Analyzing Logical Structure of a Web Site

Radek Burget *
burgetr@fit.vutbr.cz

Abstract: The today's World Wide Web consists mainly of documents written in Hypertext Markup Language (HTML). This language has been developed for describing the look of the documents and the references to other documents and therefore it has very poor facilities for describing the semantics and the structure of the contained data. Moreover, some of these facilities are often not used by the authors of the documents or they are not used in apropriate way. In our work, we are attempting to analyze the look and the stucture of a Web site represented by the facilities of the HTML language and create its logical model which would represent the data relations the same way a human user would see it. We propose a tree representation of a Web site and algorithms for the analysis of the most importatnt HTML constructions – section headings, lists, tables and links.

Keywords: HTML analysis, Semi-structured data, Information extraction

1 Introduction

Although much effort has been dedicated to develop some data format which would allow better computer processing of the World Wide Web content, today's Web consists above all of a huge amount of documents written in Hypertext Markup Language (HTML). These documents contain alltogether a vast amount of data, but the used HTML language provides very poor facilities for describing the structure and semantics of this data. It only allows description of several basic constructions which are important for the text layout like headings, tables, lists (ordered or unordered) and above all links (i.e. references to other documents). Furthermore the authors of the Web pages often don't care about the semantics of particular HTML constructions. The reason is that the authors of Web pages are mostly interested in the final look of the page only.

On the other hand, most of the Web pages are designed to allow an easy orientation for a human reader. They usually have clear structure, they are split to sections and important data is highlighted and labeled so that an average user with a bit of experience is able to find desired information effectively. In other words, the Web page usually contains significant information about its structure but this information is not expressed by means of the HTML language. It is hidden in the look of the page: in the properties of the text (e.g. text font or color) and the layout of the text.

The knowledge of the logical structure of contained data can be beneficial for the information extraction from the Web. There exist several attempts to use HTML formatting tags in order to locate relevant information in the documents either using fixed heuristical rules [3] or using machine learning approach [6]. Another approach is to create a logical model of contained data rather than using the formatting tags directly. The *HTML Struct Tree* model [2] represents the data relationships within a Web page. It is a n-ary tree in

* Faculty of Information Technology, Brno University of Technology, Bozetechova 2, 612 66 Brno, Czech Republic

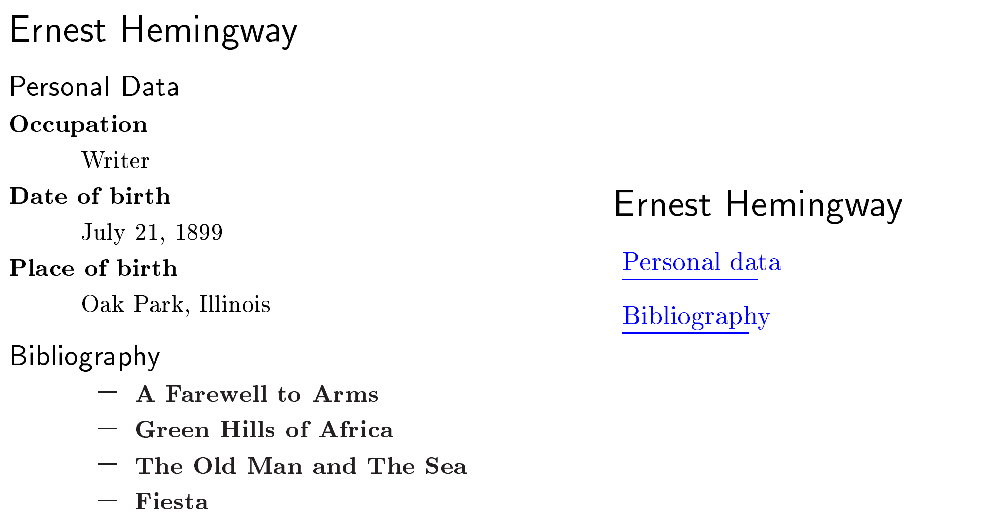
which each node corresponds to a HTML tag in the page code. The resulted model however depends on the way how the data is presented – which facilities of HTML language are used and how the data is split to Web pages.

In this paper, we propose a hierarchical model of a whole Web site that represents its logical structure and relations between contained data abstracting from the physical organization and we describe the process of HTML analysis in order to create this model.

2 Model of a Web Site

In this paper with the notion of *Web site*, we understand a Web page or the set of Web pages (single HTML documents) which concern one topic. For example, we can find many Web sites dedicated to software projects, company Web sites, personal Web sites, etc.

Figure 1 shows a web site dedicated to Ernest Hemingway in two different versions. The first site consists of one single Web page, that contains importatnt data about the famous writer. Naturaly, we can image another organization of this site. In this case, the site consists of three pages. The first page is the "main" page with links to the other pages, the second one contains personal information and the third one contains bibliographic data.



a) One-page organization   b) Multi-page organization

Fig. 1: Different ways of site organization

Although these two sites have different physical organization, the logical structure is same. On both examples we can see that the information on *Ernest Hemingway* is split to sections *Personal Data* and *Bibliography* and these sections have some internal structure. Let's represent this situation by a tree as shown in the Figure 2.

Let's consider the example in the Figure 1b with a little modification. Let's imagine that each of the pages *Personal Data* and *Bibliography* contains a link back to the main page. From the view of link structure analysis, this leads to graph representation of the site as shown in the Figure 3. However, from the view of the logical data structure the links back (the dashed lines in the figure) are not important.

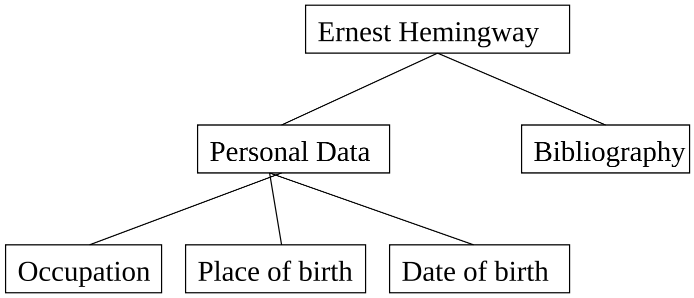


Fig. 2: Logical data structure

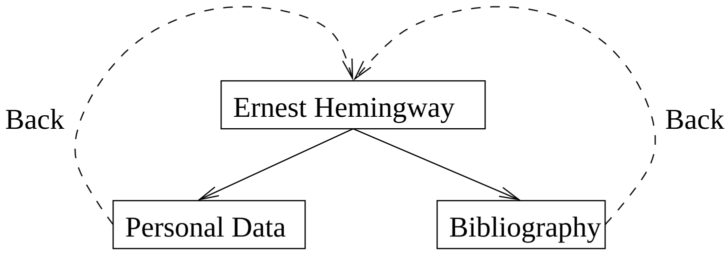


Fig. 3: Graph of link structure

We can see that we have to distinguish between the page references that are **significant** for the logical data structure and references that only serve for more effective navigation in the structure.

The process of the model creation consists of two major parts:

1. **Page-level analysis.** In this phase we analyze the code of individual pages. First, we analyze the structure of headings (HTML offers facilities for distinguishing six levels of headings). Secondly we analyze the HTML constructions contained in the page, especially **definition lists** and **tables**.
2. **Link structure analysis.** In this phase we decide which references contained in the documents are significant for the logical structure and we create a tree node for each of these links. Considering the example in the Figure 1b, we create the nodes *Personal Data* and *Bibliography* in this phase.

We start with the page-level analysis of the main page. During this analysis, we create top level nodes of the model tree (the node *Ernest Hemingway* in our example). If there was some node created that corresponds with some significant reference to another page, we repeat the analysis for the referenced page. This will produce a tree model of the referenced page which is added to the model of the whole site as a subtree of the corresponding link node.

Note that referenced pages can also contain significant links to other pages so that this algorithm is recursive. To avoid endless recursion in case of a cycle in the reference graph (e.g. the Figure 3), we have to ensure that each page is analyzed only once. For this reason we maintain a list of analyzed pages. Any link to a page that has been already analyzed, we consider as not significant. This corresponds with the nature of our logical structure model (the tree) where each node can only have one parent node.

We also have to define some bounds of the site. In other words, we have to decide which of the referenced pages belong to the analyzed site (we call them *local* pages) and which do not. We have used a simple method based on URL analysis. When the specified main page URL is of the form `http://host/prefix/file` we assume that all the pages that have the same values of `host` and `prefix` in their URL belong to the same Web site. Note that the `file` field can be empty. During the link structure analysis, we treat all the references to non-local pages as not significant.

For the purposes of information extraction, the tree model has an advantage. When we store for every tree node the data about where the corresponding HTML code is located (in which page and at what position in the document), it allows to localize a strictly bounded area in the HTML code where the desired information is located. These areas are bounded by the leaf nodes of the model tree. For example, the information of Ernest Hemingway's occupation is stored just after the document part that formed the tree node *Occupation* and this information ends at start of the next tree node or at the end of the document (if there were no more nodes found in this document).

3 HTML Code Representation

A HTML document [8] is basicaly a text file that consists of the text data which forms the contents of the document and special control sequences called *tags*. Because this form of the document is not suitable for the analysis, we had to develop some better representation.

For the document analysis we need to regard the HTML code from various views. For example, the reference to another page can be viewed either as a start point of a new subsection which is formed by another HTML page (when the reference is significant to document logical structure) or just as a simple text (ignoring the fact it is the reference).

We propose a hierarchical model of the HTML code that represents contained constructions simultaneously at different levels of abstraction. This model consists of **Page Elements**. A page element represents any HTML structure from the simple ones (parts of text, links) to complex ones (lists, tables). The elements are organized to a hierarchical tree structure. The root node of the tree represents the document as whole. Its child nodes describe the document at highest possible level of abstraction. Any of these nodes can be either a leaf node of the tree or it can be a root node of a subtree. In this case, the subtree contains page elements that can be used as an alternative representation of corresponding part of HTML code. For example, a page element that represents a table contains the contents of all the cells and additional information about the table. It describes the table as whole. As an alternative representation of this HTML code, the subtree of this node contains a set of nodes that describe the contents of individual cells of the table. During the code analysis we can decide, if this table is significant for the data structure or it's only used for achieving some text layout. In the first case we analyze the whole table, in the second case we treat it as a sequence of other structures.

We have defined following types of page elements:

Text Element. It represents a continuous part of the text with constant attributes. Actually, it's always a text between two nearest HTML tags. Each Text Element has a *content* (i.e. the text string itself) and *attributes* which carry the information about the text font and the heading level (if specified using the `<Hn>` tags). Text Element has no alternative representation and it always forms a leaf node of the code tree.

Link Element. This element represents a reference to another document that is defined using the `<A>` tag. It also has a *content* (the text of the link) and an attribute that contains the URL of referenced page. This element has always a subtree which describes the same part of the HTML code, ignoring the fact that this part is marked as a link. Usualy this subtree consists of a single Text Element.

Marker. This element only marks a HTML construction that has no additional attributes (a list, list item or heading). It always has a subtree that holds the contents of the structure. For example, the subtree of a list item Marker node contains the Text Elements or Link Elements that form the contents of the list item.

Table. It's a representation of a table and its contents. A subtree of this node always contains elements describing the table cell contents.

4 Analyzing the Link Structure

The task of link structure analysis is to find all the references (links) in a Web page that are important for the logical structure of the Web site. The references that are not significant are then treated as a simple text element.

In the first phase we exclude all the references that are not local in the Web site. If we have previously found a reference that is important for the logical data structure (in any previously analyzed page) we exclude all the references with the same destination. After this, we have a set of links that are possibly significant.

Our method is based on the fact that the structured Web sites usually contain somethink like a menu or table of contents – a set of links that lead to various sections of the Web site. We call such a structure a *guide-post*. We presume that all local references within the guide-post are significant.

For discovering a guide-posts within the page, we use regularity analysis. We create a vector of page elements using pre-order traversal of the page code tree. Let's mark a link element in this vector as `L`, text element as `T` and marker `M`. We represent the vector by a string and we search repeating string patterns which contain exactly one link element, e.g. `MLTTMLTTMLTT`... When we find a pattern with more than certain number of repetitions, we have found a guide-post.

5 Chapter Structure Analysis

The HTML language contains a facility for definition of chapter headings. There are six header levels: H1 to H6. During the chapter structure analysis, we add the tree of contained headers to our logical Web site model.

The headings are processed in that order as they are stated in the page code. We add the headings to the tree respecting their level – headings of the same level are at the same level of the subtree.

6 Table Analysis

The table is one of the most frequently used construction in the Web pages. A table can serve to present the tabular data but very often it is only used as a hidden instrument for achieving desired page layout. In the first case we would like to add the structure of the table to our logical Web site model. In the opposite case we ignore the table and we process its contents as separate page elements.

Radek Burget	
Faculty	Faculty of Information technology
E–mail	burgetr@fit.vutbr.cz

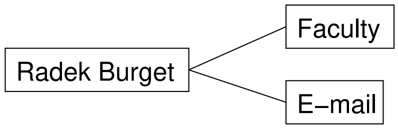


Fig. 4: A table and its tree representation

The structure of the table is determined by its headings. In the Figure 4 we can see a sample table and the structure of its headings.

HTML provides a way for distinguishing between header and non-header table cells. The authors of the documents however don't use this feature too often, they prefer highlighting the header cells with different background color or text style. In this case we have to estimate which table cells form the header.

When analyzing the table, as a first step we decide if the data is organized in rows or in columns. When no header cells are specified we expect that the header cells are in the first row or the first column and we try to prove this assupmtion by analyzing the cell styles. The header cells should have different style (text style, background, etc.) as the rest of the cells. When we find a header we analyze each column or row recursively as a separate subtable.

For example, when analyzing the table in the Figure 4, first we find the header cell *Radek Burget* because it covers the largest part of the table. We decide the table is organized in columns and it has one column. The we analyze contents of this column as a subtable. We find two header cells *Faculty* and *e-mail*. This subtable is thus organized in rows and we analyze the contents of each row. These rows both consist of one cell only thus there is no header and the analysis ends.

If we fail to find any header fields (marked in the code or estimated) in the whole table we say that the table has no structure and we treat it as individual page elements.

7 Other constructions

Another HTML construction that is very suitable for structure analysis are **Definition Lists** that explicitly define the relation between a term and its definition.

Other types of lists (ordered list and unordered list) provide very limited aid for the analysis process – we can distinguish individual items of the list but their internal structure is not defined.

8 Conclusions

Proposed algorithms for the Web site modelling are suitable for Web sites which contain some information about their structure expressed using the HTML facilities mentioned above. This is mostly the case of the sites which are primarily intended to give some information – official personal or company pages, project sites, automaticaly generated pages, etc. There is however a category of Web sites that have no structure even for a human reader.

We have implemented an experimental system using the algorithms stated above. The first test results show that this is a promising way for creating a logical model of a Web site. But the experiments have also shown many problems that have to be solved.

Firstly, the algorithm for the *guide-post* discovery based on regularity search fails when the HTML code contains some even minor irregularity (e.g. highlighted word). Improving this algorithm would significantly expand the set of Web sites for which the resulting model is valid.

Moreover, the HTML language is not limited to the facilities discussed in this paper. Some authors use alternative ways to define the page design. Typical example is ignoring the HTML header tags and using different font sizes instead. Another case is using user-defined cascade styles (CSS).

Finally, the lowest level of the structure is often not defined using the HTML language but it is formed by certain patterns in plain text such as like **Term: definition** (e.g. Position: assistant) and similar ones. This feature requires the analysis of individual text elements.

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".

Bibliography

1. George Chang, Marcus J. Healey, James A. M. McHugh, Jason T. L. Wang: *Mining the World Wide Web: an information search approach*, Kluwer Academic Publishers, 2001
2. Dan DiPasquo: *Using HTML Formatting to Aid in Natural Language Processing on the World Wide Web*, Senior Honors Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1998
3. Tao Guan, Kam Fai Wong: *KPS – a Web Information Mining Algorithm*, In the Eighth International World Wide Web Conference, Toronto, Canada, 1999
4. Paolo Atzeni, Giansalvatore Mecca, Paolo Merialdo: *Semistructured and Structured Data in the Web: Going Back and Forth.* In Proceedings of ACM SIGMOD Workshop on Management of Semi-structured Data (1997)
5. Zoé Lacroix, Arnaud Sahuguet, Raman Chandraskear: *Information Extraction and Database Techniques: A user-oriented approach to querying the Web.* In Advanced Information Systems Engineering, 10th International Conference CAiSE'98, Pisa (Italy), 1998
6. Dayne Freitag: *Information extraction from html: Application of a general learning approach.* In Proceedings of the Fifteenth Conference on Artificial Intelligence AAAI-98 (1998), 517–523
7. Serge Abiteboul: *Querying semi-structured data.* In Sixth International Conference on Data Base Theory (ICDT'97), Delphi (Greece), 1997
8. *HTML 4.01 specification*, The World Wide Web Consortium, 1999 `http://www.w3c.org/TR/html401`.