on the actual combination of the datasets used for training and testing. This is caused mainly by the difference in the way of the visual article presentation among various web sources where some sources are more similar to each other than the other ones. Therefore, the first step of our experimental evaluation was to find the classification algorithms that give satisfactory results for both the favourable combinations, where the way of visual presentation is similar in the training and testing data sources and the unfavourable combinations, where the visual presentation differs significantly. Based on the obtained results, we choose the algorithms that should be the most suitable for the application on real use-cases that are described in the Section 6.3.

For comparing the classification algorithms, we have created two datasets:

- The first dataset is used for testing the algorithms in an ‘optimistic’ situation when the training and testing documents come from the same sources. Both the training and testing sets contain two annotated documents from each source in Table 2, however, different documents from these sources have been used for creating the training and testing sets.
- The second dataset simulates a ‘pessimistic’ variant when the training set of pages is created from different sources than the testing set. We have randomly split the set of sources in two parts of the same size and we have used all the annotated documents from these sources to create the training and testing sets respectively.

These datasets should represent the favourable and unfavourable conditions for the classification. Now, our task is to choose the algorithms that give satisfactory results for both datasets.

**Table 3** Obtained precision, recall and F-measure values of the classification algorithms for the individual classes

| *Class* | *J48* | | | *BayesNet* | | |
|---|---|---|---|---|---|---|
| | *P* | *R* | *F* | *P* | *R* | *F* |
| none | 0.986 | 0.992 | 0.989 | 0.996 | 0.936 | 0.965 |
| h1 | 0.727 | 0.889 | 0.800 | 0.400 | 0.889 | 0.552 |
| h2 | 1.000 | 0.737 | 0.848 | 0.673 | 0.921 | 0.778 |
| subtitle | 0.500 | 0.143 | 0.222 | 0.081 | 0.429 | 0.136 |
| perex | 0.455 | 0.714 | 0.556 | 0.524 | 0.786 | 0.629 |
| paragraph | 0.743 | 0.663 | 0.701 | 0.386 | 0.961 | 0.550 |
| author | 0.200 | 0.091 | 0.125 | 0.273 | 0.273 | 0.273 |
| date | 1.000 | 0.222 | 0.364 | 0.292 | 0.778 | 0.424 |
| authordate | 1.000 | 0.300 | 0.462 | 0.600 | 0.600 | 0.600 |
| | *Perceptron* | | | *SVM* | | |
| | *P* | *R* | *F* | *P* | *R* | *F* |
| none | 0.988 | 0.982 | 0.985 | 0.991 | 0.986 | 0.988 |
| h1 | 0.889 | 0.889 | 0.889 | 0.842 | 0.889 | 0.865 |
| h2 | 0.906 | 0.763 | 0.829 | 0.886 | 0.816 | 0.849 |
| subtitle | 0.500 | 0.286 | 0.364 | 0.500 | 0.143 | 0.222 |
| perex | 0.417 | 0.714 | 0.526 | 0.524 | 0.786 | 0.629 |
| paragraph | 0.612 | 0.795 | 0.691 | 0.674 | 0.787 | 0.726 |
| author | 0.000 | 0.000 | 0.000 | 0.417 | 0.455 | 0.435 |
| date | 0.000 | 0.000 | 0.000 | 0.500 | 0.556 | 0.526 |
| authordate | 0.000 | 0.000 | 0.000 | 0.700 | 0.700 | 0.700 |

Note: The training and testing dataset are coming from the same sources (optimistic variant).

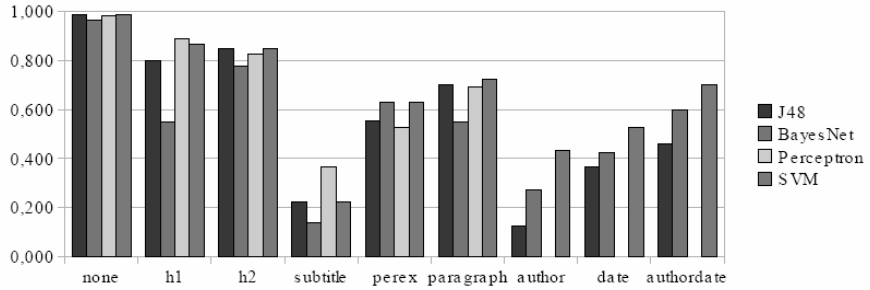
Table 3 shows the classification results for the first dataset. We include the values of precision, recall and the F-measure value computed as

$$F = \frac{2 \cdot P \cdot R}{P + R}$$

The graphical comparison of the F-measure values for the individual classification methods and classes is in Figure 5. We can see that all the methods are able to identify most visual areas that belong to the *h1*, *h2*, *paragraph* and *perex* classes. Such visual areas are contained in most of the annotated documents and they usually have some

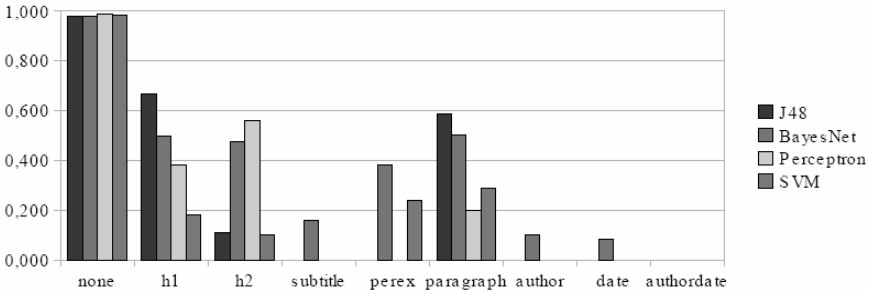
specific visual features that make them easy to recognise. On the other hand, some methods fail for the remaining classes such as *author* or *date*. These visual areas are often not contained in the documents and the way of their presentation is very variable. The *subtitle* class is used in very few web sources and therefore, it is rarely identified correctly. SVM classifier gives stable results for all the classes, the J48 and BayesNet classifier work for most classes too.

**Figure 5** Comparison of the F-measures of the algorithms for individual classes (optimistic variant)




The results for the pessimistic variant are quite different. In this case, the visual properties of the individual area classes are much more different than in the optimistic case. The obtained values are shown in Table 4 and the graphical comparison in Figure 6. In this situation, only the BayesNet classifier was able to identify at least some visual areas from all the classes; other methods fail mainly for the classes with more variable presentation.

**Figure 6** Comparison of the F-measures of the algorithms for individual classes (pessimistic variant)




The results show that none of the classification methods can be considered to be the most suitable one for the classification of visual areas. However, SVM gives reliable results in situations where the content is presented in a consistent manner. On the other hand, BayesNet classifier is usable even when the content presentation style varies significantly.

For these reasons, we have selected both the SVM and the BayesNet classifiers for running more tests that simulate the expected use cases of the proposed method.

**Table 4** Obtained precision, recall and F-measure values of the classification algorithms for the individual classes

| *Class* | *J48* | | | *BayesNet* | | |
|---|---|---|---|---|---|---|
| | *P* | *R* | *F* | *P* | *R* | *F* |
| none | 0.991 | 0.967 | 0.979 | 0.996 | 0.960 | 0.978 |
| h1 | 0.714 | 0.625 | 0.667 | 0.341 | 0.938 | 0.500 |
| h2 | 0.079 | 0.176 | 0.109 | 0.429 | 0.529 | 0.474 |
| subtitle | 0.000 | 0.000 | 0.000 | 0.095 | 0.500 | 0.160 |
| perex | 0.000 | 0.000 | 0.000 | 0.250 | 0.813 | 0.382 |
| paragraph | 0.487 | 0.736 | 0.586 | 0.375 | 0.757 | 0.501 |
| author | 0.000 | 0.000 | 0.000 | 0.063 | 0.300 | 0.103 |
| date | 0.000 | 0.000 | 0.000 | 0.043 | 1.000 | 0.083 |
| authordate | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| | *Perceptron* | | | *SVM* | | |
| | *P* | *R* | *F* | *P* | *R* | *F* |
| none | 0.977 | 0.998 | 0.987 | 0.978 | 0.994 | 0.986 |
| h1 | 0.308 | 0.500 | 0.381 | 0.333 | 0.125 | 0.182 |
| h2 | 0.875 | 0.412 | 0.560 | 0.333 | 0.059 | 0.100 |
| subtitle | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| perex | 0.000 | 0.000 | 0.000 | 0.333 | 0.188 | 0.240 |
| paragraph | 0.810 | 0.115 | 0.201 | 0.537 | 0.196 | 0.287 |
| author | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| date | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| authordate | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

Note: The training and testing dataset are coming from different sources (pessimistic variant).

### *6.3 Results for expected use cases*

The aim of the following experiment was to test the method behaviour for two typical use cases of our proposed method of automatic article annotation:

1 The training documents come from the same source as the later annotated documents. In this case, the method is used for identifying the articles or parts of articles coming from a single, previously known source.

2 There is a fixed set of training documents obtained from larger number of sources and the documents being automatically annotated come from other, previously unknown source or sources. This can be the case of sources where no training examples are available in advance.

We consider the first case to be more probable in practical application since the article sources are usually known in advance. However, the second case may occur too; for example when the data sources may be dynamically added by users or when using the method for processing ad hoc documents found on the web.

The main difference from the experiments previously discussed in Section 6.2 is that we have tested more combinations of sources for creating the training and testing sets in order to eliminate possible specific features of the individual web sources.

In the first case, only a single source is considered. Therefore, we had to increase the number of annotated documents. We have annotated 20 documents coming from the same source and we have used 6 documents for creating the training set and the rest of the documents for testing. We have repeated this for four different sources. Table 5 shows the average results for the individual classes.

**Table 5** Achieved results for the training documents coming from the same source as the later annotated documents

| *Class* | *BayesNet* | | | *SVM* | | |
|---|---|---|---|---|---|---|
| | *P* | *R* | *F* | *P* | *R* | *F* |
| none | 0.999 | 0.917 | 0.957 | 1.000 | 0.999 | 0.999 |
| h1 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| h2 | 0.727 | 1.000 | 0.842 | 1.000 | 1.000 | 1.000 |
| subtitle | 0.500 | 1.000 | 0.667 | 1.000 | 1.000 | 1.000 |
| perex | 0.286 | 1.000 | 0.444 | 0.800 | 1.000 | 0.889 |
| paragraph | 0.381 | 0.971 | 0.547 | 0.986 | 0.986 | 0.986 |
| author | 0.273 | 1.000 | 0.429 | 0.750 | 1.000 | 0.857 |
| date | 0.200 | 0.750 | 0.316 | 1.000 | 0.500 | 0.667 |
| authordate | 0.571 | 1.000 | 0.727 | 1.000 | 1.000 | 1.000 |

Since all the documents from a single source usually maintain a consistent style of presentation, we can see that both methods give very reliable results. In this case the SVM classifier gives better results that the BayesNet classifier.

For simulating the second use case, we have used the annotated documents from one source as a testing set and the documents from the remaining sources as the training set. Again, we have repeated the test for four different combinations of sources. The results are shown in Table 6.

**Table 6** Achieved results for documents coming from other, previously unknown source

| *Class* | *BayesNet* | | | *SVM* | | |
|---|---|---|---|---|---|---|
| | *P* | *R* | *F* | *P* | *R* | *F* |
| none | 0.998 | 0.879 | 0.934 | 0.971 | 0.998 | 0.984 |
| h1 | 1.000 | 0.750 | 0.857 | 0.667 | 0.500 | 0.571 |
| h2 | 0.833 | 0556 | 0.667 | 0.800 | 0.444 | 0.571 |
| subtitle | 0.254 | 0.928 | 0.399 | 0.000 | 0.000 | 0.000 |
| perex | 0.400 | 1.000 | 0.571 | 0.136 | 0.300 | 0.187 |
| paragraph | 0.247 | 0.899 | 0.388 | 0.972 | 0.507 | 0.667 |
| author | 0.250 | 0.667 | 0.364 | 0.000 | 0.000 | 0.000 |
| date | 0.500 | 0.333 | 0.400 | 0.000 | 0.000 | 0.000 |
| authordate | 0.125 | 0.167 | 0.143 | 0.000 | 0.000 | 0.000 |

In this case, the BayesNet method gives more reliable results. Generally, the results are acceptable for more common classes such as headings and paragraphs. The values for

less frequent and differently presented classes such as author and date are quite low for the reasons mentioned above. The results in both cases overcome our previous results published in Burget and Rudolfová (2009) where we have used a lower number of visual features for the visual area classification.

### *6.4 Comparison with other approaches*

In order to compare our results with the results published in Song et al. (2004), we have followed the same scenario as described in the paper. With all the annotated data, we have conducted a five-fold cross validation using the SVM classification method.

Table 7 shows the results of the cross validation for our dataset. In Song et al. (2004), the classes are defined in a more general way according to the relevance of the given content block to the article: Actual parts of the article are marked with *Level 4*, other areas of the page relevant to the document topic or useful for the user are marked with *Level 3* and *Level 2* and the remaining parts (noisy information) are marked with *Level 1*. For the classification, the levels 2 and 3 are regarded as a single class. Since in our dataset, we have only annotated the parts of the article itself, all our classes different from *none* correspond to *Level 4* and the *none* class corresponds basically to the levels 1 to 3. We have mapped the classes as described. The comparisons of the obtained results with the results of Song et al. (2004) are in the Table 8.

**Table 7** Results of the five-fold cross validation for our method by classes

| *Class* | *P* | *R* | *F* |
|---|---|---|---|
| none | 0.995 | 0.997 | 0.996 |
| h1 | 0.795 | 0.861 | 0.827 |
| h2 | 0.767 | 0.836 | 0.800 |
| subtitle | 1.000 | 0.417 | 0.588 |
| perex | 0.826 | 0.731 | 0.776 |
| paragraph | 0.912 | 0.907 | 0.910 |
| author | 0.706 | 0.414 | 0.522 |
| date | 0.778 | 0.667 | 0.718 |
| authordate | 0.750 | 0.882 | 0.811 |

We can see that the results for the *Level 4* class are comparable or slightly better for our method. The results for the remaining classes are more difficult to compare because our *none* class does not directly correspond to a single class in the reference paper. However, the results demonstrate that the visual features of the document content can be used for automatic content annotation with a sufficient precision.

**Table 8** Comparison of the five-fold cross validation results

| *Class* | *Our method* | | | *Song et al. (2004)* | | |
|---|---|---|---|---|---|---|
| | *P* | *R* | *F* | *P* | *R* | *F* |
| Level 1 | 0.995 | 0.997 | 0.996 | 0.763 | 0.776 | 0.769 |
| Level 2 | | | | 0.796 | 0.804 | 0.800 |
| Level 4 | 0.917 | 0.873 | 0.895 | 0.839 | 0.770 | 0.803 |

## 7 Applications and possible improvements

The proposed method is suitable for pre-processing the documents before their further automatic indexing, classification or other processing by a computer. It can be used for cleaning the document from the noisy information or for supplying the further processing tools with an additional information about the individual parts of the document structure.

The results show that our approach is able to automatically annotate visual areas in a page with quite a high precision at least when applied to a single source of documents. In that case, it is able to correctly annotate most of the repeating parts of the articles.

When used on new, previously unknown sources, the method can be used for detecting the most common parts of the article such as headings. For certain applications such as the page preprocessing for data mining mentioned in the introduction, the obtained precision and recall of the classification seems to be satisfactory. This precision allows to determine most of the visual areas that belong to the article and to detect the approximate bounds of the main article on the page.

In order to determine the bounds of articles in previously unknown documents more precisely, the method could be extended by using a kind of heuristics that would represent some commonly used habits in the article publication on the web. For example, in Burget (2009), we have proposed and tested some heuristics concerning the layout of the text placed below a header.

Since only visual attributes of the contents are used, the presented approach is independent on the document language. Regarding the properties of the text contents of the document, our approach is only based on some statistical properties of the text. As the next step, the visual features could be combined with the traditional text classification methods by analysing for example the term frequencies in the individual visual areas. However, this would introduce the dependency on a particular language.

## 8 Conclusions

In this paper, we have introduced a method of interesting area detection in the web pages. We have proposed a method of page segmentation tailored especially for this purpose in order to detect the basic visual blocks in the page that have a consistent visual style. Further, we have proposed the way of the detected block classification based on a set of their visual features. Finally, we have tested the proposed method on real-world data.

According to the results of the experiments, the method is suitable for an approximate detection of a published article in the web page. In case of using only one, previously known source of documents, the obtained precision allows a reliable identification of article elements in the page.

In comparison to the methods based on document code (DOM) analysis, our approach allows extracting more different visual features of the document contents, especially regarding the spatial and colour features. As shown by the experimental results, this allows achieving an interesting precision while preserving independence on the document language. However, both the visual features and the DOM model can be arbitrarily combined. The combination of the visual features with the information obtained from the DOM model seems to be a promising way for improving the quality of the page segmentation and the classification itself.

## Acknowledgements

This work was partially supported by the BUT FIT grant FIT-S-10-2 and the research plan MSM0021630528.

## References

Baumgartner, R., Flesca, S. and Gottlob, G. (2001) 'Visual web information extraction with lixto', *VLDB '01: Proceedings of the 27th International Conference on very Large Data Bases*, Morgan Kaufmann Publishers Inc., pp.119–128.

Bos, B., Lie, H.W., Lilley, C. and Jacobs, I. (1998) 'Cascading style sheets, level 2', *CSS2 Specification: The World Wide Web Consortium*.

Burget, R. and Rudolfová, I. (2009) 'Web page element classification based on visual features', *IEEE Computer Society, 1st Asian Conference on Intelligent Information and Database Systems ACIIDS 2009*, pp.67–72.

Burget, R. (2004) 'Hierarchies in HTML documents: linking text to concepts', *IEEE Computer Society, 15th International Workshop on Database and Expert Systems Applications*, pp.186–190.

Burget, R. (2007) 'Layout based information extraction from HTML documents', *IEEE Computer Society, ICDAR 2007*, pp.624–629.

Burget, R. (2009) 'Automatic web document restructuring based on visual information analysis', *Proceedings of the 6th Atlantic Web Intelligence Conference – AWIC'2009*, Vol. 10, Springer Verlag.

Cai, D., Yu, S., Wen, J-R. and Ma, W-Y. (2003) 'VIPS: a vision-based page segmentation algorithm', *Microsoft Research*.

Caldwell, B., Cooper, M., Reid, L.G. and Vanderheiden, G. (2008) 'Web content accessibility guidelines 2.0', *The World Wide Web Consortium*.

Chakrabarti, D., Kumar, R. and Punera, K. (2008) 'A graph-theoretic approach to webpage segmentation', *17th International World Wide Web Conference*.

Cohen, W.W., Hurst, M. and Jensen, L.S., (2002) 'A flexible learning system for wrapping tables and lists in html documents', *ACM, WWW '02: Proceedings of the 11th international conference on World Wide Web*, pp.232–241.

Freitag, D. (1997) 'Using grammatical inference to improve precision in information extraction', *ICML-97Workshop on Automata Induction, Grammatical Inference, and Language Acquisition*.

Gupta, S., Kaiser, G., Neistadt, D. and Grimm, P. (2003) 'DOM-based content extraction of HTML documents', *WWW2003 proceedings of the 12 Web Conference*, pp.207–214.

Kosala, R., Bussche, J.V.D., Bruynooghe, M. and Blockeel, H. (2002) 'Information extraction in structured documents using tree automata induction', *PKDD '02: Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, pp.299–310, Springer-Verlag.

Kovacevic, M., Diligenti, M., Gori, M. and Milutinovic, V. (2002) 'Recognition of common areas in a web page using visual information: a possible application in a page classification', *IEEE Computer Society, ICDM '02*, p.250.

Kushmerick, N. (1997) Wrapper induction for information extraction', PhD thesis.

Kushmerick, N. (2000) *Wrapper Verification: World Wide Web*, Vol. 3, No. 2, pp.79–94.

Lin, S-H. and Ho, J-M. (2002) 'Discovering informative content blocks from web documents', *ACM, KDD '02: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.588–593.

Liu, L., Pu, C. and Han, W. (2001) 'An XML-enabled data extraction toolkit for web sources', *Inf. Syst*., Vol. 26, No. 8, pp.563–583.

Mukherjee, S., Yang, G., Tan, W. and Ramakrishnan, I., (2003) 'Automatic discovery of semantic structures in HTML documents', *IEEE Computer Society, International Conference on Document Analysis and Recognition*.

Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Song, R., Liu, H., Wen, J-R. and Ma, W-Y. (2004) 'Learning block importance models for web pages', *ACM, WWW '04: Proceedings of the 13th international conference on World Wide Web*, pp.203–211.

Yi, L., Liu, B. and Li, X. (2003) 'Eliminating noisy information in web pages for data mining', *ACM, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Yu, S., Cai, D., Wen, J-R. and Ma, W-Y. (2002) 'Improving pseudo-relevance feedback in web information retrieval using web page segmentation', *Microsoft Research*.

## B.3 Modelling Visually Presented Element Relationships in Web Documents

Burget, R.; Smrz, P.: Extracting Visually Presented Element Relationships from Web Documents. *International Journal of Cognitive Informatics and Natural Intelligence.* vol. 7, no. 2. April 2013: pp. 13–29. ISSN 1557-3958.

# Extracting Visually Presented Element Relationships from Web Documents

*Radek Burget, Faculty of Information Technology, IT4Innovations Centre of Excellence, Brno University of Technology, Brno, Czech Republic*

*Pavel Smrz, Faculty of Information Technology, IT4Innovations Centre of Excellence, Brno University of Technology, Brno, Czech Republic*

## ABSTRACT

*Many documents in the World Wide Web present structured information that consists of multiple pieces of data with certain relationships among them. Although it is usually not difficult to identify the individual data values in the document text, their relationships are often not explicitly described in the document content. They are expressed by visual presentation of the document content that is expected to be interpreted by a human reader. In this paper, the authors propose a formal generic model of logical relationships in a document based on an interpretation of visual presentation patterns in the documents. The model describes the visually expressed relationships between individual parts of the contents independently of the document format and the particular way of presentation. Therefore, it can be used as an appropriate document model in many information retrieval or extraction applications. The authors formally define the model, the authors introduce a method of extracting the relationships between the content parts based on the visual presentation analysis and the authors discuss the expected applications. The authors also present a new dataset consisting of programmes of conferences and other scientific events and the authors discuss its suitability for the task in hand. Finally, the authors use the dataset to evaluate results of the implemented system.*

*Keywords:* *Document Analysis, Element Relationships, Logical Document Structure, Page Segmentation, Web Documents*

## INTRODUCTION

The World Wide Web is traditionally viewed as a web of linked documents. A great research effort has been put to the analysis and modeling of the relationships among the individual documents (Page et al., 1999) or even analyzing semantic document relationships in order to obtain more information about the web organization (Luo et al., 2009; Luo et al., 2011). From this point of view, the documents are usually regarded as atomic units with certain properties such as the individual keyword frequencies. On the other hand, a remarkably less effort has been devoted to modeling the information structure in the documents themselves.

The WWW documents often present structured information that consists of multiple pieces of data of different kinds together with certain relationships among them. A typical

DOI: 10.4018/ijcini.2013040102

example may be a conference programme that consists of speech titles, times, places and author names. However, the relationships are often not explicitly described in the document content. They are expressed by different, mostly visual means and the human reader is expected to interpret the visual presentation of the content appropriately in order to assign for example the appropriate author and time to a speech title.

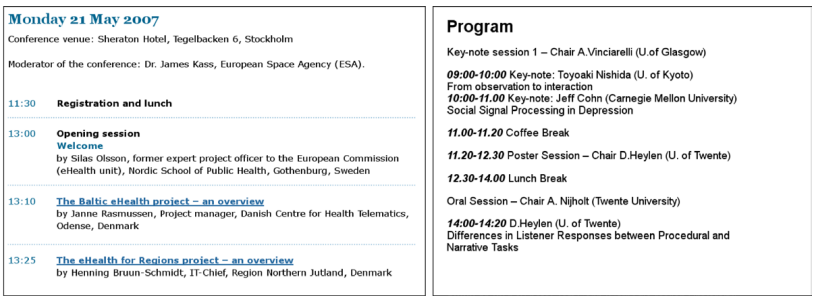
Existing approaches to structured information identification in web documents are usually based on an analysis of a larger set of documents that follow the same presentation guidelines. Then, based on a set of sample documents, we may infer a set of rules that may be later applied to other documents that follow the same guidelines. However, this does not solve a very frequent situation when we have a set of documents where each one comes from a different author and follows a different presentation style.

Let's consider two conference programmes presented in Figure 1. Both documents provide information about conference sessions, starting times, and the titles and authors of the individual presentations. However, this information is presented differently regarding the content layout, order of the individual data fields, colors and other properties and only a single exemplar of each such document is available. Moreover, the document formats may be different (for example, the HTML or PDF documents may be used).

Despite of the different formats, for a human reader, the presented relationships between the content elements remain the same and they correspond to the structure shown in Figure 2. In this example, both documents assign some times and sessions to the individual speech titles and authors. These relationships are presented visually by different font properties, indentation and other means that allow the reader to interpret the relationships without reading the text or even without understanding the used language. We may expect that these logical relationships are similar in all the conference programmes independently on how they are actually presented. Generally, we may expect that the documents presenting data of the same topic will share the same logical relationships between the individual content elements although presented in different ways. To give more examples: Published articles present the relationships between their title, authors, date of publication or even the sections and subsections. Timetables represent the relationships between the lines, places and times, etc.

In this paper, we propose a hierarchical *logical relationship model* that explicitly models the intra-document logical relationships that may be obtained by interpreting the visual presentation of the contents. This model has applications in information extraction, retrieval and other areas. Moreover, we address the problem of the automatic discovery of the logical relationships in a document. We analyze the visual presentation and content features that can be examined in order to obtain the logical relationship model. Lastly, we evaluate the proposed approach on real-world documents and we show that it can give comparable results for different document from various sources.

*Figure 1. Different presentation styles of conference programmes*



**Monday 21 May 2007**

Conference venue: Sheraton Hotel, Tegelbacken 6, Stockholm

Moderator of the conference: Dr. James Kass, European Space Agency (ESA).

11:30 **Registration and lunch**

13:00 **Opening session**
**Welcome**
by Silas Olsson, former expert project officer to the European Commission (eHealth unit), Nordic School of Public Health, Gothenburg, Sweden

13:10 **The Baltic eHealth project – an overview**
by Janne Rasmussen, Project manager, Danish Centre for Health Telematics, Odense, Denmark

13:25 **The eHealth for Regions project – an overview**
by Henning Bruun-Schmidt, IT-Chief, Region Northern Jutland, Denmark

**Program**

Key-note session 1 – Chair A.Vinciarelli (U.of Glasgow)

***09:00-10:00*** Key-note: Toyoaki Nishida (U. of Kyoto)
From observation to interaction
***10:00-11.00*** Key-note: Jeff Cohn (Carnegie Mellon University)
Social Signal Processing in Depression

***11.00-11.20*** Coffee Break

***11.20-12.30*** Poster Session – Chair D.Heylen (U. of Twente)

***12.30-14.00*** Lunch Break

Oral Session – Chair A. Nijholt (Twente University)

***14:00-14:20*** D.Heylen (U. of Twente)
Differences in Listener Responses between Procedural and Narrative Tasks

## RELATED WORK

In the area of information extraction from web documents, several authors already noted that the hierarchical relationships among the content blocks may be used for refining the identification of the particular information blocks in documents (Burget, 2004; You et al., 2013). However, in these methods, the content hierarchy is usually constructed based on some heuristics during the information extraction process and there is no explicit model defined for representing the intra-document relationships. Therefore, in this paper, we aim to create an application-independent model of the logical relationship constructed using a clearly defined algorithm.

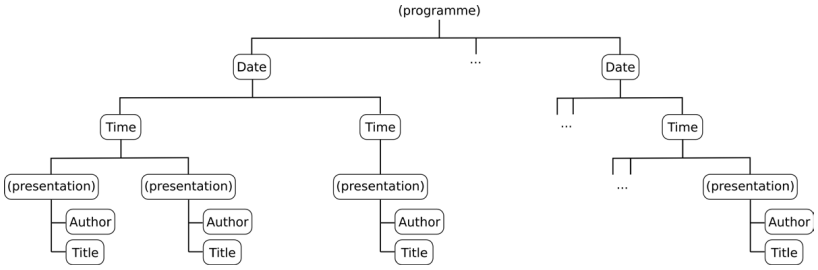
The problem of the relationship representation and discovery in documents is closely related to the areas of document layout analysis and logical structure discovery. However, unlike our proposed logical relationship model, the logical structure discovery is usually domain-oriented as explained further.

*Logical structure* of electronic documents has been studied by many authors for quite a long time. It is usually defined as a hierarchy of page segments that correspond to some visually distinguished components of its contents (Klink et al., 2000; Namboodiri & Jain, 2007; Shreve, 2006; Summers, 1995; Yashiro et al., 1989). The logical structure should be distinguished from the *layout structure*: the layout (or geometric) structure models the relationships between the document segments based on their visual presentation; the logical structure focuses on logical relationships that are given by the expected meaning of the document segments (Klink et al., 2000; Yashiro et al., 1989). The resulting logical structure is commonly represented using hierarchical structures (trees) (Klink et al., 2000; Summers, 1995) or grammars (Yashiro et al., 1989).

Discovering the logical structure usually includes assigning a meaning to the individual discovered components. The assigned meaning may be either a generic document section such as title, heading, footnote, list, etc. (Klink et al., 2000; Shreve, 2006) or a domain-specific meaning when focusing on a specific application (Nojoumian & Lethbridge, 2007; Rauf et al., 2011). Some authors even limit the logical structure discovery to the discovery of important parts of the document without explicitly modeling their relationships; for example in Luong et al. (2010), important parts of scholar articles are identified, which may be rather viewed as a classification task.

The discovery of the logical structure is usually based on an analysis of the page layout together with different visual features of the content. Shreve (2006) notes that the logical structure reflects cultural norms of document organization and the logical relationships of document elements, and that the relationships of logical structure to physical layout are also culturally determined. Stoffel and Spretke (Stoffel et al., 2010) use the positions of the text lines, their indentation, spacing and font style for the logical document structure discovery. Similarly, in Klink et al. (2000) and Namboodiri and Jain (2007), the document layout is analyzed in order to obtain the logical structure. In our older paper (Burget, 2004), we have also proposed a

*Figure 2. Expected logical structure of a conference programme*



rule-based approach to HTML document code analysis for obtaining the logical structure.

Basically, the approach shared by all the mentioned approaches consists of three steps that are often analyzed together:

1. *Obtaining the physical layout* of the content by applying a kind of page segmentation or layout analysis algorithm.
2. *Transforming the layout to structure* based on pre-defined or learned rules (common document structure recognition in Klink et al. (2000)).
3. *Interpretation of selected segments* by assigning a meaning to them based on given rules (domain dependent labeling (Klink et al., 2000)) or classification (Luong et al., 2010).

In our paper, we focus on the second step, i.e. creating a domain-independent document model that reflects the content presentation and interprets the cultural norms of the presentation as mentioned above. However, the domain knowledge is often necessary for a correct interpretation the logical relationships between the presented elements (Namboodiri & Jain, 2007). Therefore, in section "LRM Refinement by Adding Domain Knowledge", we also propose a way of incorporating domain knowledge for improving the obtained logical structure.

On the other hand, unlike most of the mentioned methods, we don't claim to assign a meaning to the individual parts of the obtained structure (the third step listed above). We consider this to be one of the possible applications of the proposed model, as we mention in section "Logical Relationship Model in Applications".

## LOGICAL RELATIONSHIP MODEL

When modeling the document structure, two kinds of models are usually distinguished (Klink et al., 2000; Yashiro et al., 1989): The *layout structure* (also called a *physical model*) describes the division of the document content to information blocks laid out on a page or pages. Generally, it is a hierarchical model that describes the basic blocks created for example by page headers, columns and other visually identifiable blocks in the documents. These blocks may be further divided to smaller sub-blocks. This model is domain-independent and it is created based on the page organization analysis.

On the other hand, the *logical structure* is domain-dependent: it assigns a meaning to selected parts of the document content and it represents the logical relationships between them as for example the hierarchical relationships between the semantic components such as headings, sub-headings and paragraphs in a document (Summers, 1995).

The purpose of the proposed Logical Relationship Model (LRM) is to provide a formally defined intermediate step between the layout structure and the logical structure. We claim that the interpretation of the visually presented relationships in the document should be separated from assigning the meaning to the individual parts of the document.

The LRM represents the logical relationships between the individual parts of the document content as they are expressed by visual means and as they are interpreted by a human reader. In the same time, it remains domain-independent although the domain knowledge may be used additionally for refining the model as proposed in section "LRM Refinement by Adding Domain Knowledge". Then the LRM may be used as a general model of the document suitable for different applications.

### The Layout and Logical Relationship Model

For defining the LRM, we will use the layout (physical) model of the document as defined by many authors, e.g. (Klink et al., 2000). Both the layout model and the LRM represent the relationships between *content elements*. With a content element we understand the smallest identifiable piece of the document content

(usually text) that can be viewed as an atomic unit. For our purpose, we will define the content element as follows:

**Definition 1 (Content Element):** Each displayed line of the text is considered to be a content element as long as it is visually consistent. For the text lines that consist of multiple parts with different visual properties, we consider the individual visually consistent parts to be separate content elements.

The difference between the layout model and the LRM is illustrated in Figure 3. Both trees correspond to the first conference programme displayed in Figure 1. The left tree shows the layout model. The whole page is split to several areas: a header (*Area 1*) and several items of the conference programme (*Area 2*, *Area 3*, etc.) visually separated by whitespace or other delimiters. The child nodes of each area correspond to the content elements or other areas that are placed inside the given area on the page. Further, we will call the detected visually separated areas the *content blocks*. As we can see from the example, the content blocks may be nested. The content elements then form the leaf nodes of the layout structure tree.

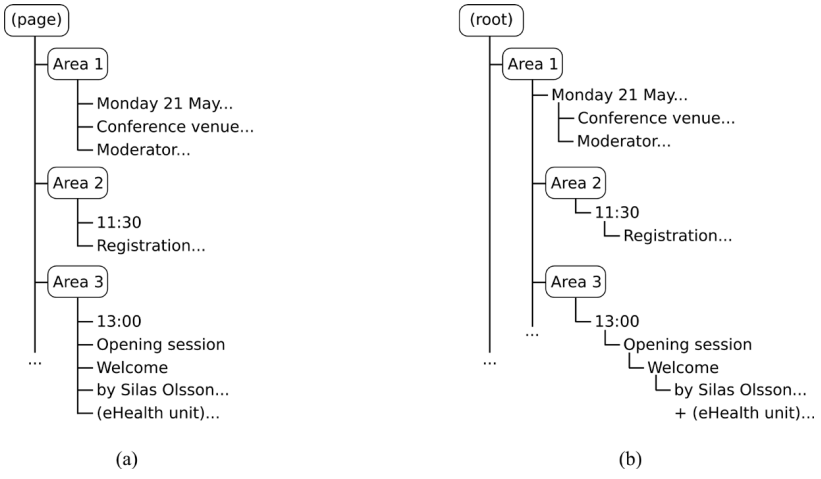
**Definition 2 (Content Block):** A content block is a visually separated rectangular area in the page detected by the page segmentation algorithm. A content block may contain either other detected content blocks or directly the content elements placed inside of the given rectangular area of the page. Therefore, we may speak about a hierarchy of content blocks in the page.

The right tree shows the LRM. It consists of the same nodes representing the content blocks and elements. However, the parent – child relationships between the nodes correspond to their visually presented logical relationships. More specifically, the parent – child relationship in the LRM means that the child node is visually presented to be *logically subordinate* to the parent node.

**Definition 3 (Logical Subordination):** Let $b_1$ and $b_2$ be two content blocks detected in the page. We say that $b_2$ is logically subordinate to $b_1$ if the content of $b_2$ elaborates or concertizes the content of $b_1$.

The most common examples of logical subordination relationships in documents are for example title – subtitle – paragraph, term

*Figure 3. The layout and the logical structure of a conference programme*



– definition, label – value, etc. In our example LRM in Figure 3, *Area 1* is visually interpreted as the heading of the programme. Therefore, all the programme items are represented as child nodes of *Area 1* in the tree. Similarly, in each item of the programme (*Area 2* and *Area 3*), the time is used as a label introducing all the remaining information. We say that *Area 2* and *Area 3* are *logically subordinate* to *Area 1*. Similarly, in *Area 3*, we can see a deeper structure because two more content elements are present that may be interpreted as sub-labels: the session title and subtitle.

The layout structure is generally obtained as a product of page segmentation. The purpose of the segmentation is to detect the visually separated content blocks in the page and represent their nesting. The LRM is given by the expected reader's interpretation of the individual content elements. In our approach, the logical subordination relationships mentioned above are obtained from the analysis of the visual presentation only. That means, we analyze whether the content blocks are *presented to be logically subordinate* and we do not analyze their exact semantics. From this point of view, the LRM remains domain-independent.

The details of the visual presentation analysis are provided in section "Overview of the Visual Analysis Approach". Further, we also provide the exact definitions of the processed layout model and the resulting LRM.

## Expected Applications of the LRM

Creating a domain-independent formal model of the visually presented relationships in the document is motivated by the necessity of processing large sets of heterogeneous documents mainly for the following tasks:

- **Logical Structure Detection:** The visually presented relationships are essential for identifying the internal structure of documents that means the headings, sub-headings, labels, etc. The generic model of these relationships may provide the information necessary for both the rule-based (Klink et al., 2000) and the machine learning-based (Luong et al., 2010) methods.
- **Information Extraction:** Explicitly described relationships in the document using LRM are suitable for extracting mainly structured data records. Then, it is possible to match the expected record structure with the structure presented in the page and to identify the content elements that correspond to the individual record fields as proposed for example in (Burget, 2004).
- **Information Retrieval.** The LRM may be also used for weighting the individual parts of the document during their indexing and retrieval. This may improve the retrieval results as shown for example in (Yu et al., 2002).

We provide more details about the expected use of the LRM in the mentioned applications in the final part of this article.

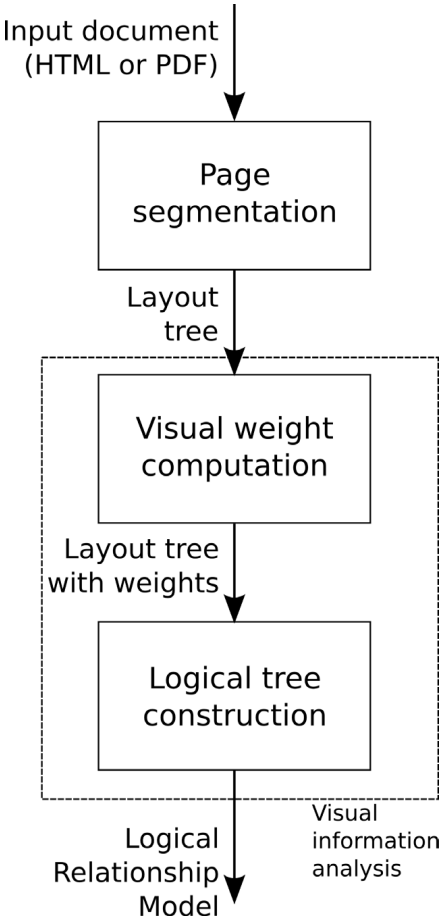
# OVERVIEW OF THE VISUAL ANALYSIS APPROACH

A general overview of our approach is shown in Figure 4. First, the layout structure is obtained using a page segmentation algorithm and represented as a hierarchical *layout tree*. We provide the details of this process in the following section. Subsequently, this tree is further processed in several steps in order to build the LRM.

First, we assume that the logical relationships between the content elements are presented to the reader using some commonly used visual means that the reader is expected to interpret properly. These visual means include for example using different font sizes, colors or indentation. We use the visual features of the contents for assigning different weights to the individual content elements and we construct a basic *logical tree* based in these weights. The details of this process are provided further in this article.

Regarding the processed document formats, we consider HTML documents including

*Figure 4. An overview of the LRM construction process*




the style information expressed by Cascading Style Sheets and PDF documents that are also frequently used on the web. These formats are supported by our implementation of the proposed method and were used for evaluation. However, the method is applicable to any document format that includes the textual information together with its visual presentation such as various popular office document formats.

## LAYOUT STRUCTURE DISCOVERY USING PAGE SEGMENTATION

The purpose of page segmentation is to discover the individual content blocks in the page and to create the layout tree. Many segmentation algorithms have been proposed for processing the web documents as for example VIPS (Cai et al., 2003). For our purpose we use the segmentation algorithm that we have published in Burget and Burgetová (2011) because it is applicable to any of the above mentioned document formats. However the further processing steps do not depend on the actual page segmentation method used.

The result of page segmentation is a tree model of the document layout structure. It can be defined as a tree:

$$L = (V_L, E_L) \quad (1)$$

where $V_L$ is an ordered set of the tree nodes and $E_L$ is a set of edges. The nodes in $V_L$ correspond to the individual visual blocks detected in the page or directly to the individual content elements. $E_L$ is a set of two-tuples that represent the visual nesting:

$$\forall v_i, v_j \in V_L : (v_i, v_j) \in \mathrm{E_L} \Leftrightarrow v_\mathrm{j} \text{ is directly nested in } v_i \quad (2)$$

This definition corresponds to Figure 3 (a) where the nodes correspond to the content elements (in case of leaf nodes) or content blocks (the remaining nodes). In case of more complex page layout, the tree depth is typically higher.

The segmentation usually defines an ordering of the tree nodes. For our segmentation algorithm, the node order corresponds to the order in which the individual content elements appear in the HTML or PDF code of the analyzed document (so called *document order*):

$$\forall v_i, v_j \in V_L : \mathrm{v_i} < \mathrm{v_j} \Leftrightarrow v_i \text{ precedes } v_j \text{ in the document code} \quad (3)$$

Based on this order, we can define the order of the child nodes of every non-leaf node from $E_L$.

## CONSTRUCTION OF THE LOGICAL RELATIONSHIP MODEL

The aim of the document authors is to make the logical structure apparent to the reader. Therefore, we analyze the common visual means used for presenting the logical relationships between the content elements and subsequently, we use the results of this analysis for constructing the LRM.

### Presentation of the Logical Structure in Documents

Logical structure of a document is presented to the reader according to existing cultural norms of document organization (Shreve, 2006). These norms represent the usual way of presenting the individual parts of the contents and their inter-relationships. For obtaining the basic awareness of the document structure, the following attributes of the content are important to a human reader:

- **Visual Properties of the Text:** The used font size, boldness, underlining or colors are often used for expressing the importance and even purpose of the individual content elements. For example, headings are usually clearly recognizable by their font size; often, there are even several levels of the headings distinguished in the document. Similarly, using a bold typeface or a different color indicates the importance of the element.
- **Content Layout:** Very often, the logical relationships between the content elements are indicated by the mutual positions of the elements. The most frequent means used in this category include indentation that is commonly used for presenting the elements subordinate to another element or columns that group the related parts of the content together. Tables can be also viewed as a special case of the content layout with a defined meaning given by the relationships between the table header and the subordinate table cells.

In order to interpret the visual and layout properties, we assign weights to the individual nodes of the layout model as described in the following section. As the next step, the LRM is constructed based on these weights.

## Assigning Weights to Layout Model Nodes

The visual and layout properties mentioned in the previous section allow the user to distinguish the level of logical superiority or subordination of the individual content blocks and elements. We express this level by assigning a *weight* to every node of the layout model defined in (1). A greater weight means that the corresponding content element appears to be more important in the document. For example, the main heading of a document should have the greatest weight, the sub-headings or smaller labels should have lower weights and plain document text that cannot be interpreted as a heading or label should have the lowest weight.

The weight is computed based on various visual properties of the text and the layout. Based on an analysis of real documents available on the web such as newspaper articles, conference programmes, timetables or even menus of the day, we have manually chosen a set of properties that are commonly used for indicating the logical superiority of the text. For each content element $v_c \in V_L$ (i.e. the leaf node of the layout tree), we compute the values of these properties. Similarly, for each non-leaf node (a content block) $v_n \in V_L$, the value of the same property is computed as an average of the property values of its child nodes.

For any layout node $v \in V_L$, we compute the values of the following properties:

- **Font Size:** Font size *fsize*(*v*) is computed as a value relative to the average font size of the whole document. That means, for the visual nodes with greater font size than the average the resulting value is *fsize*(*v*) > 1; for example, *fsize*(*v*) = 2.0 means that the font used in *v* is twice as large as the average font size of the document. Similarly, *fsize*(*v*) < 1 means that the font size in *v* is smaller than the document average.
- **Font Boldness:** The font boldness *fbold*(*v*) is equal to 1 for the nodes that use bold font only, *fbold*(*v*) = 0 means that the node does not contain any bold text.
- **Colors:** During the analysis, we find all the text colors used in the document and for each used color, we compute the percentage of the text of the given color in the whole text content. The value *cperc*(*v*) = 1 means that the whole document content uses the same color as the node *v*. Small values of *cperc*(*v*) mean that the color is quite rare in the document and therefore, it may indicate greater importance of *v*. During the color analysis, we apply a quantization to the color channels (we use four bits per red, green and blue channel) so that very similar colors are considered as a single color.
- **Indentation:** The subordinate content elements are often indicated by their indentation. For each content block, we consider up to four levels of indentation based on the comparison of the mutual positions its child elements. The level 0 (not indented) child elements obtain *indent*(*v*) = 1, the elements indented by one step obtain *indent*(*v*) = 0.75, etc. Finally the elements indented by more than four steps obtain *indent*(*v*)=0.
- **Centering:** We detect the child elements that are horizontally centered within their parent block by analyzing their position within the parent block and by comparing it with the positions of the preceding and following siblings. Centered elements have *center*(*v*) = 1, the remaining elements have *center*(*v*) = 0. When the element is centered, its indentation is not analyzed and we automatically consider *indent*(*v*) = 1.

Each of the above mentioned visual properties has different importance for computing the resulting weight. For example, the font size is the most important: a text written in larger font size is always interpreted to be superior to the text of a smaller font size. The relationships between the elements of the same size can be further distinguished by indentation etc.

In order to determine the actual importance of the individual properties computed above, we have analyzed a large set of web documents from the areas mentioned above and

we have manually investigated how the content structure is presented i.e. what are the influences of the computed visual feature values (such as the font size, color, etc.) to the expected resulting weight of the layout nodes. As a result, we have obtained the following formula for computing the weight of a layout node $v \in V_L$:

$$\begin{aligned} weight(v) &= 1000.0 \cdot fsize(v) + 2.0 \cdot fbold(v) \\ &+ 0.5 \cdot (1 - cperc(v)) \\ &+ 5.0 \cdot indent(v) + center(v) \end{aligned} \quad (4)$$

This formula puts weights to the individual computed features according to their observed importance. The weight on font size is very high because a very small change in the relative font size has a great impact to the weight of the node. For example, with a 12pt average font size, 18pt text (i.e. *fsize* = 1.5) is usually the most important header that is recognizable independently on its remaining features.

The resulting formula and the chosen set of analyzed presentation features reflect the presentation styles common in western tradition. As we show later in the evaluation section, it is generally applicable to a wide set of documents.

## Logical Relationship Model Construction

The definition of the logical structure is similar to the definition of the layout structure (1). It is also defined as a tree:

$$S = (V_s, E_s) \quad (5)$$

Where $V_S$ is a set of tree nodes; $V_S = V_L$ as defined in (1). $E_S$ is a set of tree edges that represent the logical relationships between the tree nodes. It is constructed using algorithm 6.3.

This algorithm recursively goes through the layout tree $L$. The functions firstChild(), nextChild() and parent() are used for obtaining the child and parent nodes in $L$. For each node $v_n \in V_L$, we compare the weights of its child nodes and we try to find the most appropriate parent node for each child node. Then, we add the appropriate two-tuples $(v_i, v_j)$ to the resulting set $E_S$. For each $(v_i, v_j) \in E_S$ it must hold that $weight(v_i) > weight(v_j)$ and we try to find the closest parent element where this condition is met.

As the result of this last step, we obtain a domain-independent model of the logical relationships as they come from the interpretation of the visual presentation of the content.

## LRM REFINEMENT BY ADDING DOMAIN KNOWLEDGE

In some cases, using additional domain knowledge is necessary in order to interpret the logical relationships correctly. The most important problem is to detect the content elements in the layout model that form a single logical entity. In the first conference programme shown in Figure 1, the names and affiliations of the authors are often presented in two lines, that means two separate content elements according to Definition 1 as shown in Figure 3 (a). However, from a logical point of view, they form a single logical element as shown in Figure 3 (b). This is particularly important for headings that span for multiple lines and which need to be represented as a single node in the logical structure in order to be able to assign the child nodes appropriately.

The basic problem here is to decide whether two or more neighboring visual blocks form a single information entity (e.g. a title split to several pages) or separate entities with different meaning (e.g. a title and author). Partially, this problem may be solved in the page segmentation phase by merging the neighboring content elements having the same visual style. Then, we obtain larger, visually consistent content blocks from the page segmentation as we have presented in (Burget and Burgetová, 2011). However, in some cases, the presentation style

*Algorithm 1. Logical relationship tree creation*

createLogicalStructure( $v_n \in V_L$ )
  $v_{cur}$ = firstChild( $v_n$ )
    while $v_n$ has more child nodes
      $v_{next}$ = nextChild( $v_n$ )
      createLogicalStructure( $v_{next}$ )
      $v_{par}$ = parent( $v_{cur}$ )
      $\Delta_{cur} = |weight(v_{next}) - weight(v_{cur})|$
      $\Delta_{par} = |weight(v_{next}) - weight(v_{par})|$
      if $\Delta_{cur} \leq \Delta_{par}$
        if $weight(v_{next}) < weight(v_{cur})$
          add $(v_{cur}, v_{next})$ to $E_S$
        else
          add $(v_{par}, v_{next})$ to $E_S$
        end if
      else
        $v_{anc}$ = closest ancestor of $v_{cur}$ in $L$ where
          $weight(v_{anc}) > weight(v_{cur})$
        add $(v_{anc}, v_{next})$ to $E_S$
      end if
      $v_{cur} = v_{next}$
    end while

of two different elements may be equal and the reader is expected to recognize the different nature of the presented information which is often domain-dependent (e.g. the personal names of the article authors). In other words, the reader is expected to recognize and distinguish the most common types of information from the given domain such as dates, times, personal names, places, etc. In such cases, it is not sufficient to analyze the visual presentation; an additional knowledge about the text content itself must be included into the LRM.

As a generic way of representing the additional information about the text, we propose adding *tags* to the individual nodes of the LRM that represent the domain knowledge about the content. Based on these tags, we may recognize the groups of nodes that possibly form a single information entity and we may refine the LRM. The whole process then consists of two phases:

- **Logical Tree Tagging:** To each node $v \in V_s$, a set of tags is assigned. These tags indicate the nature of their content as for example time, authors, locations, etc. They are domain dependent and they are assigned based on the analysis of the text for example using regular expressions or by employing a NER classifier. The tags and the way of their assignment must be chosen according to the application domain. The details for our testing domain of conference programmes are provided in the evaluation section. For other domains, a different set of tags should be used.
- **Tree Node Merging:** We go through the tree $S$ and for each subsequent nodes $v_i, v_j \in V_S$ that share the same parent node, we compare the tag sets assigned to these nodes. If the intersection of the tag sets is not empty, we replace these nodes with a single node that contains the text content and the child elements of the two nodes. Similarly, a longer sequence of nodes may be reduced to a single one.

The tagging itself may seem to be closely related to logical structure detection or information extraction tasks where the meaning of some content parts is also analyzed. However, in our approach, the tagging is only used for a quick and approximate estimation of the element purpose in order to refine the LRM. As it comes from the above descriptions, only the tags of the neighboring areas are compared and therefore, we do not require a great tagging accuracy across the whole model.

For example, a simple regular expression is sufficient for tagging the speech titles in conference programmes because its purpose is not to discover all the speech titles accurately but to mark the content elements that "might look as a speech title" at first glance. For a precise information extraction, a more thorough analysis of the LRM must be implemented as we discuss further in the application section.

After the refinement process, the resulting LRM should correspond to the reader's interpretation of the hierarchical logical relationships in the document.

## METHOD EVALUATION

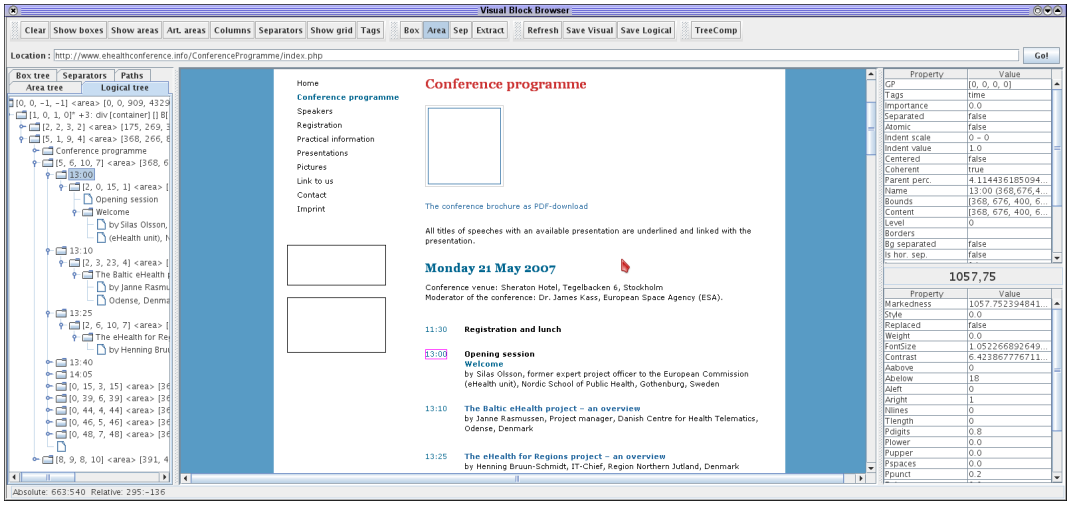
We have implemented the proposed method of the LRM construction in Java. For the page segmentation we have used our implementation of the segmentation algorithm published in (Burget and Burgetová, 2011). This implementation is based on the CSSBox[1] rendering engine that is able to render both the HTML and PDF documents. The resulting tool has both the graphical and web-based user interfaces that allow investigating the obtained layout structure visually and the LRM and their relationship to the actual page presentation (see Figure 5 and 6). The aim of the evaluation is to show, that the LRM may be used as a presentation-independent model of documents, i.e. that the structure of the obtained models is comparable for the documents from the same domain independently on how the document format and the presentation style.

As the testing domain, we have chosen the conference programmes. We have analyzed 68 different conference programmes from different areas (computer science, health, etc.) manually downloaded from the web. Each of the programmes uses a different way of the visual presentation of the given information. There were 7 PDF documents and 61 HTML pages in the analyzed set.[2]

For assigning the labels that represent the additional domain knowledge about the content, we have used the Stanford NER classifier (Finkel et al., 2005) for recognizing authors (personal names) and locations (that are recognized but not used for evaluation). For recognizing the content elements containing the remaining data (*date*, *title*, etc.), we have used regular expressions.

For the evaluation, we have expected that each programme contains the information about the individual speeches. We have focused on the speech *date*, *time*, *title* and the *names* of the authors or presenters. We have processed all the documents with the described algorithms in order to obtain their LRMs and we have

*Figure 5. An interactive application used for evaluating the obtained logical relationship models*




manually investigated whether the expected relationships shown in Figure 2 have been successfully detected and represented in the model. We have checked the following three relationships:

- **Date – Time:** Typically, there are multiple presentations or sessions taking place in the same day and this is usually taken into account in the presentation of the programme. The content element containing the date should be an ancestor node of the element containing the time of the same presentation. In some cases, the date is not explicitly presented in the programme (especially in case of one-day workshops), i.e. this relationship does not have to be present in the document.
- **Time – Title:** In the programmes the time is usually assigned to the individual speeches or to a whole session (one time element shared by multiple speeches). In both cases the content element containing the time should be an ancestor of the element containing the speech title.
- **Names (Time – Names or Title – Names)**: The content elements containing the author names and the speech title should share the same *time* ancestor element. In case the Time – Title relationship was not properly

*Figure 6. Structured record extraction using LRM and tree matching*

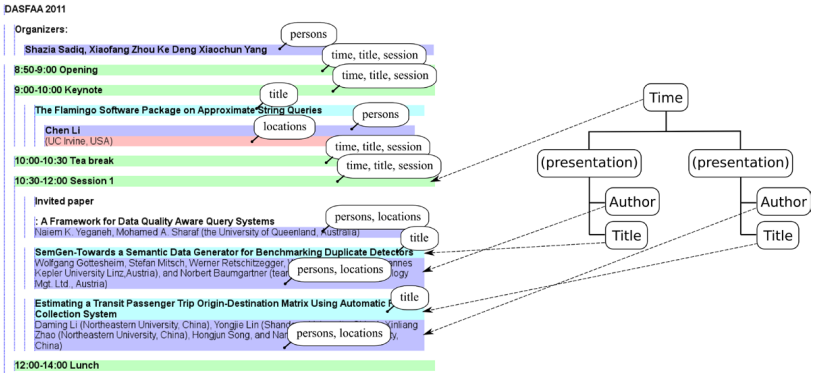


*Table 1. Results of the relationship identification in conference programmes*

| Relationship | Expected | Discovered | Ratio |
|---|---|---|---|
| Date – Time | 44 | 35 | 0.80 |
| Time – Title | 68 | 56 | 0.82 |
| Names | 68 | 64 | 0.94 |
| Total | 180 | 155 | **0.86** |

detected for some reason, it would not be possible to evaluate this relationship. In that case, we have checked whether the author names can be uniquely assigned to the correct speech in the obtained LRM, that means there exists a parent-child relationship between the *title* and *authors* elements of the same speech or they both share a unique ancestor element marked *(presentation)* in Figure 2.

We expect that all the relationships are presented consistently in a single programme. Therefore, we have considered the LRM structure to be correct, if the relationships are present in the LRM for at least 90% of the speeches presented in the programme. This allows a few speeches presented in an unexpected way (e.g. special highlighting, etc.) The results of the evaluation are shown in Table 1. We have expected the Date – Time and Time – Title relationships to be present in all 68 documents; the date was contained in 44 documents. From the total 180 relationships, 155 were correctly detected that gives the overall success ratio of 86%.

As for the incorrectly processed documents, the most common problem is a more complex way of data presentation than expected – typically a table with a more complicated (two-dimensional) structure that is not analyzed properly. In a few cases, there was an insufficient weight assigned to the *time* content blocks which caused the time to be represented as a sibling of the title and authors instead of the ancestor. This indicates that the evaluation of the presentational cues could be further improved.

## LOGICAL RELATIONSHIP MODEL IN APPLICATIONS

The motivation for creating the LRM was to provide a general document model for the tasks mentioned below. These tasks are usually solved separately; however, from the point of the LRM usage, they overlap significantly.

### Logical Structure Detection

When discovering the logical document structure, the task is usually to recognize the important parts of the documents such as headings, labels, publication dates or authors and to model their relationships. Existing approaches are based either on pre-defined rules (Klink et al., 2000) or machine learning methods (for example, Conditional Random Fields approach is used in Luong et al. (2010)). In both cases, LRM may provide an important information regarding the visually presented relationships between the individual content parts that is not directly available in the document. This information may be used for the rule construction (e.g. the author must be subordinate to an article heading) or for an automatic classification. This approach may be combined with further classification of the elements based on different criteria (such as font properties) as proposed for example in Burget and Burgetová (2011).

### Information Extraction

Information extraction presents the most important expected application of the LRM. With the hierarchical LRM, the information extraction problem may be viewed as a generalization of the logical structure detection. While in

the logical structure detection, the task is to identify the headings and labels, in information extraction, the task is to identify generally any information contained in the document such as for example names, presentation titles, personal data, etc. The visually expressed relationships may provide an important cue for identifying the pieces of data that form a single extracted record.

Since the LRM represents an explicitly and formally described structure of a document, it can be used for identifying a particular information element in a way similar to the above logical structure detection task. When an approximate content element tagging is applied to the LRM, the logical relationships represented in the LRM may be used for the disambiguation of the meaning of the individual content elements. For this, the tree matching methods may be used that compare the individual LRM subtrees with the expected structure of the information to be extracted in order to identify complete data records. Based on this comparison, we may decide which relationships are (not) acceptable in the extracted records and we may choose the most probable candidate elements that contain particular information.

This approach is demonstrated in Figure 6. The left part shows the output LRM tree obtained from a real conference programme using our implemented tool and visualized using a web interface. The different background colors and the annotations show the tags that have been assigned to the individual LRM nodes during the refinement phase. We may note that the tagging gives only approximate information in this stage. For example, an erroneous speech title (invited paper) starting with a colon has not been detected properly. Further, it is usually difficult to distinguish the session titles and the speech title without some additional heuristics or deeper natural language analysis. Therefore, the session titles have both the title (the speech title) and session (the session title) tags meaning that the particular node may be potentially interpreted in both ways. In addition, the same nodes have the time tag assigned because they also contain the time information in this case.

The right part of the figure shows the expected data record structure. Using a tree matching algorithm as proposed for example in Burget (2004), we search for the best matching subtrees in the LRM. Using the approximate matching that allows some missing or overlapping tags, we may resolve the above mentioned duplicities and missing tags. Compared to the previous work (Burget, 2004), LRM provides a more generic, formalized and robust model of the document structure while the information extraction process remains very similar. Our preliminary experiments with the tree matching algorithms show that the combination of the logical relationship model and the text analysis including the NER tagging gives very promising results in the extraction of structured records from documents.

Similarly, in You et al. (2013), the discovered hierarchical relationships are used as an input for a Tree-structured Conditional Random Fields classifier (Tang et al., 2006). Again, our proposed LRM may be used to formalize this approach in a similar way.

## Information Retrieval

Using the LRM for information retrieval tasks allows considering the document structure in the document indexing and retrieval. Thus, this task overlaps with the logical structure detection as well. For example in Yu et al. (2002), page segmentation is used to recognize the important parts of the page to be indexed. With the LRM, we may assign weights to the individual content elements based on their position in the LRM tree (for example based on the distance from root node or from some recognized important node such as main heading). This may be used for distinguishing the relevance of the individual parts of the document contents. During the document retrieval, the LRM may be used for answering structured queries (e.g. retrieving documents containing certain keywords in a particular hierarchical relationship) or for evaluating the similarity of document structure.

## CONCLUSION

In this paper, we have proposed a format and presentation-independent document model (LRM) that represents the visually presented relationships in the document contents. We have also presented a method of extracting the logical relationship model from HTML and PDF documents based on the interpretation of their visual presentation. The obtained model may be further refined using a domain-dependent text content analysis including a NER classification.

In order to evaluate the proposed LRM construction methods and to show the generality of the model, we have created a testing set of conference programmes in various formats and with very different presentation styles and we have proposed evaluation criteria regarding the discovered resulting structure. We have also implemented an interactive tool that allows investigating the obtained structure and comparing it with the actual presentation in the document.

The obtained results show that the method gives comparable results for documents from different sources that use different formats and visual presentation. This demonstrates that the Logical Relationships Model may be used as a presentation-independent model for further document processing such as logical structure discovery, information retrieval or extraction.

Finally, we have suggested a way of using the LRM in the most important web document processing tasks with a focus to information extraction based on tree matching algorithms. Our preliminary experiments show that the explicitly modeled content element relationships represented by the LRM may be very useful mainly for extracting structured records from the documents.

## ACKNOWLEDGEMENT

This work was supported by the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070. The research leading to these results has received funding from the European Community's 7th Framework Programme FP7/2007-2013 under grant agreement number 270001 – Decipher.

## REFERENCES

Burget, R. (2004). Hierarchies in HTML documents: Linking text to concepts. In *Proceedings of the 15th International Workshop on Database and Expert Systems Applications* (pp. 186–190). IEEE Computer Society.

Burget, R., & Burgetová, I. (2011). Automatic annotation of online articles based on visual feature classification. *International Journal of Intelligent Information and Database System*, *5*(4), 338–360. doi:10.1504/IJIIDS.2011.041322

Cai, D., Yu, S., Wen, J.-R., & Ma, W.-Y. (2003). *VIPS: A vision-based page segmentation algorithm*. Microsoft Research.

Finkel, J. R., Grenager, T., & Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05)* (pp. 363–370).

Klink, S., Dengel, A., & Kieninger, T. (2000). Document structure analysis based on layout and textual features. In *Proc. of International Workshop on Document Analysis Systems*, Brazil (pp. 99–111). IAPR.

Luo, X., Xu, Z., Li, Q., Hu, Q., Yu, J., & Tang, X. (2009). Generation of similarity knowledge flow for intelligent browsing based on semantic link networks. *Concurrency and Computation*, *21*(16), 2018–2032. doi:10.1002/cpe.1460

Luo, X., Xu, Z., Yu, J., & Chen, X. (2011). Building association link network for semantic link on web resources. *Automation Science and Engineering. IEEE Transactions on*, *8*(3), 482–494.

Luong, M.-T., Nguyen, T. D., & Kan, M.-Y. (2010). Logical structure recovery in scholarly articles with rich document features. *International Journal of Digital Library Systems*, *1*(4), 1–23. doi:10.4018/jdls.2010100101

Namboodiri, A., & Jain, A. (2007). Document structure and layout analysis. In B. B. Chaudhuri (Ed.), *Digital document processing, Advances in pattern recognition* (pp. 29–48). Springer London.

Nojoumian, M., & Lethbridge, T. C. (2007). Extracting document structure to facilitate a knowledge base creation for the uml superstructure specification. In *Proceedings of the International Conference on Information Technology* (pp. 393–400).

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The pagerank citation ranking: Bringing order to the web*. Technical Report 1999-66, Stanford InfoLab.

Rauf, R., Antkiewicz, M., & Czarnecki, K. (2011). Logical structure extraction from software requirements documents. In *Proceedings of the Requirements Engineering Conference (RE), 2011 19th IEEE International* (pp. 101–110).

Shreve, G. M. (2006). Corpus enhancement and computer-assisted localization and translation. In K. J. Dunne (Ed.), *Perspectives on localization* (pp. 309–332). Amsterdam, Philadelphia: John Benjamins Publishing Company.

Stoffel, A., Spretke, D., Kinnemann, H., & Keim, D. A. (2010). Enhancing document structure analysis using visual analytics. In *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10)* (pp. 8–12). New York, NY: ACM.

Summers, K. (1995). Toward a taxonomy of logical document structures. In *Electronic Publishing and the Information Superhighway: Proceedings of the Dartmouth Institute for Advanced Graduate Studies* (pp. 124–133).

Tang, J., Hong, M., Li, J., & Liang, B. (2006). Tree-structured conditional random fields for semantic annotation. In Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., … Aroyo, L. (Eds.), The semantic web - ISWC 2006 (Vol. 4273 of Lecture Notes in Computer Science, pp. 640–653). Springer Berlin Heidelberg.

Yashiro, H., Murakami, T., Shima, Y., Nakano, Y., & Fujisawa, H. (1989). A new method of document structure extraction using generic layout knowledge. In *Proceedings of the International Workshop on Industrial Applications of Machine Intelligence and Vision*, Tokyo, Japan (pp. 282–287).

You, Y., Xu, G., Cao, J., Zhang, Y., & Huang, G. (2013). Leveraging visual features and hierarchical dependencies for conference information extraction. In Ishikawa, Y., Li, J., Wang, W., Zhang, R., & Zhang, W. (Eds.), Web technologies and applications (Vol. 7808 of Lecture Notes in Computer Science, pp. 404–416). Springer Berlin Heidelberg.

Yu, S., Cai, D., Wen, J.-R., & Ma, W.-Y. (2002). *Improving pseudo-relevance feedback in web information retrieval using web page segmentation*. Microsoft Research.

## ENDNOTES

1 http://cssbox.sourceforge.net

2 A complete table with the programme URLs and their evaluation and the copies of the documents are available at http://www.fit.vutbr.cz/~burgetr/publications/ijcini

*Radek Burget received his PhD in Information Technology in 2004 from the Brno University of Technology. He is an assistant professor at the Faculty of Information Technology, Brno University of Technology. His research interests include data mining methods, semi-structured data modeling, knowledge engineering and the semantic web.*

*Pavel Smrz is an associate professor and research project leader at the Faculty of Information Technology, Brno University of Technology. His research interests include knowledge technologies, large-scale information extraction from text and multimedia, and acceleration of machine-learning algorithms by means of parallel platforms. He authored more than 60 papers in scientific journals and conference proceedings.*

# Appendix C

# Extraction of Structured Records

## C.1 Information Extraction from Web Sources Based on Multi-aspect Content Analysis

Milička, M.; Burget, R.: Information Extraction from Web Sources based on Multi-aspect Content Analysis. In *Semantic Web Evaluation Challenges, SemWebEval 2015 at ESWC 2015*, *Communications in Computer and Information Science*, vol. 2015. Springer International Publishing. 2015. ISBN 978-3-319-25517-0. ISSN 1865-0929. pp. 81–92.

# Information Extraction from Web Sources Based on Multi-aspect Content Analysis

Martin Milicka( ) and Radek Burget

Faculty of Information Technology, IT4Innovations Centre of Excellence,
Brno University of Technology, Bozetechova 2, 612 66 Brno, Czech Republic
{imilicka,burgetr}@fit.vutbr.cz

**Abstract.** Information extraction from web pages is often recognized as a difficult task mainly due to the loose structure and insufficient semantic annotation of their HTML code. Since the web pages are primarily created for being viewed by human readers, their authors usually do not pay much attention to the structure and even validity of the HTML code itself. The CEUR Workshop Proceedings pages are a good illustration of this. Their code varies from an invalid HTML markup to fully valid and semantically annotated documents while preserving a kind of unified visual presentation of the contents. In this paper, as a contribution to the ESWC 2015 Semantic Publishing Challenge, we present an information extraction approach based on analyzing the rendered pages rather than their code. The documents are represented by an RDF-based model that allows to combine the results of different page analysis methods such as layout analysis and the visual and textual feature classification. This allows to specify a set of generic rules for extracting a particular information from the page independently on its code.

**Keywords:** Document modeling · Information extraction · Page segmentation · Content classification · Ontology · RDF

## 1 Introduction

The documents available on the web present a large and ever growing source of information. However, extracting information from the HTML documents remains a challenging tasks mainly because of the high variability of the markup, loose structure of the documents and very rare use of any kind of semantic annotations that could be used for recognizing a particular information in the document.

The research in this area includes many different approaches including a direct HTML code analysis by different methods [7,8], DOM analysis [6], page layout [2] or other visual feature analysis [10]. As the research results show, the web documents are too variable for the a simple and straightforward solution. The document processing cannot be based only on single aspect such as the text content, visual features or document structure because each approach is suitable


F. Gandon et al. (Eds.): SemWebEval 2015, CCIS 548, pp. 81–92, 2015.
DOI: 10.1007/978-3-319-25518-7_7

for a different kind of documents. Therefore, we propose an approach that can combine multiple aspects of the document.

The documents may be described on different levels of abstraction starting with the code through the rendered page layout and visual features of the contents to a logical structure as it is expected to be interpreted by a human reader. We propose an ontology-based document model that is able to capture all the mentioned kinds of information. For each level of the description, we use a specific ontology. The highest abstraction level represents the target domain of the extracted information.

In this paper, we apply this approach to the processing of the CEUR Workshop proceedings as a part of the ESWC 2015 Semantic Publishing Challenge. We employ a combination of algorithms such as page segmentation or content classification for building the proposed model from source documents. Based on a combination of different features, we propose the way of extracting the logical structure of the document. This structure is finally transformed to the specific domain ontology. This approach allows to abstract from the HTML implementation details and increase the robustness of the extraction.

## 2 System Architecture

The presented information extraction system is based on our recently developed FITLayout[1] framework [9] – a generic framework for web page segmentation and its further analysis. The complete architecture overview is shown in Fig. 1. Implementation details specific for the Semantic Publishing Challenge 2015 are described later in Sect. 4.

Unlike most existing information extraction systems, our system does not analyze the HTML or CSS code of the input documents directly. Instead, it operates on the rendered page trying to use the same information as the user who is actually browsing the page. This allows to abstract from the HTML-related problems such as irregular code structure, invalid markup, etc.

The individual documents (CEUR pages) are processed independently on each other. The processing consists of several steps. The results of each step are stored to an internal RDF repository; each step adds more information to the model of the processed document. First, source pages are rendered using a built-in rendering engine that provides the information about the layout and visual features of the individual pieces of the contents. Additionally, basic text analysis steps are applied on the document in order to recognize important entities in the text such as dates, times, capitalized sequences or personal names. Subsequently, the obtained model is analyzed and the desired information such as editors, paper titles, authors, etc. is recognized using a set of quite simple rules based on the actual presentation of the individual content parts. Based on the recognized parts of the contained information, we build a *logical structure* of the document that represents the semantic relationships. Finally, this structure is transformed to the resulting linked data set.

[1] http://www.fit.vutbr.cz/~burgetr/FITLayout/.

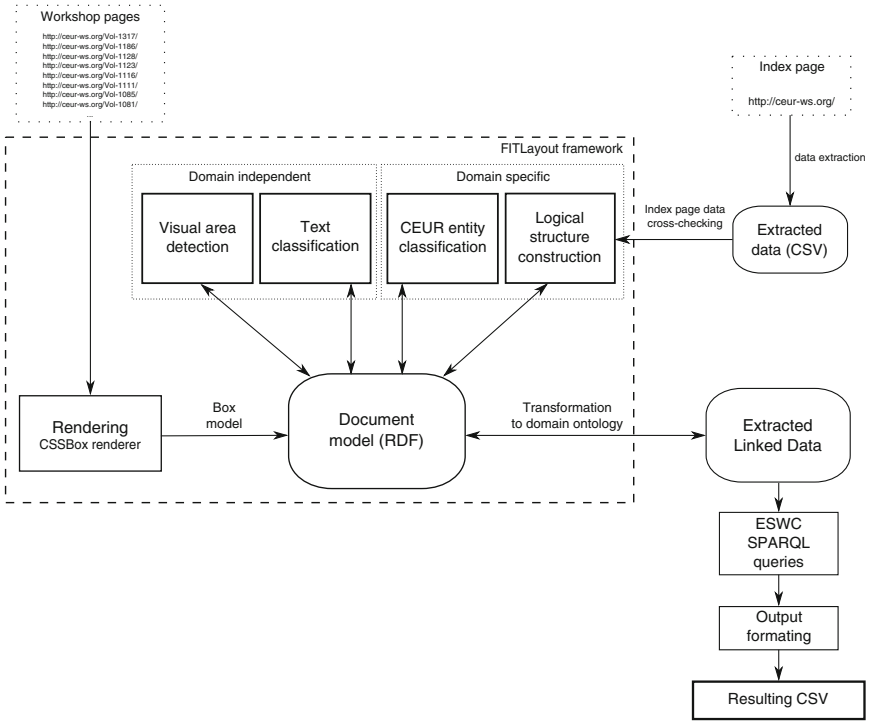



**Fig. 1.** Extraction system architecture

### 2.1 Page Rendering

The rendering engine processes the input HTML and the linked CSS files and produces the information about the content layout in the page. The layout is represented by a *box model* generally defined in the CSS specification [1]. This model describes the positions of the individual pieces of content in the resulting page and their further visual properties (fonts, colors, etc.) Each box corresponds to a rectangular area in the rendered page. The boxes are organized in a hierarchical structure called a *box tree* that roughly corresponds to the source DOM structure.

The obtained box tree is transformed to RDF data using the FITLayout box model ontology described in Sect. 3.1. In the subsequent steps of the model building, more information is added to the page model as the result of the individual analysis methods.

### 2.2 Model Building

The model building phase consists of four analysis steps. The first two of them are domain-independent; they are not specific for the SemPub2015 task. The other two steps are specific for the target domain. The details of the individual steps are described later in Sect. 4.

1. **Visual area detection.** We identify all the boxes in the box tree that are visually distinguishable in the resulting pages. These boxes form the basic

*visual areas*. We construct a tree of visual areas based on their visual nesting in the rendered page. The resulting area tree is described using the corresponding FITLayout segmentation ontology (see Sect. 3.2). Later, each area may be assigned any number of text *tags* that represent the expected meaning of the area at different levels.
2. **Text classification.** We go through the leaf areas of the visual area tree and we identify important generic entities in the text such as dates, times or personal numbers. Based on the discovered entities, we assign tags to the areas that indicate the type of their content.
3. **CEUR entity classification.** Based on the previous two steps, i.e. the layout model and the properties of the text, we identify the CEUR entities such as the workshop title, editor names, paper titles and authors, etc. Their discovery is based on mutual positions of the corresponding areas and regular patterns in the presentation styles. The areas that correspond to the individual CEUR entities are again marked by the appropriate tags. For example, a visual area that obtained a *persons* tag in the previous text classification step (i.e. it contains some personal names) is likely to obtain the *editors* or *authors* tag depending on where the area is placed within the page.
4. **Logical structure construction.** The purpose of the logical structure is to represent the relationships among the CEUR entities tagged in the previous steps. For example, the title, authors and page numbers that belong to a single paper, papers that belong to a single section, etc. In a domain-dependent way, we transform the tagged area tree to the logical structure tree where the logical nodes correspond to particular text strings (e.g. the names themselves) and the parent-child relationships correspond to the semantic subordination of the entities (e.g. the title, authors and pages are child nodes of a paper node). Each node is marked with a single tag that specifies its semantic.

The whole process corresponds to the transition from the rendered page model (the box tree) through the page layout model (the visual area tree) to its semantic interpretation (the logical area tree). In the next step, the resulting logical model can be transformed to the target domain ontology.

### 2.3 Output Dataset Generation

The resulting logical structure tree that is obtained from the model building phase and stored in the intrenal RDF repository contains the complete information extracted from the source page together with its structure. The output dataset generation only consists of transforming the data represented using the FITLayout internal visual area ontology to the target domain ontology described in Sect. 3.5. This is implemented as a single SPARQL query[2] on the internal RDF repository.

[2] https://github.com/FitLayout/ToolsEswc/blob/master/sparql/logicalTree2domain.sparql.

## 3 Ontological Model

The ontological model describes the processed document at multiple levels of abstraction. We have defined five abstraction levels of document description where each higher level adds specific knowledge to the previous one. Each level of description is characterized by its ontology. The hierarchy of levels is shown in Fig. 2. We can see that all the levels can be divided in two groups: domain-independent and domain-specific. The tagging level in the middle joins the two parts together.

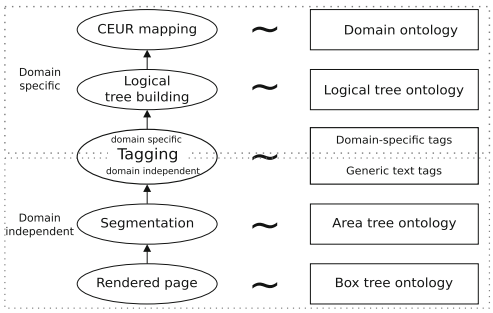



**Fig. 2.** Ontological model

### 3.1 Rendered Page Level

At the level of the rendered page, the ontology-based model corresponds to the document box model where its rendering is based on the source data presented in the HTML document and visual features defined by Cascading Style Sheets (CSS).

The schema of the presented *Box model ontology* is on Fig. 3(A). Every class is based on the *Rectangle* class which defines characteristic size, position and visual features. A *Box* denotes a base displayed document element. It follows the definition from the CSS formatting model [1]. The boxes may be nested, which creates a hierarchical structure similar to the Document Object Model (DOM). The *Page* class represents the whole rendered page. The *belongsTo* property denotes the relationship between the Page and some rectangular objects (boxes) that create the contents of the page. The *Box* can be further specialized into the *ContainerBox* or *ContentBox* classes where the *ContainerBox* may contain other boxes (allows nesting). The *ContentBox* represents a Box that contains a connections of content objects like images, textual information or common objects like Flash, video, etc.

### 3.2 Segmentation Level

Page segmentation generally detects the visually distinguished segments in the rendered page (we call them *visual areas* in this paper). There exist many page segmentation algorithms; one of the most popular ones is called VIPS [4].

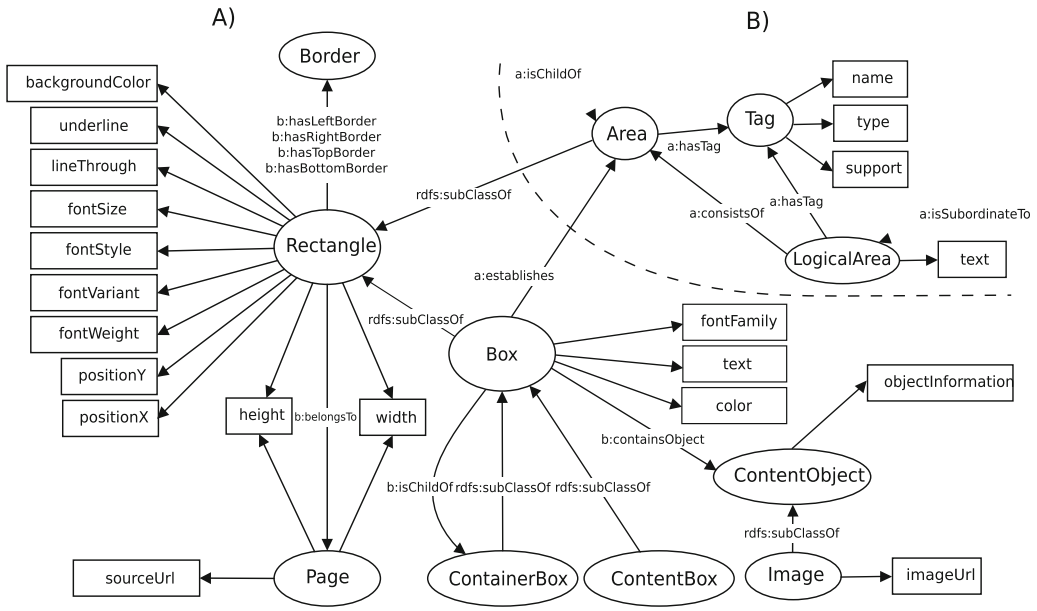



**Fig. 3.** (A) Box tree ontology (http://fitlayout.github.io/ontology/render.owl#) (B) Area based ontology (http://fitlayout.github.io/ontology/segmentation.owl#)

The segmentation model extends the Box model by a possibility of representing the visual areas. In the Fig. 3(B) we can see a part of segmentation ontology design. The basic *Area* class is defined as a specialization of the *Rectangle* class from the Box model ontology. It represents the visual areas detected during the page segmentation. A visual area is usually created by a collection of boxes contained in this visual segment. Visual areas may be nested and create a hierarchy based on their visual nesting similarly to boxes.

### 3.3 Tagging Level

The tags are represented by the *Tag* class (in Fig. 3(B)); multiple tags may be assigned to a single visual area. Each tag is represented by its *name* and *type* where the type represents a set of tags with the same purpose (e.g. the tags obtained from text classification) and the name corresponds to the actual tag value.

In Sect. 4, we give an overview of the tags used for the given domain. Some of them are domain-independent (Table 1), some are domain-dependent (Table 2).

### 3.4 Logical Tree Level

The logical structure represents the actual interpretation of the tagged visual areas in the target domain. Each logical area corresponds to a semantic entity identified as a text string contained in some visual areas (e.g. an author name). It is represented by the *LogicalArea* class in (Fig. 3). Each logical area has a single tag assigned that denotes its meaning in the target domain (e.g. a paper title).

The logical areas are organized to a hierarchical structure again (using the *isSubordinateTo* property). However, unlike the visual areas, where the hierarchy represents the visual nesting, for logical areas, the hierarchy corresponds to the logical relationships among the entities – e.g. a paper and its title or page numbers.

The resulting logical area tree provides a generic representation of the extracted information and its structure and it can be directly mapped to the target domain ontology.

### 3.5 Domain Level

The domain ontology defines the entities and their properties in the target domain. It is used for the resulting data set produced by our extraction tool. For the the CEUR proceedings domain, we use a combination of existing ontologies shown in Fig. 4 that is greatly inspired by [8] with some simplifications.

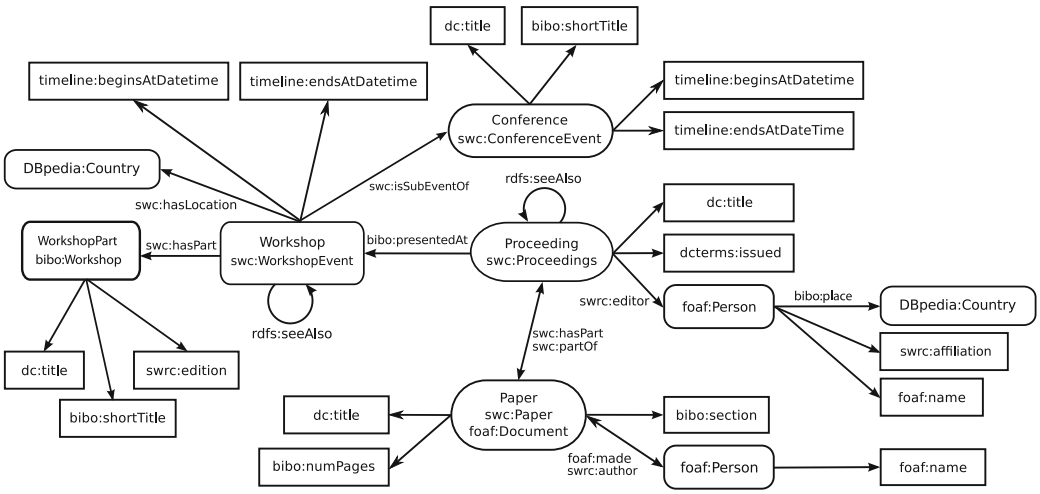



**Fig. 4.** Domain ontology – ESWC proceedings

## 4 System Implementation

The FITLayout framework used as a base for our system implements a variety of general tasks such as page rendering, page segmentation and text feature analysis. Moreover, it allows to implement custom extensions and add them to the page processing chain. For the purpose of the CEUR proceedings processing, we have implemented several domain-specific extensions that include the CEUR entity recognition and a custom logical structure builder specific for this particular task[3].

We made several experiments with using the microformats available in some of the CEUR volume pages for training a visual feature classifier that would

[3] https://github.com/FitLayout/ToolsEswc.

be later used for the remaining volumes. However, the presentation style of the individual volumes is quite variable in terms of the used fonts, layout or information ordering. Therefore, we have decided to process the individual pages independently. In the final version of our tools, we do not use any kind of classifier training (apart from the pre-trained Stanford NER classifier used for recognizing the personal names as described in Sect. 4.2). Instead of this, we just gather statistic about the frequently used presentation patterns and styles used in the currently processed page and we assume the most frequent one to be consistently used in the page as described in Sect. 4.3. The microformats are not used at all in the end because their availability is not guaranteed.

In the following sections, we explain the most important details of the whole information extraction process.

### 4.1 Layout Analysis

The FITLayout framework assumes a usage of a page segmentation method for the construction of the visual area tree. However, due to the relatively simple layout of the CEUR proceedings, we decided not to use a full-featured page segmentation algorithm. Instead, we just use a basic visual area recognition algorithm that corresponds to the initial step of our previously published page segmentation algorithm [3]. From the box tree obtained from rendering, we choose the boxes that are visually distinguishable in the page: they directly represent a piece of text or image content or they have some visible visual separator: a separating background color or a border around the box.

For the CEUR proceedings, the resulting layout model is usually very flat: Most of the content areas are directly the child nodes of the root node because there is usually no visual nesting used in the layout. The only exception is the title of some of the proceedings that is visually framed.

### 4.2 Generic Text Tagging

Area tagging is used to roughly classify the visual areas that contain certain kind of information. The FITLayout framework provides a set of general purpose taggers that assign tags of the `FitLayout.TextTag` type to the visual areas by a simple analysis of the contained text mainly using regular expressions. Table 1 describes the text tags we have used for the given task and the way of their assignment to the visual areas.

The used regular expressions are quite general (especially for the paper titles), and the used generic NER classifier is not 100 % accurate neither. Therefore, the tag assignment obtained in this phase provides just a rough and approximate classification of the areas. Further refining is performed in the following CEUR entity recognition phase.

### 4.3 CEUR Entity Recognition

The CEUR entity recognition consists of assigning another set of tags to the discovered visual areas. These tags correspond to the individual types of information

**Table 1.** Tags added during the text feature analysis (tag type `FitLayout.TextTag`)

| Tag | Meaning | Way of recognition |
|---|---|---|
| *dates* | A date in recognizable format | Regular expressions and specific keywords (months) |
| *pages* | Page span specification | Regular expression |
| *persons* | Personal names | Stanford NER classifier [5] |
| *title* | Paper title | Regular expression |

that we want to extract from the source document. The complete list of the assigned tags (with the type `ESWC`) and their meaning is in Table 2.

**Table 2.** Tags used for the CEUR entity annotation (tag type `ESWC`)

| Tag | Meaning |
|---|---|
| *vtitle* | Volume title |
| *vcountry* | Workshop location (country) |
| *veditor* | Editor name |
| *vdate* | Date(s) of the workshop |

| Tag | Meaning |
|---|---|
| *subtitle* | Volume subtitle (proceedings) |
| *title* | Paper title |
| *authors* | Paper author(s) |
| *pages* | Paper pages |

The transition from the general *text tags* listed in Table 1 to the *semantic tags* listed in Table 2 corresponds to the disambiguation and refining of the rough text classification. We assume that some text tags may be missing or may have been assigned incorrectly. Some tags are ambiguous, e.g. the *persons* tag may indicate author or editor names depending on context.

For assigning the semantic tags, our refining algorithms take into account the following aspects:

- *Common visual presentation rules* – there exist some commonly used rules for visual formatting of the presented information in a document. E.g. a title or subtitle is written in larger font or at least bolder than a normal text.
- *Regularity in presentation style* – we assume that all the information of the same meaning (e.g. all paper titles) is presented with the same visual style (fonts, colors, etc.) in a single proceedings page.
- *Regularity in layout* – some proceedings put author names before the paper title, some put them below or on the same line. However, this layout is again consistent through the whole proceedings page.
- *Locality of the information* – information of the same kind is presented in one area of the page. We can identify an area containing editors, papers, etc. The order of these area remains the same in all the proceedings pages.
- *Textual hints* – some key phrases such as “Edited by” or “Table of Contents” are commonly used in most of the proceedings. When they are found in the page, they can be used to refine the expected area where a particular information is located within the page.

Our algorithm works in the following steps:

1. We discover the position of the workshop title and the repeating layout and style patterns that (together with the assigned text tags from Table 1) correspond to the published papers and their authors and similarly for editors and their affiliations.
2. Based on the discovered patterns, we guess approximate areas in the rendered page that are likely to contain a particular information: the workshop title, subtitle (proceedings information), editors, papers and submission details. If the text hints such as "Edited by" are present in the pages, the expected area bounds are adjusted appropriately.
3. In these areas, we find the most frequent font style used for each type of information (e.g. author names) and the most frequent layout pattern (authors before or after the title, etc.) Then, we assign the appropriate semantic tags from Table 2 to all the visual areas using the same font style that correspond to the discovered layout pattern. This solves the possible inaccuracy of the text tag assignment.

The workshop title is discovered by its font size (it's always written with the largest font size used in the page). The editor area is guessed by searching personal names between the workshop title and the papers (the "Table of contents" text may be used as a hint when present) and the subtitle is located between the title and the editors.

As the result, we obtain a refined tagging of the visual areas that indicates their semantics.

### 4.4 Logical Structure Construction

The last logical structure construction phase has two main goals:

- Extract the text data from the tagged visual areas. The area may contain multiple kinds of information (e.g. several author names, several editors or some additional text that should be omitted).
- Put together the information that belongs to a single entity: the name and affiliation of a single editor or the title, authors and pages of a single paper.

These goals correspond to the construction of a tree of *logical areas* as defined in Sect. 3.4. The text extraction corresponds to the identification of the logical areas and the relationships among the areas (denoted using the *a:isChildOf* property) are used for creating a tree of logical areas where the child nodes specify the properties of its parent node.

We have implemented a custom logical tree builder that goes through the visual area tree and creates the logical areas organized in subtrees depending on the assigned semantic tags. For this, some more text processing is usually required: splitting the author area to several author names by separators, completing the editor affiliations by matching the different kinds of symbols and bullets and extracting the data such as workshop date from longer text lines.

The countries in the editor affiliations are recognized by a simple matching with a fixed list of countries and their DBPedia resource IRIs (a CSV extracted from DBPedia).

The workshop and conference acronym extraction is based on a simple text parser that recognizes all the acronyms and the ordinals in the text. In order to distinguish between the workshop and the conference acronyms, we try to locate the particular keywords (e.g. "colocated with") in the subtitle and we also compare the sets of acronyms found in the title and the subtitle since the conference acronym is very rarely present in the main title.

Some information such as the paper IRIs must be obtained from the underlying code from the `id` or `href` attributes. Therefore, in our stored rendered page model, we maintain the information about the source DOM nodes that produce the given box displayed in the page.

The resulting logical structure description is added to the FITLayout internal RDF repository and it can be directly transformed to the output linked data set by mapping to the target ontology.

### 4.5 CEUR Index Page Processing

The CEUR proceedings index page is a specific source of information. We use this page for locating the related workshops (the *see also*) information, the date of publication. We also use the volume title from the index page in the final output because the title in the individual pages is slightly different in some cases.

Since the index page is just a single HTML document with a specific style and quite a regular structure, we have just used a simple "old school" Unix `awk` script for extracting this data directly from the HTML code. This script produces a CSV output that is used by the logical tree builder to complete the logical structure.

## 5 Conclusions

In this paper, we have presented a web information extraction approach based on a complex modelling of different aspects of the processed document. Our system analyzes the rendered document and in multiple steps, it guesses and later disambiguates the semantics of the individual text parts by combining the page segmentation and text classification methods with specific extraction rules based on visual presentation of the content. This approach allows to avoid HTML-related implementation details. The extraction task is specified on quite a high level of abstraction that ensures the tolerance of the method to different variations in the processed documents.

**Acknowledgments.** This work was supported by the BUT FIT grant FIT-S-14-2299 and the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070.

## References

1. Bos, B., Lie, H.W., Lilley, C., Jacobs, I.: Cascading style sheets, level 2, CSS2 specification. The World Wide Web Consortium (1998)
2. Burget, R.: Layout based information extraction from HTML documents. In: ICDAR 2007, pp. 624–629. IEEE Computer Society (2007)
3. Burget, R., Rudolfová, I.: Web page element classification based on visual features. In: 1st Asian Conference on Intelligent Information and Database Systems ACIIDS 2009, pp. 67–72. IEEE Computer Society (2009)
4. Cai, D., Yu, S., Wen, J.R., Ma, W.Y.: VIPS: a Vision-based page segmentation algorithm. Microsoft Research (2003)
5. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL 2005, pp. 363–370 (2005)
6. Hong, J.L., Siew, E.G., Egerton, S.: Information extraction for search engines using fast heuristic techniques. Data Knowl. Eng. **69**(2), 169–196 (2010). http://dx.doi.org/10.1016/j.datak.2009.10.002
7. Hong, T.W., Clark, K.L.: Using grammatical inference to automate information extraction from the web. In: Siebes, A., De Raedt, L. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 216–227. Springer, Heidelberg (2001)
8. Kolchin, M., Kozlov, F.: A template-based information extraction from web sites with unstable markup. In: Presutti, V., et al. (eds.) SemWebEval 2014. CCIS, vol. 475, pp. 89–94. Springer, Heidelberg (2014). http://dx.doi.org/10.1007/978-3-319-12024-9_11
9. Milicka, M., Burget, R.: Multi-aspect document content analysis using ontological modelling. In: Proceedings of 9th Workshop on Intelligent and Knowledge Oriented Technologies (WIKT 2014), pp. 9–12. Vydavatelstvo STU (2014)
10. You, Y., Xu, G., Cao, J., Zhang, Y., Huang, G.: Leveraging visual features and hierarchical dependencies for conference information extraction. In: Ishikawa, Y., Li, J., Wang, W., Zhang, R., Zhang, W. (eds.) APWeb 2013. LNCS, vol. 7808, pp. 404–416. Springer, Heidelberg (2013). http://dx.doi.org/10.1007/978-3-642-37401-2_41

## C.2 Matching Visual Presentation Patterns

Burget, R.: Information Extraction from the Web by Matching Visual Presentation Patterns. In *Knowledge Graphs and Language Technology: ISWC 2016 International Workshops: KEKI and NLP&DBpedia*. Lecture Notes in Computer Science vol. 10579. Springer International Publishing. 2017. ISBN 978-3-319-68722-3. pp. 10–26. doi:10.1007/978-3-319-68723-0_2.

# Information Extraction from the Web by Matching Visual Presentation Patterns

Radek Burget( )

Faculty of Information Technology, Centre of Excellence IT4Innovations,
Brno University of Technology, Bozetechova 2, 612 66 Brno, Czech Republic
burgetr@fit.vutbr.cz

**Abstract.** The documents available in the World Wide Web contain large amounts of information presented in tables, lists or other visually regular structures. The published information is however usually not annotated explicitly or implicitly and its interpretation is left on a human reader. This makes the information extraction from web documents a challenging problem. Most existing approaches are based on a top-down approach that proceeds from the larger page regions to individual data records, which depends on different heuristics. We present an opposite bottom-up approach. We roughly identify the smallest data fields in the document and later, we refine this approximation by matching the discovered visual presentation patterns with the expected semantic structure of the extracted information. This approach allows to efficiently extract structured data from heterogeneous documents without any kind of additional annotations as we demonstrate experimentally on various application domains.

**Keywords:** Web data integration · Information extraction · Structured record extraction · Page segmentation · Content classification · Ontology mapping

## 1 Introduction

The World Wide Web contains a vast amount of documents containing data records presented in a regular, visually consistent way using different kinds of lists, tables or other logical structures. Typical examples include product data, events, exchange rates, sports results, timetables and many more. Although the structure of the presented information is generally predictable for every application domain, the actual data records may be presented in the HTML documents in countless ways.

For large and consistent data sources such as Wikipedia, it is possible to define extraction templates that may be reused for a great number of pages. However, for heterogeneous sources where every document may use different presentation patterns, this approach is not feasible. The great variability in presentation and almost no semantic annotations available in HTML documents


M. van Erp et al. (Eds.): ISWC 2016 Workshops, LNCS 10579, pp. 10–26, 2017.
https://doi.org/10.1007/978-3-319-68723-0_2

make the automatic integration of such web sources to structured datasets (such as DBPedia) a challenging problem.

In this paper, we present a method for the discovery and extraction of structured records in web documents. In contrast to most current approaches that perform a complex analysis of the document HTML code or its visual organization in order to detect repeating structures (top-down approach) [1,8,13,14], we use an opposite (bottom-up) approach: We start with the smallest consistent text elements and we match the visual relationships among these elements with the expected structure of the extracted records. This way, we are able to automatically discover the visual patterns used for presenting the data records in the given document.

The most important benefits of the presented approach are the following:

- The extraction task specification is based only on a generic domain knowledge consisting of the logical relationships among the individual data fields to be extracted and a very general specification of allowed values for each data field.
- No templates need to be used and no labels or annotations are required in the source documents.
- The method can be easily adapted for any target domain as it allows integration of arbitrary domain-specific knowledge (such as dictionaries or extracted data formats) and different data field recognition methods (from domain-specific heuristics to general NLP methods such as named entity recognition). We demonstrate the method application to different target domains in Sect. 9.
- The method is independent on the format of the input documents. We use the HTML and PDF documents as the most important information source but any other document type where the styled text is available may be used as well.

We also demonstrate that our information extraction method may be integrated with DBPedia in two ways: (1) DBPedia may be used for the recognition of candidate data fields in the extracted records and (2) the extracted records may contain new data that may be linked back to existing DBPedia resources. This allows integrating new web sources to DBPedia.

## 2 Related Work

Information extraction from web documents is a research area that is interesting for different applications. The most important application areas include extracting data results from query result pages [1,8,9,12–15] (obtained either from general search engines or specialized ones such as product search) or obtaining structured data buried in large sets of web documents [5,10].

When considering the recently published approaches, we may identify two basic groups from the perspective of the used representation of the input document: (1) code-based approaches that use a representation of the input document code (mainly DOM for HTML documents) [6,9,10,15] and (2) vision-based

approaches that use some kind of visual representation of the rendered page that may be obtained by adding some visual features to the document code model [1,8] or by using a standard page segmentation algorithm [13,14]. However, regardless of the used document representation, all the mentioned approaches expect HTML documents at the input.

Most existing methods are based on a top-down approach which is basically presentation-driven. After creating the document model as mentioned above, the model is usually preprocessed in order to filter the content blocks regarded as noise or to locate the most probable regions of interest (called a result section [12], data sections [14] or data region [8]). Then, the individual data records are identified based on the detection of repeating structures in the model by frequency measures [9] or visual pattern detection [1,12,14]. The structure of the extracted information is inferred from the discovered records while using additional information such as explicit labels present in the page [1,12,14,15] or even the query interface in case of the query result extraction [12,13]. This presentation-driven approach is suitable for many applications such as the deep web crawling. On the other hand, in case of information extraction from web sources for the semantic web, structured databases or particular applications, the structure of the extracted information is typically available in advance (for example as a domain ontology) and the task is to locate the corresponding data records in the input documents.

We have identified only a few approaches that are based on a previously known ontological model of the information being extracted. The classical work by Embley et al. [6] uses a conceptual domain model that defines the lexical and non-lexical classes and the relationships among them. However, before the conceptual model may be used for information extraction, a complex input document preprocessing is required that does not take the into account the domain model and it is based on heuristics tightly related to the HTML language constructions. Similarly, our earlier work [2] uses complex vision-based document preprocessing for creating a logical model of the processed document in a form that can be later matched with an ontology-based domain model.

Our approach we present in this paper shares many ideas regarding the ontological specification of the target domain with the work of Embley et al. [6]. However, instead of a complicated document preprocessing that presents a potential point of failure, we attempt to use the ontological specification as early as possible. As we mention in the introduction, our approach proceeds in a bottom-up manner leaving the presentation style analysis to later stages. This allows to avoid the complex document preprocessing that is usually HTML-specific and it presents a potential source of errors.

We have successfully tested some of the presented concepts during the SemPub 2015 challenge [5]. Our solution [11] was however tailored to a given particular application. In this paper, we present a new method based on a general model of the target domain.

## 3 Task Specification

The goal of our method is extracting information corresponding to ontological *concepts* (classes) from documents. In Fig. 1, we show a sample class (a conference paper) that is taken from a larger ontology we used for a particular information extraction task [11].

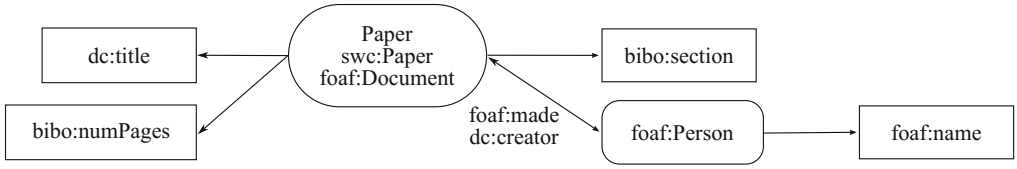



**Fig. 1.** Sample ontology representing a concept (Paper) and its data and object properties using the concepts and properties from the Bibliographic Ontology, FOAF Ontology and the Semantic Web Conference Ontology. The ovals represent the object properties and the rectangles represent its data properties.

According to the usual terminology in this area (for example [6]), the information about the instances of the given class (*individuals*) is represented by *data records* in the source documents. Each data record consists of multiple *data fields*, sometimes also called data units [13] that provide the lexical representation of some data properties (lexical properties) of the individual. The data fields are represented as text strings contained in the document text. Thus, a data record can be defined as a set of data fields that describe the same individual.

The task we investigate in this paper is to recognize all the data records in the source documents that belong to a single entity that is known in advance. Considering the Paper class in Fig. 1 as the input concept specification, the task is to recognize all the data records in the source documents that contain the information about individual papers containing their titles, author names, sections and pages.

## 4 Method Overview

We assume processing of web documents containing multiple data records corresponding to the same concept. The data records are presented in one or more source documents in a visually consistent way (we discuss the visual consistency in more detail in Sect. 7). The key idea is to discover the most frequent visual presentation patterns that occur in the source documents and that are used for presenting the data records. Subsequently, the data records are extracted using the discovered patterns. The method in general does not involve any learning phase on a training set of documents. For every extraction task, it only analyses the presentation patterns in the given source document. However, a trained

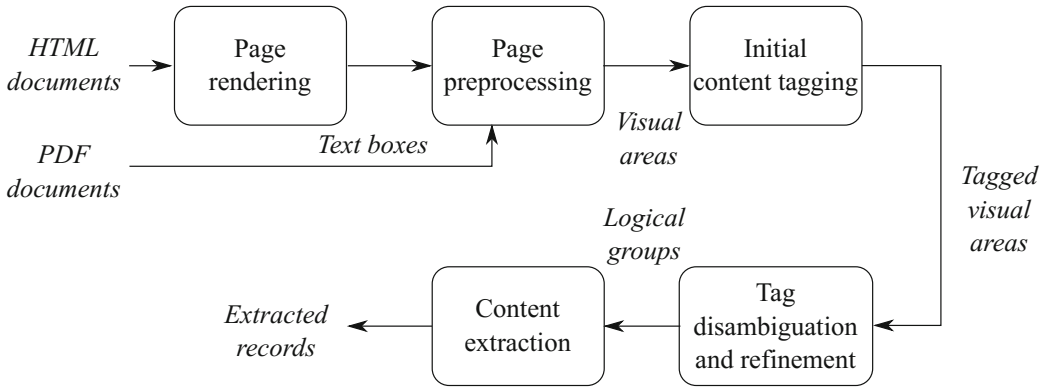



**Fig. 2.** Method overview

classifier may be used as one of the sources of the necessary background knowledge for certain application domains. We demonstrate one such application in Sect. 9.4.

Figure 2 shows the overview of our method. It operates on a visual representation of the source documents that is independent on the underlying code. Therefore, the first step is the document preprocessing that consists of creating an uniform representation of the source documents as a set of *visual areas*.

Next, in the initial tagging step, we perform an approximate recognition of the individual parts of the document content. This step gives a rough idea about the possible meaning of the individual visual areas; that means which visual areas might possibly correspond to some particular data fields. The result is represented by adding *tags* to the respective visual areas. Since the initial tagging is only approximate, some visual areas may obtain multiple tags and some of them may be tagged incorrectly.

Therefore, in the next step, we discover the most frequent presentation patterns used in the source documents and we use them to disambiguate and refine the assigned tags. The most supported visual patterns are then used for recognizing the desired data records.

In the following sections, we discuss the details of all the individual steps.

## 5 Input Document Preprocessing and Representation

The purpose of the input document preprocessing is to create a unified, format-independent model of the document content and its visual presentation. This step is typical for all the visually oriented information extraction approaches; we may mention the Visual block model used in [1], Page layout model [2] or the Visual block tree in [13].

Typically, all these models have a hierarchical structure which corresponds to the typical visual organization of the content in a web page. However, in our approach, we do not take into account the overall visual organization of the

page such as visually separated block or sections. Instead, we employ a bottom-up approach, that considers only the individual parts of the text content, their visual style and mutual visually expressed relationships. Therefore, we do not need to represent the complete visual hierarchy of the page and we use only a simplified flat model consisting of a set of *visual areas* as we define below.

The input of the preprocessing step is a set of *text boxes* contained in the source document. With a text box, we understand a rectangular area in the displayed page with a know position, size containing a portion of the document text. For HTML documents, the information about the text boxes is available from a rendering engine after the document has been rendered. In case of PDF documents, this information is directly available in the source document. In both cases, the information about the visual style of the contained text (such as the used font or color) is also available for each box.

In the preprocessing step, we create visual areas from the text boxes. A visual area provides an abstraction over the rendered boxes. It is a rectangular area in the rendered page that corresponds to one or more displayed text boxes depending on the chosen *granularity* as we explain below. We define a visual area $a$ as follows:

$$a = (rect, text, style, B) \tag{1}$$

where $rect = (x, y, w, h)$ is a rectangle representing the area position and size in the rendered page, $text$ is the text string contained in the area and $style$ is the area style:

$$style = (fs, w, st, c, bc) \tag{2}$$

where $fs$ is the average font size used in the visual area, $w \in [0, 1]$ is the average font weight where 1 means the whole area written in bold font and 0 means the whole area written in regular font, $st \in [0, 1]$ is the average font style (1 for italic font, 0 for regular font) and $c$ and $bc$ are the foreground and background colors used in the area. Finally, $B = b_1, b_2, \ldots, b_n$ is the set of boxes contained in the area ($n > 0$).

As the result of the preprocessing step, we obtain a set $A$ of all the visual areas in the page:

$$A = a_1, a_2, \ldots, a_m \tag{3}$$

where $m$ is the total number of visual areas in the page. Then, for any pair of visual areas $a_i, a_j \in A$ the corresponding sets of boxes $B_i$, $B_j$ are disjoint ($B_i \cap B_j = \oslash$) and the corresponding rectangles $rect_i$, $rect_j$ do not overlap.

### 5.1 Visual Area Granularity

The granularity of the visual areas generally depends on the application domain. In Sect. 9, we give several examples of the application domains and we discuss the chosen visual area granularity for each of them. The highest possible granularity

is obtained when for each visual area $a_i \in A$, the corresponding set of boxes $B_i$ contains a single box ($|B_i| = 1$). However, for most application, we choose a higher granularity ($|B_i| \geq 1$). Typical choices are the following:

- *Inline-level granularity* – the visual areas are formed by sets of neighboring boxes (based on their positions on the page) that are vertically aligned to a single line and they share a consistent visual style as defined in (2). This level approximately corresponds to inline-level elements used in HTML documents.
- *Block-level granularity* – the visual areas are formed by sets of boxes that form a visually separated block of text in the page (for example a text paragraph). We use a simple block detection method proposed in [3] that is based on the discovery of clusters of adjacent boxes based on their positions in the page.

Depending on the chosen granularity, we obtain a larger or smaller set $A$ of visual areas that represent the elementary pieces of the document content in the following steps of information extraction.

## 6 Initial Content Tagging

The purpose of the initial content tagging is to recognize all the visual areas that possibly might correspond to an extracted data field. Shortly, we want to identify the pieces of information that possibly "look like" some data field (for example a paper author name) when viewed separately. Each visual area is considered separately and it is assigned *tags* that indicate its possible meanings.

Based on the target domain, we define a set $T = t_1, t_2, \ldots, t_n$ of tags that may be assigned to visual areas. Each tag is identified by its name and it represents a particular data field to be extracted. For example, for the domain of conference papers shown in Fig. 1, we obtain the following set of tags corresponding to the data properties of the papers:

$$T = \{\text{title}, \text{authors}, \text{section}, \text{pages}\} \tag{4}$$

where the individual tags denote the paper title, author names, section title and page numbers respectively. For each tag, we define a *tagging* function that assigns a *support* to every visual area and tag:

$$\mathit{tagging} : A \times T \to \mathbb{R}_{[0,1]} \tag{5}$$

For a visual area $a \in A$ and a tag $t \in T$, the assigned support is a number $s \in [0, 1]$ that represents the probability that the visual area has the meaning that corresponds to the given tag. When $s > 0$, we say that the tag $t$ has been assigned to $a$ with the given support; for $s = 0$, we say that $t$ has not been assigned to $a$. Multiple tags may be assigned to a single area (for example, the number *"15"* may be recognized as both hour and minutes in the time domain).

The initial tagging represents a highly approximate estimation of the meaning of the individual visual areas which is used as a starting point for further

refining. We note that some of the tags (such as *title* and *section*) cannot be reliably distinguished when considering the visual areas separately. In that case, the visual area may obtain both tags (that means it may correspond to both the paper and section title) and later, the tags are disambiguated using the presentation context as described in Sect. 7.

From the practical point of view, we implement the *tagging* function as a set of *taggers* where the tagger is a procedure that is responsible for computing the support of a single particular tag given a visual area. The tagger implementation may be very variable but generally, we consider the following approaches to the tag assignment that may be combined arbitrarily:

- *DBPedia concept annotation* for example using the *Spotlight* tool [4].
- *Named entity recognition (NER)* may be used for recognizing the entities such as personal names or locations depending on the used NER classifier.
- Occurrences of *keywords* (for example month names), *numerical values* in given ranges or specific *regular expressions*.
- *Visual classification.* As we have shown in our earlier work [3], it is possible to use the visual features of the areas such as the used font, colors, position within the page or amount of contained text to create a classifier, that is first trained on a set of manually annotated documents and then, it may be used for recognizing the meaning of new, previously unseen visual areas in new documents. Unlike the remaining tagging methods, the visual classification approach requires a training set of documents for setting up the classifier as we show on a practical example in Sect. 9.4. However, the trained classifier may be later used for a whole set of documents coming even from different web sources.

For each tag, there is a single tagger defined that takes into account different criteria. The tagger may combine multiple methods with different supports. For example, the personal names may be recognized by DBPedia concept annotation (with the highest support) but the NER classifier may be used as a fallback solution (with a lower support) for recognizing the names that have no corresponding DBPedia resource.

## 7 Tag Disambiguation

After the visual areas have been approximately tagged, we disambiguate the tags by considering combinations of the data fields that are expected in the extracted data records (for example considering the *title* – *authors* or *title* – *pages* combination in our example in Fig. 1). We assume that all the data records are presented in a visually consistent way in the source document. Based on this assumption, we define presentation constraints on the data records that must apply for considering the records to be visually consistent. Then, the disambiguation task consists of finding the best matching record presentation and layout that meets the visual consistency constraints on one side and covers as many tagged visual areas as possible on the other side.

### 7.1 Visual Presentation Constraints

For considering the data records to be visually consistent, we require both the consistent presentation style of the individual data fields and consistent layout of the individual fields that form a single data record.

**Text Style Consistency.** For the individual text fields, we require that the visual areas with the same tag assigned (for example all the paper titles) have the same visual style in the document. We have defined the area style as a tuple of visual features (2). Let's consider a set of set of visual areas $A_t$ that have the tag $t$ assigned and let $S_t$ be a set of styles of all the visual areas in $A_t$. Then, let $n_f$ be the number of visual features that have equal values for all the styles $s \in S_t$. We say that $A_t$ has a consistent style if $n_f$ is over certain threshold.

Based on our practical experiments, we allow one visual feature that is often used by the document authors to further distinguish the individual records (for example some papers considered to be more important have a bold title or use a different color). Therefore, we use $n_f = 4$ for our experiments.

**Content Layout Consistency.** The layout consistency constraint is based on our assumption that the layout relationships between the individual data fields expressed by their mutual positions in the page are the same for all data records. For this purpose, we define four relations $R_{side}, R_{after}, R_{below}, R_{under} \subseteq A \times A$ that are defined based on the positions of the areas in the page. Considering a pair of visual areas $a_1, a_2 \in A$ and their respective positions $rect_1, rect_2$ in the page, we define the relations as follows:

- $(a_1, a_2) \in R_{side}$ when $a_1$ and $a_2$ are on the same line (their $y$ coordinates overlap), $a_2$ is placed to the right of $a_1$ without any other visual area being placed between $a_1$ and $a_2$ and the horizontal distance between $a_1$ and $a_2$ is not larger than 1 em[1] (shortly, $a_2$ placed next to $a_1$).
- $(a_1, a_2) \in R_{after}$ when $a_1$ and $a_2$ are on the same line and $a_2$ is placed to the right of $a_1$ anywhere on the line ($a_2$ is on the same line after $a_1$).
- $(a_1, a_2) \in R_{under}$ when $a_1$ and $a_2$ are placed roughly in the same column (their $x$ coordinates overlap) and $a_2$ is placed below $a_1$ without any other visual area being placed between $a_1$ and $a_2$ and the vertical distance between $a_1$ and $a_2$ is not larger than 0.8 em ($a_2$ is placed just below $a_1$).
- $(a_1, a_2) \in R_{below}$ when $a_1$ and $a_2$ are placed roughly in the same column (their $x$ coordinates overlap) and $a_2$ is placed anywhere below $a_1$.

As we may notice, $R_{side} \subseteq R_{after}$ and $R_{under} \subseteq R_{below}$. For each pair of data fields, we choose the most supported one by trying to cover as many tagged visual areas as possible using each relation. Since one-to-many relationships are allowed between the data fields, any of the above relations may turn out to be the most supported one.

[1] In typography, 1 em is a length corresponding to the point size of the current font.

### 7.2 Matching the Visual and Semantic Relationships

The tag disambiguation in our approach is based on discovering the most supported combinations of the tagged areas in the page. Considering the target domain described by an ontology (such as our example in Fig. 1), we find the binary relationships with the one-to-many or one-to-one cardinality between the different data properties in the ontology. We assume that the same semantic relationships between two data properties are represented by the same layout relation between the corresponding visual areas for all the data records in the page and in the same time, the visual areas corresponding to the same data type properties have the consistent visual style as defined in Sect. 7.1.

In our sample ontology, we may identify the following one-to-many (or one-to-one) relationships that are expected to have a corresponding visual representation in the document: *section* – *title*, *title* – *author*, *title* – *pages*. Note that the paper title may be viewed as a record-identifying field here as defined in [6].

Let's consider a single relationship between the properties represented by the tags $t_1$ and $t_2 \in T$. Let $s_{min}$ be a minimal value of the tag support (5) that is required for considering the area to have the given tag assigned and let $A_{t_1}$ and $A_{t_2}$ be the sets of visual areas that have the respective tags assigned:

$$A_{t_1} = \{a \in A : ((a, t_1), s) \in tagging \wedge s \geq s_{min}\} \tag{6}$$

$$A_{t_2} = \{a \in A : ((a, t_2), s) \in tagging \wedge s \geq s_{min}\} \tag{7}$$

and let $S_{t_1}$ and $S_{t_2}$ be the sets of all styles (2) of the visual areas that belong to $A_{t_1}$ and $A_{t_2}$ respectively. We define a *configuration* of a record extractor as follows:

$$c = (s_{t_1}, s_{t_2}, R) \tag{8}$$

where $s_{t_1} \in S_{t_1}$, $s_{t_2} \in S_{t_2}$ and $R$ is a layout relation as defined in Sect. 7.1. For each such configuration, we may find a set of matching pairs of visual areas:

$$M_c = \{(a_1, a_2) : a_1 \in A_{t_1} \wedge a_2 \in A_{t_2} \wedge style(a_1) = s_{t_1} \wedge style(a_2) = s_{t_2} \wedge (a_1, a_2) \in R\} \tag{9}$$

where $style(a_1)$ and $style(a_2)$ are the styles of $a_1$ and $a_2$ respectively. The goal of the tag disambiguation to find a configuration $c$ with the largest set $M_c$ of the corresponding area pairs.

The whole tag disambiguation algorithm for a pair of tags $t_1, t_2$ corresponding to a one-to-many relationship in the domain ontology may be summarized in the following steps:

1. Compute $A_{t_1}$, $A_{t_2}$ and the corresponding sets of styles $S_{t_1}$ and $S_{t_2}$ with the minimal support $s_{min}$ set to a higher value (we use $s_{min} = 0.6$ for considering only the tags assigned with some safe support).
2. Compute $M_c$ for all possible configurations $c$ and find the resulting configuration $c_x = (s_{t_1 x}, s_{t_2 x}, R_x)$ where $M_{c_x}$ is the largest set of matching pairs.

3. Decrease $s_{min}$ and recompute $A_{t_1}$, $A_{t_2}$ and $S_{t_1}$ and $S_{t_2}$ in order to consider even the areas with the tags assigned with a low support (we use $s_{min} = 0.1$).
4. Recompute $M_{c_x}$ for the previously discovered configuration $c_x$.

After the last step, $M_{c_x}$ contains visually consistent pairs of visual areas $(a_1, a_2)$ that correspond to the same pairs of data fields in the data records.

This process may be generalized to consider multiple one-to-many relationships: we just search for multiple configurations $c$ while maintaining the consistency of $s_{t_1}$ and $s_{t_2}$ and we obtain one set $M_{c_x}$ for each one-to-many relationship. For the one-to-one relationships, the process is equal; the only difference is in the $M_c$ size computation where we consider all the $(a_1, a_i)$ pairs (for all available values of $i$) as a single pair when computing the size of $M_c$.

## 8 Record Extraction

The obtained sets of matches $M_c$ identify the visual areas that contain the corresponding data fields from all the data records discovered in the document. Since the visual areas are directly linked to text boxes from the source document (1), the text content contained in the area may be obtained by a simple concatenation of the text contents of the text boxes.

Depending on the target domain and the area granularity chosen in the preprocessing step (see Sect. 5.1), it may be necessary to further postprocess the extracted text. The postprocessing includes converting the text content to particular data types (such as numbers or dates) or cleaning the text from an additional content. Finally, the obtained values may be mapped to the appropriate ontological properties.

## 9 Experimental Evaluation

We have implemented the proposed method of data records extraction in Java using our FITLayout framework[2]. The framework is able to process the HTML and PDF input documents by using the CSSBox rendering engine[3]. In order to demonstrate the applicability of the method, we have chosen four application domains, each having some specific features. Although it is not our primary aim to outperform the existing methods in terms of precision, we provide the evaluation of the achieved precision and recall for each sample application in order to show that the obtained results are usable in practice.

### 9.1 Conference Papers

For the conference paper domain, we have used a dataset from the Semantic Publishing Challenge at the ESWC 2015 conference [5]. The input dataset consists

[2] http://www.fit.vutbr.cz/~burgetr/FITLayout/.
[3] http://cssbox.sourcegorge.net/.

**Table 1.** Results for the conference papers task: number of records extracted, precision, recall and F-measure for two different data sets.

| Data set | #rec | P | R | F |
|---|---|---|---|---|
| (A) Complete dataset (115 documents) | 2420 | 0.976 | 0.955 | 0.966 |
| (B) Only documents containing page numbers | 883 | 0.997 | 0.975 | 0.986 |

of 148 selected CEUR workshop proceedings pages[4] from the years 1994–2014 containing the metadata about 2,500+ papers. The input HTML documents are very variable regarding both the code and the visual style. On this dataset, we would like to demonstrate that our approach is able to automatically adapt to the presentation style used in each document and based on the specified domain knowledge, it is able to extract the paper information from a large set of diverse documents.

The extraction task is defined by ontology in Fig. 1 and a set of taggers for assigning the *title*, *authors*, *section* and *pages* tags. For tagging the possible *authors*, we have used the Stanford NER classifier [7] for personal name recognition. The remaining taggers are implemented using regular expressions defining the allowed format of the corresponding data fields.

Since not all the documents contain the page numbers and sections, we have run two experiments: (A) on the complete data set (148 documents) with matching only the *title* – *authors* pairs and (B) on a subset of documents containing the sections and page numbers (67 documents) with matching the complete records. We have used the evaluation data provided by the SemPub Challenge organizers to evaluate our results and we provide the obtained results in Table 1. As we may notice, we have obtained better results for the (B) dataset which has two main reasons: first, the (B) dataset contains newer documents that are more visually consistent and second, by adding *pages* and *section* tags, the disambiguation is more efficient (more inconsistent combinations may be excluded from the result).

### 9.2 Sports Results

For the demonstration of the DBPedia concept matching usage, we have chosen the sports results domain as an example of integrating a rapidly changing external data source with DBPedia. We have extracted the records containing athlete *name*, *country* and current *points* from the current tennis and cycling rankings available on the web.

We have used DBPedia Spotlight web service for recognizing the athlete names (the matched DBPedia resource should be instance of `dbo:Athlete`) and countries (instance of `dbo:Country`). Moreover, we have used Stanford NER classifier for recognizing the personal names a locations in case no corresponding resource is available in DBPedia. All visual areas containing a numeric value are considered a possible *points* value and tagged with the corresponding tag.

[4] http://ceur-ws.org/.

For every source document[5], we have prepared the "golden standard" data for evaluation by manually transforming the source HTML code to a structured CSV table using a text editor. The results in Table 2 show that based on the assigned tags, our method is able to automatically infer the presentation pattern used for presenting the data records and extract the records with a high precision. In a few cases, the personal names are not identified correctly (there is no corresponding DBPedia resource and the NER classifier failed to recognize the name) which is the reason of lower recall. The resulting extracted records are linked to the corresponding athlete resources in DBPedia. This demonstrates the possibility of an easy integration of an external resource with DBPedia without any predefined templates.

### 9.3 Timetables

Timetables provide a data source containing an extremely low amount of labels and other additional information that could be used for the data interpretation. Actually, a timetable often contains only the data (hours, minutes, station names) formatted in a specific way leaving its interpretation to a great extent on the experience of the human reader. Motivated by a practical need, we have used the timetables available at the official Czech public transportation timetable portal.[6] The timetables are published here in PDF files (see Fig. 3 for an example) providing a good example of processing data-rich PDF documents.

**Table 2.** The sports results tasks

| Source | #rec | P | R | F |
|---|---|---|---|---|
| ATP rankings (tennis.com) | 200 | 1.000 | 0.935 | 0.966 |
| WTA rankings (tennis.com) | 200 | 1.000 | 0.925 | 0.961 |
| Road cycling rankings (uci.ch) | 2488 | 1.000 | 0.933 | 0.965 |
| Mountain bike rankings (uci.ch) | 1627 | 1.000 | 0.978 | 0.989 |

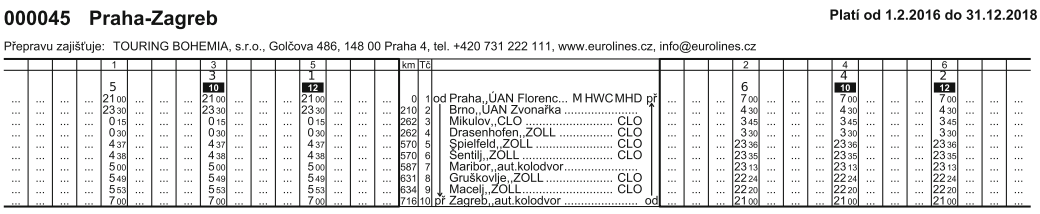



**Fig. 3.** A sample timetable

[5] The URLs of the source documents were http://www.tennis.com/rankings/ATP/, http://www.tennis.com/rankings/ATP/, http://www.uci.ch/road/ranking/ and http://www.uci.ch/mountain-bike/ranking/ respectively.

[6] http://portal.idos.cz.

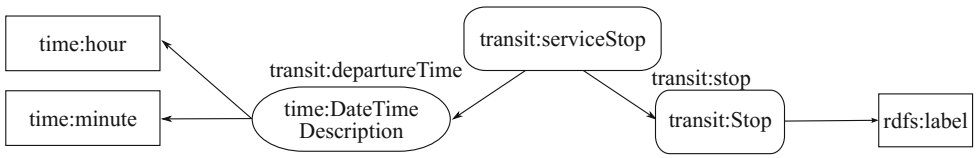



**Fig. 4.** Ontology used for timetables. The concepts and properties come from the OWL Time ontology and Transit ontology

The domain knowledge is represented by an ontological description in Fig. 4 and taggers for the tags *hours* and *minutes* based on the recognition of numbers in the corresponding range and for *stops* (stop names) based on matching with a fixed list of existing stops (which is available in this domain) combined with regular expressions used when the matching fails.

We have tested our method on 30 different time table documents from the above mentioned portal. The extracted data was compared with a golden standard that was created manually by transforming the PDF files to CSV data using a semi-automatic transformation based on regular expressions tailored for the particular documents. Because the time tables contain a large amount of $(hour, minute, stop)$ records (we have obtained 5130 records in total), the tag disambiguation works very efficiently in this case and we have extracted all the records correctly ($P = R = 1.0$). It is worth noting that all the *hour* and *minute* pairs have been identified correctly although the initial tagging is very ambiguous (many visual areas share both tags after the initial text classification).

### 9.4 News Articles

We have chosen the news articles domain to demonstrate a different application scenario. Unlike the documents in the previous domains that typically contained many data records (papers or times), the news web pages usually contain one full article in a document. However, each news website contains many such documents that follow a visually consistent presentation style. Therefore, we may treat a set of documents as a single input page containing multiple articles.

For this task, we view the individual news articles as data records containing data fields that we have assigned the following tags: *title* (article title), *author* (author name), *pubdate* (date of publication) and *paragraph* (a paragraph of text). Considering the title to be the record-identifying field, the *title – paragraph* pairs correspond to a one-to-many relation, the *title – author* and *title – pubdate* pairs are one-to-one relations.

Due to the specific properties of the news domain where it may be difficult to recognize the individual parts such as titles and subtitles by text classification only, we employ a visual classification approach that allows to assign the tags to the areas based on their visual appearance. This approach that we have presented in detail in [3] uses the visual features of the individual visual areas: Font features such as the font size, weight and style, spatial features (position in the page and size), text features (numbers of characters and lines) and color features

**Table 3.** Results for the news articles task: precision, recall and F-measure with and without using tag disambiguation

| Method | Precision | Recall | F-measure |
|---|---|---|---|
| Visual classifier only | 0.593 | 0.790 | 0.678 |
| Visual classifier + disambiguation | 0.978 | 0.986 | 0.982 |

(luminosity, contrast). The values of the features are expressed numerically and used as an input for a generic classifier[7]. Therefore, in contrast to the other applications presented in the previous sections, a training set of documents is required for setting up the classifier. Later in the classification step, the trained classifier directly assigns tags to the visual areas in new documents.

For testing, we have used the news articles on reuters.com and cnn.com news portals. We have taken 30 documents with articles from each website. We have manually annotated the source documents by manually assigning the appropriate tags to the individual visual areas in the documents.[8] Then, 5 documents from each web site were used for training the classifiers (one for each source website) based on the visual features of the manually tagged areas. Later, the trained classifiers were used for assigning tags to all the visual areas in the complete dataset from the given website.

The results obtained are shown in Table 3. The first row shows the values obtained by comparing the classification results with the manually assigned tags. This corresponds to the scenario presented in [3]. The second line shows the result with disambiguation where the visual classification results were used as the initial tagging for the tag disambiguation process described in Sect. 7. As we may see, the disambiguation greatly improves the resulting precision and recall.

## 10 Conclusions

We have presented a record extraction approach from web documents that is based on searching the most frequent visual presentation patterns in the documents while assuming that multiple instances of the records are available in the documents. The extraction itself is based only on the knowledge available for the target domain that includes the expected structure of the extracted records and an estimation of possible values (or alternatively a style) of the data fields. We consider this as the main benefit of the presented approach. As the result, the method is independent on the source document format, and it does not rely on any kind of templates used or labels or annotations present in the source documents. The experimental results demonstrate the applicability of the approach for different scenarios and document sources.

[7] For our experiments, we have used the J.48 classifier from the WEKA package (which is an implementation of the C4.5 decision tree classifier) mainly for its speed.

[8] The used FITLayout framework provides a graphical annotation tool that was used for this task.

**Acknowledgments.** This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science – LQ1602.

## References

1. Anderson, N., Hong, J.: Visually extracting data records from query result pages. In: Ishikawa, Y., Li, J., Wang, W., Zhang, R., Zhang, W. (eds.) APWeb 2013. LNCS, vol. 7808, pp. 392–403. Springer, Heidelberg (2013). doi:10.1007/978-3-642-37401-2_40
2. Burget, R.: Hierarchies in HTML documents: linking text to concepts. In: 15th International Workshop on Database and Expert Systems Applications, pp. 186–190. IEEE Computer Society (2004)
3. Burget, R., Burgetová, I.: Automatic annotation of online articles based on visual feature classification. Int. J. Intell. Inf. Database Syst. **5**(4), 338–360 (2011)
4. Daiber, J., Jakob, M., Hokamp, C., Mendes, P.N.: Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the 9th International Conference on Semantic Systems (I-Semantics) (2013)
5. Iorio, A.D., Lange, C., Dimou, A., Vahdati, S.: Semantic publishing challenge – assessing the quality of scientific output by information extraction and interlinking. In: Gandon, F., Cabrio, E., Stankovic, M., Zimmermann, A. (eds.) SemWebEval 2015. CCIS, vol. 548, pp. 65–80. Springer, Cham (2015). doi:10.1007/978-3-319-25518-7_6
6. Embley, D.W., Campbell, D.M., Jiang, Y.S., Liddle, S.W., Lonsdale, D.W., Ng, Y.K., Smith, R.D.: Conceptual-model-based data extraction from multiple-record web pages. Data Knowl. Eng. **31**(3), 227–251 (1999)
7. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by Gibbs sampling. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL 2005, pp. 363–370 (2005)
8. Goh, P.L., Hong, J.L., Tan, E.X., Goh, W.W.: Region based data extraction. In: 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp. 1196–1200, May 2012
9. Hong, J.L., Siew, E.G., Egerton, S.: Information extraction for search engines using fast heuristic techniques. Data Knowl. Eng. **69**(2), 169–196 (2010). doi:10.1016/j.datak.2009.10.002
10. Kolchin, M., Kozlov, F.: A template-based information extraction from web sites with unstable markup. In: Presutti, V., Stankovic, M., Cambria, E., Cantador, I., Di Iorio, A., Di Noia, T., Lange, C., Reforgiato Recupero, D., Tordai, A. (eds.) SemWebEval 2014. CCIS, vol. 475, pp. 89–94. Springer, Cham (2014). doi:10.1007/978-3-319-12024-9_11
11. Milicka, M., Burget, R.: Information extraction from web sources based on multi-aspect content analysis. In: Gandon, F., Cabrio, E., Stankovic, M., Zimmermann, A. (eds.) SemWebEval 2015. CCIS, vol. 548, pp. 81–92. Springer, Cham (2015). doi:10.1007/978-3-319-25518-7_7
12. Su, W., Wang, J., Lochovsky, F.H.: ODE: ontology-assisted data extraction. ACM Trans. Database Syst. **34**(2), 121–1235 (2009). doi:10.1145/1538909.1538914

13. Weng, D., Hong, J., Bell, D.A.: Extracting data records from query result pages based on visual features. In: Fernandes, A.A.A., Gray, A.J.G., Belhajjame, K. (eds.) BNCOD 2011. LNCS, vol. 7051, pp. 140–153. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24577-0_16
14. Weng, D., Hong, J., Bell, D.A.: Automatically annotating structured web data using a SVM-based multiclass classifier. In: Benatallah, B., Bestavros, A., Manolopoulos, Y., Vakali, A., Zhang, Y. (eds.) WISE 2014. LNCS, vol. 8786, pp. 115–124. Springer, Cham (2014). doi:10.1007/978-3-319-11749-2_9
15. Zheng, X., Gu, Y., Li, Y.: Data extraction from web pages based on structural-semantic entropy. In: Proceedings of the 21st International Conference on World Wide Web, WWW 2012 Companion, pp. 93–102. ACM, New York (2012). doi:10.1145/2187980.2187991

## C.3 Integration of Unstructured Web Data Sources

Burget, R.: Model-Based Integration of Unstructured Web Data Sources Using Graph Representation of Document Contents. In *15th International Conference on Web Information Systems and Technologies*. SciTePress. 2019. ISBN 978-989-758-386-5. pp. 326–333.

# Model-based Integration of Unstructured Web Data Sources using Graph Representation of Document Contents

Radek Burget[a]
*Faculty of Information Technology, Brno University of Technology, Bozetechova 2, Brno, Czech Republic*

Keywords: Information Integration, Domain Modelling, Document Processing, Structured Record Extraction.

Abstract: Unstructured or semi-structured documents on the web are often used as a media for publishing structured, domain-specific data which is not available from other sources. Integration of such documents as a data source to a standard information system is still a challenging problem because of the very loose structure of the input documents and usually missing semantic annotation of the published data. In this paper, we propose an approach to data integration that exploits the domain model of the target information system. First, we propose a graph-based model of the input document that allows to interpret the contained data in different alternative ways. Further, we propose a method of aligning the document model with the target domain model by evaluating all possible mappings between the two models. Finally, we demonstrate the applicability of the proposed approach on a sample domain of public transportation timetables and we present the preliminary results achieved with real-world documents available on the web.

## 1 INTRODUCTION

Despite much effort dedicated to the development of different technical means for annotating the semantics of the presented data such as Microformats[1], RDFa[2] and others, the World Wide Web is still an extremely large source of mostly unannotated documents. These documents often contain structured and potentially useful data presented in a way that is convenient for human readers but it is completely unsuitable for automated processing. Therefore, using the documents as a data source for traditional information systems that are based on structured data models presents a challenging task.

A typical domain-oriented information system uses a structured data representation and storage (for example a relational database), which has been designed based on the analysis of the target domain, identification of the individual entities, their properties and the relationships among them. However, on the web, many potential sources of domain-specific data have the form of documents designed primarily for human readers. Although the data contained in these documents follow basically the same structure that comes from the target domain, their integration to an existing information system is difficult because of the very loose way of their presentation without any formal annotation.

In (Burget, 2017), we have mentioned several domains, where this situation is quite typical such as scholarly data (conference proceedings contents), sports results or public transport time tables. In all these (and many other) domains, the data has a fixed and predictable structure that potentially allows its integration to existing applications in the respective domains. However, the corresponding data sources often have the form of periodically published documents (mostly web pages; PDF documents are typical for some domains such as timetables) whose human interpretation is assumed for understanding the presented data.

Traditionally, the integration of such web sources is implemented using different kinds of *wrappers* that recognize data fields in the documents by analyzing the underlying document code – mostly the HTML code represented as a Document Object Model (DOM) (Schulz et al., 2016). For each data source (the source of the input documents), the corresponding code patterns are different and therefore, a specific wrapper must be prepared. Such approach is reliable and feasible when considering a limited number of previously known data sources that provide a larger number of documents but it is not practical at

[a] https://orcid.org/0000-0001-5233-0456
[1] https://microformats.io/
[2] https://rdfa.info/



Burget, R.
Model-based Integration of Unstructured Web Data Sources using Graph Representation of Document Contents.
DOI: 10.5220/0008350103260333
In *Proceedings of the 15th International Conference on Web Information Systems and Technologies (WEBIST 2019)*, pages 326-333
ISBN: 978-989-758-386-5

all, when the input documents come from previously unknown sources, each document has been prepared independently and uses a completely different way of data presentation.

In this paper, we propose a model-based approach aiming to overcome the specific details of the individual documents by an automatic discovery of a mapping between the previously defined domain data model and the presented data records. The main presented contributions are the following:

- We present a technology- and language-independent graph-based model of the document contents that allows to interpret the contained data in different alternative ways.
- We propose a method for evaluating the possible mappings between the created document model and the target domain model that describes the expected structure of the contained data and for choosing the best mapping based on a statistical analysis.
- We demonstrate the application of the described approach on a sample domain of public transport timetables.

We also include preliminary results of this work in progress that show the applicability of the proposed document model and mapping methods on real-world documents.

## 2 RELATED WORK

The research in the topic of data record extraction from web documents has been running for over 20 years. Apart from historical HTML-based approaches (Schulz et al., 2016), due to the evolution of the web technology (mainly in the HTML and CSS languages and the dynamic web pages) and the increasing complexity of web documents, the recent approaches usually combine the analysis of the document code, with visual presentation properties (Potvin and Villemaire, 2019; Shi et al., 2015). However, most of the current methods use DOM[3] as the primary document representation (Figueiredo et al., 2017; Guo et al., 2019; Lockard et al., 2018; Shi et al., 2015; Yuliana and Chang, 2018). This limits the applicability of the methods to specific HTML documents where the DOM elements accurately delimit the desired data fields.

From the data integration point of view, the current methods infer the schema of the extracted records from the source documents themselves (Figueiredo et al., 2017; Shi et al., 2015; Yuliana and Chang, 2018). In all cases, the approach is to find a specific region or multiple regions (Figueiredo et al., 2017) that contain the records and then, a flat internal structure of the records is determined based on finding the regular patterns in the document code and by comparing the similarity and other characteristics of the repeating sequences. This approach allows easy application of the methods to any document independently on its domain; however, the integration of the extracted data to a domain information system requires further interpretation and transformation of the extracted records.

In contrast to the above mentioned data-driven approaches, there has been significantly less attention given to the research of the model-driven approaches. (Embley et al., 1999) uses a conceptual domain model that is directly mapped to HTML code based on different heuristics. In (Potvin and Villemaire, 2019) a flat list of extracted data field is used and (Lockard et al., 2018) integrates the extracted data with an existing knowledge base.

In our previous research (Burget, 2017), we have proposed a basic approach for matching individual binary relationships in a domain model to visual presentation patterns in the documents. In this paper, we generalize the matching to the whole domain models and above all, we introduce a formal graph-based document model of the input documents that makes the matching possible.



## 3 THE DATA INTEGRATION TASK

The data integration task we consider in this paper is the following: We have a (potentially unlimited) collection of unstructured input documents on the source side and a structured domain-specific information system on the target side.

The target information system is typically designed based on the analysis of the particular domain, which results in a domain data model such as a entity-relationship diagram (ERD) or its equivalent depending on the used design methodology. The model captures the basic entity sets, their properties (attributes) and the relationships among them. Independently on whether an ERD or another formalism is used, we may define a domain model for our purpose as follows:

**Definition 1.** *A domain model is a tuple* $D = (E, P, R)$*, where* $E$ *is a set of entity sets,* $P$ *is a set of properties (attributes in ERD) and* $R \subset (E \times (E \cup P))$ *is the set of relationships.*

[3] https://www.w3.org/DOM/

Figure 1 shows a simple ERD for the public transport timetables domain. Note that we consider just a part of the ERD that is relevant to the considered data sources; the complete ERD for a real-world information system would be obviously significantly larger.

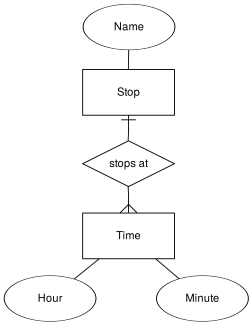



Figure 1: An entity-relationship model for the domain of public transportation timetables with two entity sets (*Time*, *Stop*), three properties (*Hour*, *Minute*, *Name*) and one relationship (*stops at*).

At the input, we assume a collection of *documents* that contain visually presented structured *data records* consisting of several *data fields*. The documents come generally from different sources and therefore, the way of data presentation, formatting or the implementation may be different for every single document. However, we put the following assumptions on the input documents:

- We assume formatted text documents where the document creator may specify the *visual properties* (fonts, colors, etc.) for every part of the document text as well as the *visual organization* of the contents (alignment, spacing, etc.) by any means. For the web sources, the HTML web pages and PDF documents are the most typical but our content model presented below in section 4 is independent on the actual technology.
- Every document contains multiple *data records* consisting of the *data fields* that may be directly mapped to the properties in the target ERD (i.e. without any additional transformations) and the records are *consistent* regarding their structure and visual presentation (their visual properties and organization as mentioned above).

Figure 2 gives the overview of the document processing process. First, the visual properties and positions of all parts of the document text are computed. This is the only task that depends on the document type. For some document types such as HTML, this requires rendering the document by a web browser. In PDF documents, the necessary information is available directly. In the next steps, we identify the *text chunks* that represent the candidate substrings of the document text that potentially could represent a data field. Based on the extracted text chunks, we build a *page contents model*, which is basically a graph that describes the visual properties of the individual chunks and the visually presented relationships among them. We describe the model and its construction below in section 4.

The key part of the information integration process consists of finding the most appropriate mapping between the created document contents graph and the domain model. For this purpose, we also represent the domain model as a graph of the entity properties and the relationships among them and we search for a best mapping between the two graphs. The details of this process are described in section 5.

In the following sections, we will use the already mentioned public transport timetables as a sample domain. Our goal is to integrate the data about the stops and the corresponding times from the timetable documents as shown in Figure 1. We believe, this domain is suitable for illustrating the individual steps for the following reasons:

- It is challenging. The timetables are a good example of source documents that present data in a very ambiguous way and even the human readers need some experience to interpret the data properly in some more complex cases.
- It is practically useful. Although there exist different portals and aggregators in this domain, they are usually limited to certain countries, regions or groups of companies and they typically do not provide their structured data to third parties.
- There are many highly diverse documents from different transportation companies available on the web.

However, the presented integration approach is not limited to a single domain as long as the above mentioned assumptions on the input documents are met.

# 4 DOCUMENT CONTENTS MODEL

The goal of the proposed document contents model is to capture the possibly relevant parts of the document contents and their mutual relationships based on their visual presentation. We define the model as a graph:

**Definition 2.** *The document contents model is defined as a graph* $G = (C, E)$*, where* $C$ *is a set of* text chunks

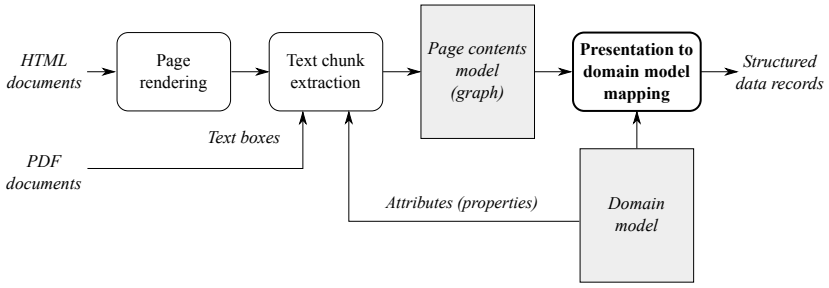



Figure 2: An overview of the data integration process.

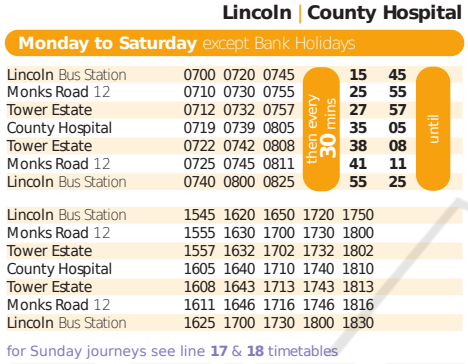



Figure 3: An example time table.

*that represent the relevant parts of the contents together with their visual formatting and form the vertices of the graph; $E \subset C \times C$ is a set of graph edges, that represent the* relationships *among the chunks as expressed by the document layout.*

With a *text chunk*, we understand any piece of content (a substring of the document text), that possibly represents a value of a domain property. In the moment of the chunk extraction, we do not decide, whether the given substring really represents a part of a data record; the goal is to identify all substrings that "look like" a value of a given property when considered separately.

**Definition 3.** *A* text chunk *is a tuple $c = (t_c, s_c, p_c)$, where $t_c$ is the text of the chunk (the actual substring of the document text), $s_c$ represents the visual style of the text and $p_c$ represents the position of the chunk as displayed in the resulting page.*

**Definition 4.** *The* chunk style *is further defined as $s_c = (fs, w, st, c, bc)$ where $fs$ is the average font size, $w \in [0,1]$ is the average font weight from 0 (normal font) to 1 (bold font), $st \in [0,1]$ is the average font style (1 for italic font, 0 for regular font) and c and bc are the computed foreground and background colors of the displayed chunk.*

**Definition 5.** *The* position *$p_c = (x, y, w, h)$ describes the x and y coordinates of the chunk in the page and its width w and height h.*

The edges $E$ of the graph represent the mutual *relationships* among the chunk pairs. Based on their mutual positions, we identify specific relationships that are interesting for further analysis of the whole data record organization. For example, two chunks may be in a *onRight*, *below*, *sameLine* or another relation as described in section 4.2.

Both the chunks and the relationships are extracted from rendered documents as shown in Figure 2. In the next sections, we provide the details of the chunk and relationship extraction.

## 4.1 Chunk Extraction

For the chunk extraction, we use a *connected line* as a smallest unit of the rendered document.

**Definition 6.** *A* connected line *represents a part of the document text that is positioned on a single line (considering the y coordinates of the individual characters) and it does not contain an empty space wider than a certain threshold $\Delta x$.*

In our experimental setup, we have used $\Delta x = 2.5f$ where $f$ is the average font size used at the considered line. The goal of this setting is to ensure that the normal text formed by space-separated words forms single lines and the parts separated with a larger space create separate connected lines.

**Example 1.** *In the example timetable in Figure 3, the header and footer text forms continuous text lines; however, the stop names and the time data are separated by a larger space. Thus, we obtain the connected lines "Lincoln Bus Station", "0700 0720 0745", "15" and "45" for the first line of the schedule, etc. Note that the connected lines are not necessarily consistent regarding their style; they may contain text with different font weights, colors, etc. as we may notice for the station names.*

For each domain model property $p \in P$ (see Definition 1), we extract a set $C_p$ of chunks from all the connected lines. The algorithm for the chunk discovery within the connected lines depends greatly on the type of the property $p$. For our sample domain, we have used a simple algorithm that finds all one- or two-digit numbers in the appropriate range for hours and minutes; the chunks for stop names are discovered using a simple regular expression allowing a sequence of alphanumeric characters and some commonly used punctuation. In our previous experiments on other domains (Burget, 2017), we have also mentioned the usage of named entity classifiers (Finkel et al., 2005) for recoginizing personal names and locations or even using the DBPedia Spotlight tool (Daiber et al., 2013) for recoginizing entities from the DBPedia dataset. Advanced algorithms for numeric value discovery have been proposed as well (Neumaier et al., 2016).

As the result of the chunk extraction, we obtain a complete set of chunks for all the properties $C = C_{p_1} \cup C_{p_2} \cup \ldots \cup C_{p_n}$ where $n = |P|$. Note that the chunk detection itself may be quite inaccurate as we use very approximate methods for the chunk extraction. The extracted chunks may even overlap; e.g. in the "Monks Road 12" string, we discover three chunks: the whole string forms the *name* chunk, the "12" substring forms the *hour* and *minute* chunks because both interpretations are possible. It is the task of the mapping phase (described in section 5) to complete the data records and exclude the incorrectly discovered chunks.

### 4.2 Relationship Modelling

After the chunks have been detected, we analyze all the chunk pairs $(c_1, c_2) \in C \times C$ and we investigate whether there is an relationship between $c_1$ and $c_2$ given by their mutual positions $(x_1, y_1)$ and $(x_2, y_2)$. We have identified several relationships that are interesting for further analysis. Every relationship is defined by a relation $E_x \subset C \times C$ and we say that there is a spatial relationship $x$ between $c_1$ and $c_2$ iff $(c_1, c_2) \in E_x$. Currently, we consider the following relationships:

- *onRight* – $(c_1, c_2) \in E_{onRight}$ when $c_1$ and $c_2$ are placed on the same line just next to each other and $c_2$ is on the right side of $c_1$.
- *after* – $c_2$ is on the same line anywhere to the right of $c_1$.
- *sameLine* – $c_1$ and $c_2$ are on the same line regardless their mutual positions.
- *below* – $c_2$ is placed just below $c_1$.
- *lineBelow* – $c_2$ is placed on a line that is just below $c_1$.

As we may see, a chunk pair may belong to multiple relations as the spatial relationships (e.g. *after* and *sameLine*) are not mutually exclusive.

Finally, the complete set of relationships is then $E = \bigcup E_x$ for all the relations $x$ listed above. Together with the set $C$ of chunks, it creates the document content graph as defined in Definition 2.

## 5 MAPPING TO THE DOMAIN MODEL

Our information integration approach is based on the assumption that some of the extracted text chunks may be mapped to the individual properties of the domain model as defined in Definition 1 and similarly, some discovered spatial relationships among them may be mapped to the domain model relationships. During the mapping phase, we find all possible *mappings* from the constructed document contents graph to the domain model, we evaluate them and finally, we use the best mapping found.

Below, we describe the representation of the domain model used for the final mapping. Further in section 5.2, we define a mapping formally and finally in section 5.3, we discuss the way of evaluating the individual mappings and finding the most suitable one.

### 5.1 Domain Model Transformation

As the first step, we transform the domain model to a simplified graph model that describes only the properties and relationships as the entity sets have no direct representation in the documents. An example model for the timetables domain is shown in Figure 4. The properties are divided into *groups* (the dashed boxes) where each group corresponds to a set of properties that are always presented together in a 1:1 relationship.

**Definition 7.** *The* domain graph model *is a graph $D_g = (G, R_g)$ where $G = \{G_1, G_2, \ldots, G_n\}$ is a set of property groups, $G_i \subset P$ and $G_i \cap G_j = \varnothing$ for any $1 \leq i, j \leq n$. $P$ is the set of domain properties. $R_g \subset G \times G$ is a set of relationships between groups.*

The domain graph is constructed from the domain model defined in Definition 1 as follows:

- All the properties of a single entity set belong to the same group.
- If two entity sets are in a 1:1 relationship, all their properties belong to a single group.

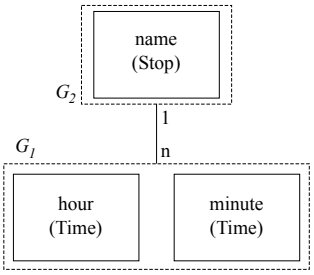



Figure 4: A domain graph model corresponding to the domain model shown in Figure 1 that represents the property groups and the relationships among them.

- The 1:M relationships are transformed to the relationships between the respective groups.

Currently, we don't consider M:N relationships in our method because they are difficult to represent in the documents in an understandable way and therefore, they are very rarely used in source documents.

Considering the example from Figure 1, we obtain two groups of properties: $G_1 = \{hour, minute\}$ and $G_2 = \{name\}$ as shown in Figure 4. Subsequently, we analyze the possible mappings between the document contents graph and the domain graph model.

## 5.2 Mapping Representation

When considering a particular document represented by the document contents graph, there exist many possible mappings between the chunks and the properties in the domain graph and similarly, between the relationships in the two graphs. In our approach, a mapping presents a hypothesis about the visual presentation of data records, which is subsequently evaluated and compared with other hypotheses.

Based on the above mentioned assumption that there exist multiple visually consistent data records in the source documents, a mapping basically describes two aspects of the records:

1. The visual style of the text chunks used for presenting each property $p \in P$ in the input document.
2. The actual spatial relationships (as mentioned in section 4.2) among the property values.

Let's consider the chunk style defined in Definition 4 and let $S$ be a set of all distinct chunk styles used in the input document. Further, let $R_s$ be the set of all spatial relationships between chunks discovered in the input document. Then, we may define the mapping between a property group $G_i$ in the domain graph and the document contents graph as follows:

**Definition 8** (Group Mapping.)**.** *For each group $G_i \in G$, the mapping is defined as $m_i = (f_{si}, f_{ri})$ where $f_{si} : G_i \mapsto S$ is a morphism that assigns a chunk style to each property in $G_i$ and $f_{ri} : G_i \times G_i \mapsto R_s$ assigns spatial relationships to the property pairs.*

The $f_{rg}$ morphism does not necessarily assign a relationship to all possible property pairs. For a unique description of the mapping, it is sufficient that the property pairs form a connected graph. For example, considering three properties $a$, $b$ and $c$, the mapping may contain $(a,b) \mapsto onRight$, $(a,c) \mapsto below$ (which can be read as $b$ is on the right side of $a$ and $c$ is below $a$). We obtain a connected graph of properties and therefore, it is not necessary to find any relationship for the remaining combinations such as $(b,c)$. Considering the group $G_1$ in Figure 4, it is sufficient to find one of the morphisms $(hour, minute) \mapsto r$ or $(minute, hour) \mapsto r$, where $r \in R_s$.

Similarly, we define an *inter-group mapping* that corresponds to the way how the connection of two groups is visually presented in the document:

**Definition 9** (Inter-group Mapping.)**.** *For a pair of groups $(G_i, G_j) \in G \times G$, the inter-group mapping is $m_{ij} = (p_i, p_j, r)$ where $p_i \in G_i$, $p_j \in G_j$ and $r \in R_s$.*

In other words, we define a spatial relationship $r$ between two properties where the first property belongs to the first group and the second property belongs to the second group. Again, we have to find enough mappings between the group pairs so that we obtain a connected graph of groups. Then, the complete mapping is $m = (M_G, M_I)$ where $M_G$ is a set containing a group mapping for each group in $G$ and $M_I$ is a set of the inter-group mappings.

**Example 2.** When considering our example timetable in Figure 3 and the domain graph in Figure 4, we find many different styles used for the presentation of the *hour* values in the document (when considering the style of all the chunks in $C_{hour}$) and similarly for the *minute* and *name* properties (the style morphisms $f_{s1}$ and $f_{s2}$). Moreover, we find different ways how the $(hour, minute)$ pair is possibly presented, e.g. *minute* is on the right side of *hour* or *minute* is below *hour* or *hour* is below *minute*, etc. (the $f_{ri}$ morphism). And finally, we find the possible presentation of the inter-group relation, e.g. *hour* is on the same line as *name*. Since *hour* and *name* are in separate groups in the domain graphs, we know that *hour* actually represents a complete $(hour, minute)$ group and there may exist multiple such pairs related to a single name because of the 1:N relationship between the groups.

By considering all combinations of chunk styles, and the applicable intra-group and inter-group rela-

tionship representations, we obtain a set $M$ of all possible mappings from the contents graph to the domain model graph.

### 5.3 Evaluation of the Mappings

The last step is the evaluation all the mappings and choosing the most suitable one. For each mapping $m \in M$, we apply the style and spatial relationship mappings on the document text chunks and as a result, we obtain a set of candidate data records, where each data field of the record is represented by a text chunk.

For the evaluation, the following aspects of the discovered candidate records are important:

- The number of chunks actually covered by the records. Although we admit that some of the chunks may have been incorrectly identified, we assume that the correctly identified ones prevail and thus, more chunks contained in the discovered records indicate a better result.
- Visual consistency of the records. The given mapping defines the actual visual style of the chunks mapped to the individual properties as well as the spatial relationships among them (e.g. hours and minutes being at the same line). However, the records may differ in the distance and alignment of the particular chunks. When evaluating consistency, we compare the individual records and we observe the variance of the corresponding distances among the chunks. The lower the overall variance is, the more consistent (and thus better) are the records.

Additionally, we allow using certain number of wildcards in style specifications. It is quite common in visual presentation that some of the records or data fields are distinguished from the others by a different background color, font style, etc. In our experiments, we allow one wildcard in the chunk style, i.e. based on the style defined in Definition 4, one the $(fs, w, st, c, bc)$ attributes may be disregarded.

As we may notice, the two evaluation criteria mentioned above are contradictory to some extent. It is easy to cover a large number of chunks and discover many data records when we allow low visual consistency of records and vice versa. For our experiments we have empirically set the total mapping score to $s = 0.6p + 0.4c$ where $p$ is the percentage of chunks contained in the records and $c$ is the visual consistency, $p, c \in [0..1]$.

## 6 EXPERIMENTAL EVALUATION

For the evaluation on real-world documents, we have implemented the proposed method in Java. For input document processing, we have used the CSSBox[4] rendering engine for HTML documents and the PDFBox[5] library for reading PDF documents.

Our preliminary tests (being this a work in progress) were run on 30 timetables in PDF available online on the websites of various transportation companies that operate in different en countries (Czechia, Spain, Italy and the United States). As a second use case, we have extracted the publication data (authors, titles and sessions) from CEUR Workshop Proceedings[6] (HTML documents).

The tests have shown the practical usability of the proposed document contents model described in section 4 as well as the domain mapping method. However, in about 10% of input documents, the correct mapping was not evaluated as the best one and the evaluation function had to be adjusted for obtaining correct results. Therefore, we consider the mapping evaluation the main issue for our ongoing research.

## 7 CONCLUSIONS

In this paper, we have proposed an approach to the integration of the data contained in web documents to structured information systems. Unlike most of the existing approaches that derive the data structure from the input documents, our method is driven by a previously defined domain model of the information system.

In order to make the information integration possible, we have designed a graph-based model of the document contents and subsequently, we have proposed a method for finding the best mapping of the document contents model to the domain model. Our preliminary results show that the approach allows integration of real-world HTML and PDF documents and mapping of the published data to the fixed domain model. The evaluation of the possible mappings seems to be the most challenging topic for our next research.

[4]http://cssbox.sourceforge.net
[5]https://pdfbox.apache.org/
[6]http://ceur-ws.org/

## ACKNOWLEDGEMENTS

This work was supported by the Ministry of the Interior of the Czech Republic as a part of the project Integrated platform for analysis of digital data from security incidents VI20172020062.

## REFERENCES

Burget, R. (2017). Information extraction from the web by matching visual presentation patterns. In *Knowledge Graphs and Language Technology: ISWC 2016 International Workshops: KEKI and NLP&DBpedia*, Lecture Notes in Computer Science vol. 10579, pages 10–26. Springer International Publishing.

Daiber, J., Jakob, M., Hokamp, C., and Mendes, P. N. (2013). Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*.

Embley, D. W., Campbell, D. M., Jiang, Y. S., Liddle, S. W., Lonsdale, D. W., Ng, Y.-K., and Smith, R. D. (1999). Conceptual-model-based data extraction from multiple-record web pages. *Data Knowl. Eng.*, 31(3):227–251.

Figueiredo, L. N. L., de Assis, G. T., and Ferreira, A. A. (2017). Derin: A data extraction method based on rendering information and n-gram. *Information Processing & Management*, 53(5):1120 – 1138.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370.

Guo, J., Crescenzi, V., Furche, T., Grasso, G., and Gottlob, G. (2019). Red: Redundancy-driven data extraction from result pages? In *The World Wide Web Conference*, WWW '19, pages 605–615, New York, NY, USA. ACM.

Lockard, C., Dong, X. L., Einolghozati, A., and Shiralkar, P. (2018). Ceres: Distantly supervised relation extraction from the semi-structured web. *Proc. VLDB Endow.*, 11(10):1084–1096.

Neumaier, S., Umbrich, J., Parreira, J. X., and Polleres, A. (2016). Multi-level semantic labelling of numerical values. In *The Semantic Web – ISWC 2016*, pages 428–445, Cham. Springer International Publishing.

Potvin, B. and Villemaire, R. (2019). Robust web data extraction based on unsupervised visual validation. In *Intelligent Information and Database Systems*, pages 77–89, Cham. Springer International Publishing.

Schulz, A., Lässig, J., and Gaedke, M. (2016). Practical web data extraction: Are we there yet? – a short survey. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 562–567.

Shi, S., Liu, C., Shen, Y., Yuan, C., and Huang, Y. (2015). Autorm: An effective approach for automatic web data record mining. *Knowledge-Based Systems*, 89:314 – 331.

Yuliana, O. Y. and Chang, C.-H. (2018). A novel alignment algorithm for effective web data extraction from singleton-item pages. *Applied Intelligence*, 48(11):4355–4370.

# Appendix D

# Software Tools

The following software tools have been implemented and used for the evaluation of the methods proposed in most of the published papers.

## D.1 CSSBox HTML Rendering Engine

**Project web site:** http://cssbox.sourceforge.net/
**License:** GNU Lesser General Public License v3.0 (LGPL)

CSSBox is an (X)HTML/CSS rendering engine written in Java. Its primary purpose is to provide a complete and further processable information about the rendered page contents and layout. The input of the rendering engine is the document DOM tree and a set of style sheets referenced from the document.

The output is an object-oriented model of the page layout. This model can be directly displayed but mainly, it is suitable for further processing by the layout analysis algorithms as for example the page segmentation or information extraction algorithms.

The availability of an explicitly represented and fine-grained model of the rendered is the main difference from the conventional rendering engines such as WebKit or Gecko.

CSSBox was used by several other researchers for evaluating their methods related to web document processing [53, 55, 77, 129]. CSSBox or its components are used by several third-party software products that include Atlassian JIRA (uses the CSS parser, which is a part of CSSBox), Vaadin Liferay Portal (its Tori component), Daisy pipeline (document transformations into accessible formats for people with print disabilities) or Atlassian Botocss tool (CSS transformations for e-mail). Currently (March 2020), GitHub reports 84 third-party repositories mentioning CSSBox in its dependencies.

## D.2 pdf2dom PDF parser

**Project web site:** http://cssbox.sourceforge.net/pdf2dom/
**License:** GNU Lesser General Public License v3.0 (LGPL)

Pdf2Dom is a PDF parser that converts the input documents to a HTML DOM representation. The obtained DOM tree may be then serialized to a HTML file or further processed. The inline CSS definitions contained in the resulting document are used for making the HTML page as similar as possible to the PDF input. A command-line utility for converting the PDF documents to HTML is included in the distribution package. Pdf2Dom may be

also used as an independent Java library with a standard DOM interface for DOM-based applications or as an alternative parser for the CSSBox rendering engine in order to add the PDF processing capability to CSSBox.

Currently (March 2020), GitHub reports 260 third-party repositories mentioning Pdf2Dom in its dependencies.

## D.3 FitLayout Framework for Page Analysis

**Project web site:** http://www.fit.vutbr.cz/~burgetr/FITLayout/
**License:** GNU Lesser General Public License v3.0 (LGPL)

FitLayout is an extensible web page segmentation and analysis framework written in Java. It defines a generic Java application interface for representing a rendered web page and its division to visual areas using the visual document model described in section 2.2. In the same time, the ontology-based representation of the document content and its storage in an RDF repository is supported.

The FitLayout framework was used for implementing all the designed page processing methods including the designed page segmentation methods from chapter 3, content classification based on visual features and logical structure discovery presented in chapter 4 as well as the structured record extraction discussed in chapter 5.

Finally, the framework also provides tools for controlling page analysis process and examining the obtained results through a graphical user interface as well as for the preparation and interactive annotation of training data sets.