Hierarchies in HTML Documents: Linking Text to Concepts

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

Abstract

For the successful setting of the Semantic Web, it is necessary to provide tools for linking the large amounts of data that are currently available in HTML documents to the Semantic Web ontologies. Due to the enormous variability of the HTML code, it is very limiting to define direct bindings between patterns of the HTML code and the concepts. We propose an approach based on modeling the visual part of the rendered document and describing the key characteristics of the data presentation in a general way. As a next step, we propose the way for using this model for locating the instances of the concepts in the document using the approximate tree matching algorithms and regular expressions.

1. Introduction

Despite the rapid development of the Semantic Web technologies, most documents available in the World Wide Web are still written in HTML. Considering the enormous amount of data potentially available in these documents, it seems to be very interesting to use the "legacy" Web as a data source for the upcoming Semantic Web. However, HTML is suitable for defining the visual appearance of the documents but it doesn't contain any means for the formal representation of the content semantics.

On the other hand, there are many documents in the Web that are primarily intended to simply present some structured data such as price lists, timetables, contact information, etc. Such documents are called *data intensive* [5] and typically, they are automatically generated from a back-end database. In this type of documents, the information is usually presented in a clear and structured form so that the user can find the desired information with a minimal effort. For this reason, the document usually contains a hierarchical navigational structure of headings and labels, that denote the meaning of each part of the text or data value, which allows the user to proceed from the most general general data (e.g. from a main heading that gives the basic idea about the topic of the document) the shortest way to the desired specific values. A model of this hierarchy is usually called a *logical document structure* [20]. Several approaches have been proposed for the discovery of the logical document structure in an HTML document [11, 13, 21, 17] or other types of documents [20].

The user gets the idea about the logical document structure by means of various visual cues – the document is visually split to several parts that can be nested, more important headings are written with larger font size, important words are highlighted etc. In contrast to the HTML code, where the authors of the documents are only limited by the capabilities of the web browser, the visual information must respect established rules that can be interpreted by the users. For example, the bold text is always considered as more important as well as the text written in large font.

On the contrary, the semantic web is based on data with exactly defined semantics. There must exist an ontology that describes the relations among the concepts that are being described. Concerning the text documents, the concepts can be generally divided to *lexical concepts* that have a direct string representation and *non-lexical* concepts such that correspond to non-lexical real world entities [7, 16]. When using the HTML documents as the data source for semantic web, the task is to locate the instances of the lexical concepts in the document while ignoring extraneous text. The first phase of this process is to locate the documents from the corresponding domain; i.e. the documents that correspond to a particular ontology. In this paper, we don't discuss this "information retrieval" phase and we assume that the documents contain the appropriate data.

Figure 1 shows a simple ontology for faculty personal pages. The solid rectangles present the non-lexical concepts while the dashed rectangles present the lexical ones. Let's assume that we want do locate the instances of the lexical concepts, i.e. the name, telephone, e-mail and affiliation of a person in an HTML document that presents a homepage of a person. Our approach is based on the definition of the relation between the logical structure of the document and the ontology.

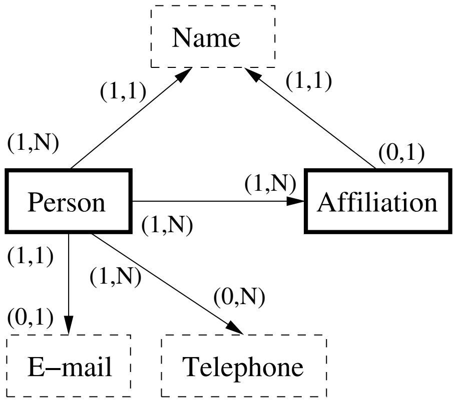

Figure 1. Sample ontology for personal pages

The principle of our approach is shown in Figure 2. As an input, a *presentation hierarchy* of the concepts must be specified. This hierarchy describes, how the concept instances are expected to be presented in the document. A possible presentation hierarchy for our example is shown in the figure 3. The name of the person is usually superior to the remaining concepts – it forms a title or a heading of the document. Then, the document is split to the personal information part (contact) and affiliation part.

The next steps involve the analysis of the visual infor-

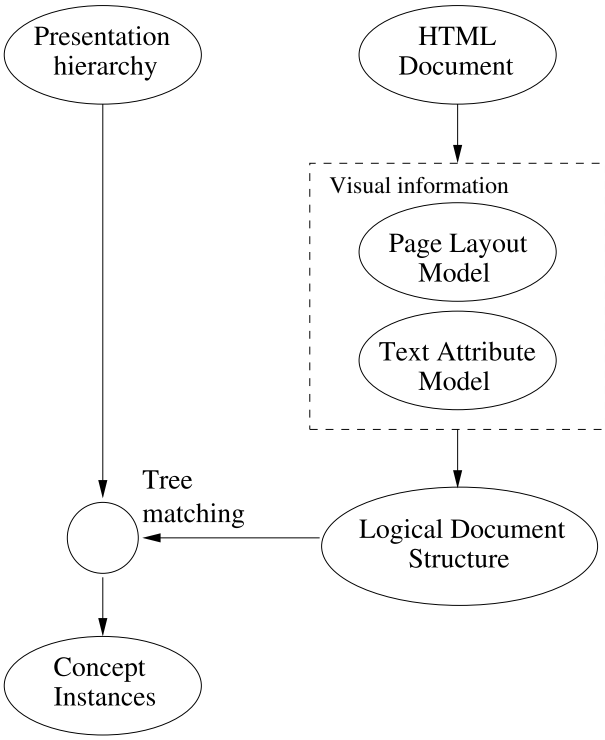

Figure 2. Method overview

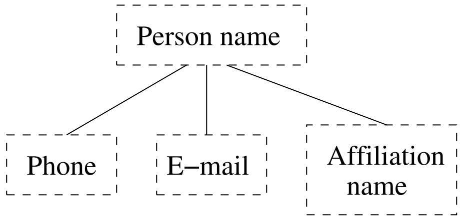

Figure 3. A presentation hierarchy

mation present in the document, modeling the logical document structure and locating the concept instances in the document. These steps are discussed in following sections. Section 2 gives an overview of the related work, sections 3 and 4 describe the process of the visual information modeling and the logical document structure discovery and section 5 discusses the use of tree matching algorithms for locating the instances of the lexical concepts in the document.

2. Related Work

Our approach is closely related to the area of *information extraction* from HTML documents. There have been several approaches developed recently, that are based on the construction of *wrappers*. All these approaches are based on an assumption that there can be found a relation between certain patterns of the HTML code or the text and the concepts. Since this assumption doesn't apply in general, it is necessary to find this relation for each individual class of documents. For this task, wrapper induction methods have been developed, that are mostly based on grammatical inference [5, 12, 14, 15] or inductive learning [6, 10, 19]. The bottleneck of these methods is the need of a training set of documents. The conceptual modeling approach has been proposed by Embley et al. for unformatted text data [7] and for extracting records from HTML [8].

The visual aspect of the documents is usually analyzed in order to obtain a model of the semantic structures used in the document and the relations among them. This is particularly important for processing the PostScript and PDF documents. Summers [20] introduces the notion of *logical document structure* defined as a hierarchy of segments of the documents, each of them corresponds to a visually distinguished semantic component of the document. Other authors use the notions of *document map*, *document structure tree* [13] and *logical schema* [3] in similar sense. In case of the HTML documents, the logical structure of the document can be discovered either by the analysis of the rendered document [11] or by the analysis of the document code [17, 21]. The mentioned approaches however usually model the logical structure to the level of text blocks. For our purpose, it is necessary to create more fine-grained model that contains the relations among the smallest visually distinguishable parts of the document.

The connection between the semantic and HTML documents is shown in following works: the HTML to XML document transformation is discussed in [4] while the lexical and non-lexical concept matching for the task of the integration of XML sources is discussed in [16]. Finally, the information extraction from the logical document structure is mostly inspired by Shasha's work [18].

3. Modeling the Visual Information

As results from Figure 2, the logical document structure is discovered by modeling and the analysis of the visual information that is available in the document. We can distinguish two components of the visual information: the page layout and the attributes of the text. The page layout is used to give the reader a basic idea about the document organization. Typically, the document consists of a main part that holds the information content and several additional parts that are visually separated by color or various separators (lines, boxes, etc.) The typographical attributes of the text give the reader more fine-grained visual information about the text. Individual parts of the text are usually distinguished by the font size or weight, underlining, different colors, etc. so the reader can distinguish the section heading from its content and the important information from the less important one.

The model of the page layout is based on *visual areas*. A visual area is a part of the document that can be potentially visually separated. Visual areas in a documents can be nested. The root area is always formed by the document itself and it can contain a hierarchy of subareas. Let's assign each area a visual identifier $v_i \in I$ where the root area has $v_0 = 0$ and the subareas have the identifier $v_{i+1} = v_i + 1$. Then, the model of the page layout can be represented as a tree of the area identifiers v_i

$$V = (v_0, S)$$

where v_0 is the root visual area (the document) and S is a set of subtrees directly under v_0. This model represents the information about how the visual areas are nested in the document.

For modeling the attributes of the text, we introduce a notion of *text element*, which is each part of the HTML code that is surrounded by HTML tags but it doesn't contain any HTML tags itself. A text element represents a part of the text that has constant values of all visual attributes, which can be modified with the HTML tags only. Thus, a text element can be regarded as the smallest visually distinguishable part of a document. The whole text of a document can be modeled as a string of the form

$$T = e_1 e_2 e_3 \dots e_n$$

where $e_i \in S \times I \times I \times I$ are text elements. We write e_i as a quadruple

$$e_i = (s_i, v_i, x_i, w_i)$$

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

The expressiveness x of an element indicates how much the element is highlighted in the document. This value is computed by a simple heuristic: the normal text with the default size that is not highlighted has $x = 0$. The expressiveness grows direct proportionally to the font size. Further, we increase the value of x by one when the text is bold, underlined or written in different color.

The weight w of an element expresses the superiority of the element in the logical document structure; i.e. the level of heading in the text. The normal text has the weight of 0 and the most important heading has the highest weight. The weight is computation is based on following heuristic: the heading nor a label may not lie inside of a block of text. Thus, the text elements that start a text block have the weight equal to the expressiveness. The elements inside of the text block have the weight of 0.

Both the model of visual areas and the text attribute model are build during one-pass parsing of the HTML code. In HTML, there are only few means that can be used for creating a visual area: tables, lists, paragraphs, frames, horizontal rules and the generic styled areas. The tree of area identifiers is constructed by maintaining a stack of open areas while processing the code. For creating the text elements, information about current text style is maintained and it is modified appropriately by interpreting the HTML tags and CSS styles. Each HTML tag encountered finishes current text element and the values of expressiveness and weight are computed from current text style. Each text data encountered start a new text element.

4. Modeling the Logical Document Structure

Considering the model of the visual areas V and the model of the text attributes T, the logical structure of a document is a tree of text elements

$$L = (e_0, T)$$

where e_0 is the root text element (the document title) and T is a set of subtrees directly under e_0. The leaf nodes of this tree are formed by the normal text of the document while the inner nodes are formed by the hierarchy of headings and labels in the document.

In our approach, this hierarchy is created in two steps. First, we create a *frame* of the logical structure which is a

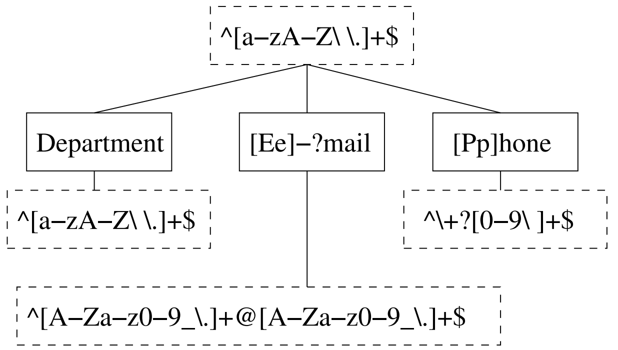

Figure 4. Sample query tree

tree of the text elements where e_i is an ancestor of e_j iff the corresponding visual area identifier v_i is an ancestor of v_j in V. In the second step, recursively for each node r in this tree we compare the weights of its child nodes c_1 to c_n and for any $c_i, i > 1$ we change the parent node of the corresponding subtree from r to $c_j, 0 < j < i$ when necessary so that the element of a lower weight is always a descendant of the element with a higher weight within the visual area.

The resulting tree models the logical document structure as expressed by the visual information in the document. Each node of the tree is a text element and the hierarchy of the elements respects the hierarchy of visual areas and the weight of the elements that results from the visual attributes of the text.

5. Tree Matching

The task of the tree matching algorithms is to locate the text elements in the logical document structure that correspond to the lexical concepts in the ontology. We use regular expressions for matching the text elements to concepts. Further, we assume that some of the values are labeled in the document which means that the text element containing the label appears as an ancestor of the appropriate text element in L. By defining the expected format of each lexical concept and its label using regular expressions we can transform the presentation hierarchy to a tree of regular expressions which can be handled as a structured query Q to the tree of the logical document structure.

Figure 4 shows a possible result of the transformation of the presentation hierarchy from Figure 3. The dashed rectangles correspond to the lexical concepts, the solid rectangles represent the expected labels. We are looking for all the subtrees of L that approximately match the query tree.

For tree matching, we use the **pathfix** algorithm for approximate searching in unordered trees [18] which is based on matching the root-to-leaf paths in the trees. For our purpose we have introduced following modifications:

1. The node from Q matches a node in L iff the appropriate text element matches the regular expression in the query node.
2. We allow n nodes in a query tree path that don't match any node from L (non-existing labels and data) and m nodes in L that can be skipped when they don't match any node from the query tree (extraneous data in the document).

This algorithm is used for comparing the query tree Q to all the subtrees in L where the root node matches the regular expression in the root node of Q. Each matching subtree found represents an instance of the presentation hierarchy and its text elements are the desired instances of the lexical concepts.

6. Conclusions

In this paper, we have proposed a new approach to linking parts of the text to concepts which is based on the analysis of the visual information in the document. In contrast to the methods based on direct HTML code analysis, our approach is independent on the way how HTML was used for achieving the desired presentation and allows using additional technologies such as Cascading Style Sheets (CSS). Moreover, the presentation hierarchy as the only input of the process can be defined intuitively without any requirements to the knowledge of HTML.

This method is however only usable for structured documents that contain sufficient amount of visual cues and where the important data forms separate text elements. The method is not suitable for documents, where the data appears in blocks of unformatted text. Further evaluation of the proposed method is on-going.

Acknowledgement This work has been supported by the long term grant project of Ministry of Education No. J22/98:262200012 "Research of information and control systems".

References

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

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

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

[4] C. Y. Chung, M. Gertz, N. Sundaresan. "Reverse Engineering for Web Data: From Visual to Semantic Structures",

18th International Conference on Data Engineering (ICDE 2002), IEEE Computer Society, 2002.

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

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

[7] D.W. Embley et al. "A conceptual-modeling approach to extracting data from the web", *Proceedings of the 17th International Conference on Conceptual Modeling (ER'98)*. Singapore, 1998

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

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

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

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

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

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

[14] R. Kosala et al. "Information Extraction in Structured Documents using Tree Automata Induction", *Principles of Data Mining and Knowledge Discovery, Proceedings of the 6th International Conference (PKDD-2002)*. 2002

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

[16] R. dos Santos Mello, C.A. Heuser. "A Bottom-Up Approach for Integration of XML Sources", *International Workshop on Information Integration on the Web*. 2001

[17] S. Mukherjee, G. Yang, W. Tan, I.V. Ramakrishnan. "Automatic Discovery of Semantic Structures in HTML documents", *International Conference on Document Analysis and Recognition (ICDAR)*. 2003

[18] D. Shasha, J.T.L Wang, H. Shan, K. Zhang. "ATreeGrep: Approximate Searching in Unordered Trees", *14th International Conference on Scientific and Statistical Database Management (SSDBM'02)* Edinburgh, Scotland, 2002

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

[20] K. Summers. "Toward a taxonomy of logical document structures", *Electronic Publishing and the Information Superhighway: Proceedings of the Dartmouth Institute for Advanced Graduate Studies (DAGS '95)*. Boston, USA, 1995

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