Extracting Logical Schema from the Web

Vincenza Carchiolo, Alessandro Longheu, Michele Malgeri
{ car, alongheu, mm } @ iit.unict.it
Dipartimento di Ingegneria Informatica e delle Telecomunicazioni
Facoltà di Ingegneria – Università di Catania
V.le A. Doria 6 – I95125 – Catania

Abstract. One of the main problems arising when facing with web data is the lack of explicit structure (schema), which presence may help in understanding semantics of information. In this paper we present an approach to extract logical schema of a web-site starting from page schemas. We first define a page model aiming to divide the contents into logical sections, i.e. parts of a page each of which collects related information. Then, we define a site model in which both physical and logical links between different page sections are represented; the former are existing hyperlinks, while the latter represent links between sections containing semantically related information. Finally, we show how such schemas can be used to improve both browsing and searching.

1. Introduction

One of the main problems arising when facing with web data is the lack of explicit structure, i.e. schema, whose presence may help in understanding semantics of information, improving the way users locate desired data, in contrast with actual browsing, in which data comprehension is mainly based on intuition [1]. Data are classified in structured and semi- or unstructured, being schema in the former case clearly distinct from data, while in the latter case it is contained within data [2], [3], [4], rising the need of "extracting" schema to improve data management [5], [6] (Web data falls in this latter case). Schema for web data can be constructed at different levels: giving a schema for a set of logically related sites; examining a single site, where the schema is generally a graph [7], or finally structuring web pages, introducing a level of describing information that is grainer than the page.
In this paper, we consider both web pages structuring as well as site schema detection. We first define a page model [8], aiming to divide the contents into "logical sections", i.e. parts of a page each collecting related information. To construct such sections, HTML source could be used to discover data structure, e.g. a paragraph (<*p*>) may be viewed as a logical unit of information; however, HTML has not been designed to clearly distinguish data structure from its representation, hence HTML source analysis should be used together with some semantic analysis (e.g. information retrievial techniques) to get to significant results. Our method combines both structural and semantics information about the page. The idea of dividing a page into sections is similar to those presented in [9], [10], [11]; however, such approaches are structure-oriented (i.e., they do not take into account semantics). The second step is to define a site model, in which we highlight both physical and logical links existing among different sections of site pages. While the former represents existing hyperlinks, the latter represent semantic links between distinct sections containing semantically

A.-H. Tan and P. Yu (Eds): PRICAI 2000 Workshop on Text and Web Mining, Melbourne, pp. 64-71, August 2000.

related information. Several models for web sites have been proposed [12], [13], [14], [7]; however, they do not explore semantic links across site's pages. The final goal of pages and sites structuring is the improvement of both searching and browsing, which can benefit from having an explicit structure, e.g. defining new query languages [4] or developing browsers where schema is used to facilitate data comprehension.
Our technique is mainly oriented to restructure existing sites; this means that: several version of HTML may be used across different pages, or documents can also contain non-standard tags, or finally there may be not present an uniform design of the site; in this sense, [12] propose a method for structuring sites when they follows specific design rules, while [15] shows a set of principles which should be taken into account when building a site from scratch. Here, the hypotesis we made is that HTML document be well-formed, i.e. all elements should have start and end tags, and tags nesting should be correct; if it is not the case, the document can be easily transformed into a well-formed one [8].
In section 2 we present logical sections, showing how pages can be divided into them. In section 3 we introduce a site model using page schemas, also investigating on how to use site schema for a better browsing and searching. Section 4 present conclusions and future work.

2. Modeling web pages

2.1 Logical sections

Generally, a web page contains several types of information, e.g. text data, hyperlinks, images, put by the author in a single page; though they do not form a monolithic block, rather it is possible to divide the page into a set of *logical sections* which aim to reflect the logical structure given by the author to that page.
The first section we define is the ***document information*** section, containing general information (metadata) concerning the document (e.g. document type specified in *<!doctype>*, referring DTD, author, etc.). This section is placed at the beginning of the document surrounded by the *<head>* tag.
Another section is the ***logical heading***, which includes general information used to present the page and/or site, e.g. the name, title or the logo of the site, followed by location and/or phone numbers, or mixed together with links to main services (Fig. 1). This section is generally placed near to the beginning of the document.

Fig. 1 - logical header sections

Similarly, we introduce a ***logical footer*** section, generally placed at the bottom of page and whose content (Fig. 2) can be a mail link to webmasters, or text with copyright, privacy, legal or advertising information.

The next section we introduce is the ***logical link***, a set of HTML tags representing a link towards another page. A logical link always include a physical one, through the *<a href...>* tag (Fig. 3 shows links using images and plain text). Several links properties can be defined [8], as internal/external (with respect to the site URL), granularity (general, as Yahoo [16] categories, or specific) and semantic correlation.

Fig. 2 - logical footer sections

Fig.3 - logical links

Next, we introduce the ***index*** section, a group of logical links having similar characteristics (internal/ external, granularity, correlation); for well designed sites, such links should also have similar graphical appearance (i.e. the same set of HTML tags). Properties mentioned above can be extended to index, computing a weighted average of values of its links.

Another section is the ***logical data***, where the semantic of the page is mainly placed, e.g. Microsoft CEO interview inside CNN [17] or game review built with tables in Fig. 4. The last section we introduce is the ***interactive***, i.e. forms where users interact with the page, such as usual search bar [16].

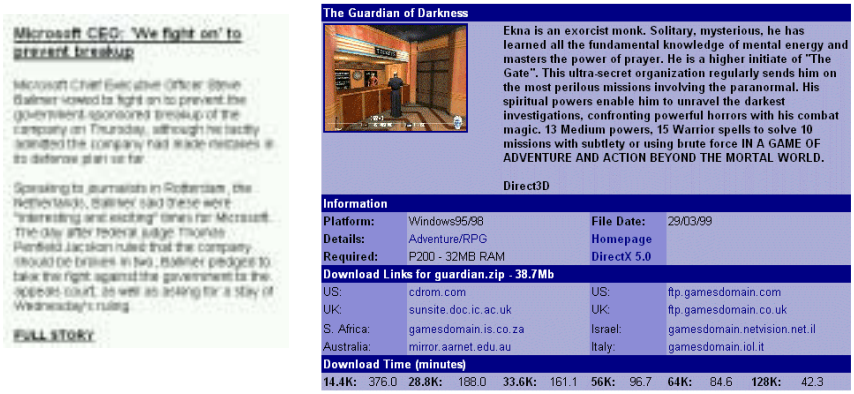

Fig. 4 - logical data sections

2.2 Structuring web pages.

Our first step in structuring a page is to model the page as a tree where nodes and edges reproduce the nesting of HTML tags of the document itself. Then, we locate specific tags which are supposed to have a primary role in structuring the page. We consider tags used to format text in paragraph, table and any similar graphic presentation, as *<p>* and *<table>* (they are implicitly used to create the logical

structure of the document); tags used to divide the page, as <*hr*> (usually separating portions of text having different semantics); tags used for frames and forms. Such considerations lead us to select the following html tags as primary nodes: *table, p, map, hr, a, frame, iframe, head, form, select, input, td, tr, option, area*. We then associate to each primary node a set of properties, used to structure the page:

- the *relative depth*, i.e. the number of primary nodes present in the path from root node (i.e. the <*html*> tag) to current node (relative depth starts from 0);
- the *absolute depth*, which is the number of nodes from root node to current node (it is the ordinary depth, starting from 0 as relative depth);
- the *number of levels*, i.e. the average absolute depth from current node to its leaves;
- finally, the *number of leaves*, considered just if the primary node *directly* contains leaves (i.e. no other nodes, primary or not, must be present between the primary node and its leaves, otherwise this parameter is set to 0).

After primary nodes and their properties are defined, we introduce the *collection*, a set of HTML tags containing repeated structures. For instance:

```
<p>...</p>
<a href...>...</a><br>
<a href...>...</a><br>
<a href...><img src..>...</a><br>
<a href...>...</a><br>
<a href...>...</a><p>
<hr>
```

Tab. 1

This collection represent a set of links; the idea is to detect such repeated structures using their structural *similarity*, in order to further discover logical sections (each made by one or more collections). We use the term similarity since, as shown in the example above, links are not represented with identical tags sequences. The idea of collection facilitates the structuring of a page, since it avoids to consider both each single tag, which is a too detailed level, or the entire page, which should be instead divided into more granular parts in order to detect its semantics.

Collections are built using primary nodes and their properties defined previously. Indeed, we first consider just repeated tags sequences starting with primary nodes, then we use primary nodes properties to evaluate similarity for such sequences, so we group them into a collection, creating a structural backbone of the page (note that, these sequences can be viewed are *subtrees* having primary nodes as *root*).

In our approach, we start from deepest primary nodes, comparing subtrees whose roots are primary nodes having the same primary node as ancestor (we refer to these subtrees as *brothers*). For instance:

```
(1)     <table>
(2)             <tr>
(3)                     <font size=+1>
(4)                     <td><strong>…<em>…</em>…</strong></td>
(5)                     </font>
(6)                     <td><strong>…</strong ></td>
(7)             </tr>
(8)             <hr size=4>
(9)             <tr>
(10)                    <td><em>…</em></td>
```

```
(11)                    <td><strong>…</strong></td>
(12)           </tr>
(13)    </table>
```

Tab. 2

In this tree, the sequence in row 4 is a subtree consisting of <*td*>, <*strong*> and <*em*> tags, with <*td*> as root (this tag is also one of the deepest, i.e. it does not include any other primary node); the sequence in row 6 has also a <*td*> as root (and it is also one of the deepest), and rows 4 and 6 are brother subtrees since their roots are children of the same <*tr*> tag. We note that a primary node is an ancestor for primary nodes even if there are other non-primary nodes placed between the ancestor and its children, e.g. in row 4 a <*font*> tag is present between <*tr*> and the first <*td*>.

Brother subtrees are then compared to evaluate their structural similarity. In order to do this, we adopt a vector model approach typical of information retrievial techniques [19], where distances between a query vector (containing weights for requested index terms) and documents vectors (each containing weights for index terms inside the document) are evaluated to establish the degree of relevance of each document with respect to the query. In our model, we use the same idea, first defining a vector for each primary node using the set of properties defined previously. Then, adopting a formula in which each vector component is properly weighted [20], we evaluate distances for weighted vectors, establishing if brother subtrees are similar; if so, they belong to the same collection, and the collection grows until other similar brother subtrees are found. When there are no more similar subtrees, a new collection is created, while when brother subtrees are all examined, the algorithm continues from the upper level of the tree.

This method tends to emulate the schema that intuition suggests: indeed, from Tab. 2, the subtree in row 6 should be considered similar to that in row 4, since they are the same tag (<*td*>), with the same attributes (in this example, both with no attributes), and children of the same <*tr*> tag, even if they have both different absolute depth and number of levels and leaves. However, the difference between absolute depth (4 vs 3), number of levels (2 vs 1) and number of leaves (0 vs 1) is not too high, hence they are considered to be similar (the formula takes into account all these considerations). Subtrees in row 4 and 6 are then grouped into the same collection. The algorythm then tries to add other subtrees, e.g. <*td*> tags in rows 10 and 11, but the subtree <*hr*> is placed between subtrees 4-6 and 10-11, and this should be interpreted as a separation, hence the collection stops; it will then consist of rows from 2 to 7, with the <*tr*> in row 2 as root. Further, the <*hr*> forms a separate collection, which will not be mapped into any logical section, since it contains no text (i.e. no data), and it is used just as a separator. The algorythm finally groups tags of rows from 9 to 12 into a single collection, using the same criterias which lead to the collection 2 - 7.

Once primary nodes are located and collections are constructed around them to detect the structural skeleton of the page, collections can be mapped to logical sections.

The ***document information section*** comes form the (unique) collection rooted by the <*head*> tag (this collection indeed contain all metadata about the document).

The ***header section*** is associated to the collection containing information related to those present in the previous collection (e.g. inside <*title*>) or to the page URL. To locate header section, all collections must be indexed [19] so their semantics (i.e. keywords) can be extracted and analyzed in order to find which collections satisfy the

previous criteria. If more collections are candidate to be header, the nearest to the beginning of the page is chosen.
The ***footer section*** generally contains links to some e-mail (e.g. the webmaster), or information about copyright, privacy, legal terms; it can also contain the same information of the header. Hence, even in this case, collections semantics has to be analyzed. If more sections are candidate, the nearest to the end of the page is chosen.
Index sections could be located simply searching collections containing *<a href>* tags, but we could have a data section containing text with some link inside. On the other hand, we could look for *<a href>* tags with no text inside, but we could also have an index section with each link described by some text. In order to distinguish such situations, we compare the number of words contained in text portions with the number of links, in order to classify such collection as data or index section. An index section can be splitted e.g. in order to separate all external from internal links or to create index sections differing for the granularity.
A ***logical data section*** can be associated to collection containing text, but as shown before, a more accurate analysis must be performed to check first if the collection should be eventually interpreted as a different section [20].
Finally, ***Interactive sections*** comes from collections containing *<form>*, *<input>*, *<select>* tags.

3. Site model

To model the site, we consider graph models [21]. The main difference with respect to other proposals [12], [13], [14], [7] is the use of logical sections as base element which allows a finer granularity in representing the structure of information inside a site. The proposed model aims to create *logical* links that connect semantically related sections across different pages. To do this, each logical section is first indexed [19], and then associated to an inverted file, which contains keywords of the text of a section, together with their position and frequency. Inverted files are then compared, looking for matches between keywords, thus evaluating semantic correlation between corresponding sections. Matches may have a different degree (e.g. the keywords may be the same or simply synonyms), so a thresold may be chosen to set the minimal correlation for a match. Then, in order to allow comparisons between sections having a different number of keywords, we normalize the number of match with respect to the number of keywords, so correlation degree is $C = N_{match} / N_{terms}$. Note that this formula does not take into account type of section being compared, which is also an important parameter: indeed, if two index sections have a high correlation degree, the logical link may be considered as no meaningful since they probably carry the same information, while if two data sections across different pages have a high correlation degree, users will find it useful to have them connected.
The result of the analysis on correlation between sections is a map of the site represented as a graph with logical sections as nodes, and where both physical and logical links are represented (physical links are existing links between pages). In order to allow users to interact with this schema, a structure-aware browser may presents two frames inside browsing area (Fig. 5).
The former (placed on the left) shows the graph, while the latter is the area where

pages are shown. In this way, an user can navigate the page either viewing it as usual or selecting a specific section from the graph (it will be highlighted on right frame). Moreover, it is also possible to move across pages by exploiting physical and semantic links (e.g. the latter could be useful when the user wants to follow a concept throughout different related pages). Finally, browsing can be further improved by giving some condition on sections (or their properties), in order to have a semantically customized browser, e.g. a user can define a template for viewing just specific sections of pages (for instance, avoiding external unrelated indexes). Also searching could be improved, e.g. defining query languages allowing to use schema information.

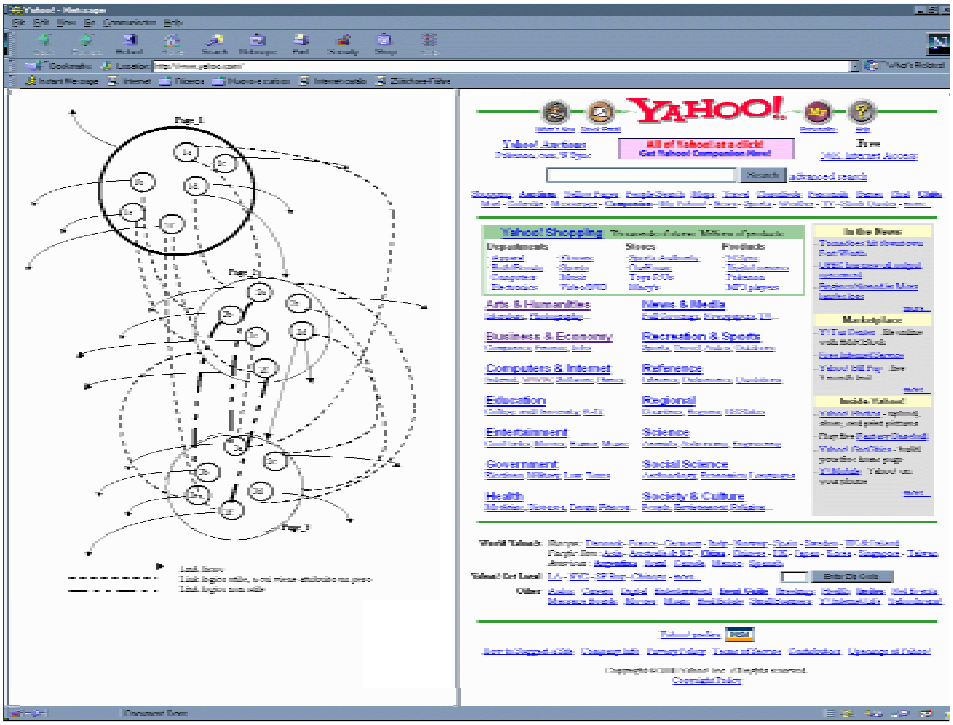

Fig. 5 - Site map inside a structure-aware browser

4. Conclusions and future work

This paper presents an approach to extract the logical structure of a web-site starting from the content of its pages. The approach starts modeling the page according to its physical structure and the meaning of data it contains. The page is then divided into collections, which represent a structural schema of the page, and subsequently collections are mapped into logical sections, where semantic is given to each collection. Next, a model of the site is presented. It uses logical sections to create a map of the site showing both physical links (hyperlinks) and logical links discovered across different sections semantically related. Such map is represented as a graph,

which can be used inside structure-aware browsers, improving the way information is accessed through the web.
Further work is needed to extend the model in order to take into account also scripts, using, for instance, DOM [18]. We are also investigating on how to use XML capabilities for web structuring [22], [23], [24]. We are also considering *user profile* (which stores user actions and preferences) to use schema to automatically provide users with filtered pages according to their preferences. Moreover, *page profile* can be defined to create one schema for similar pages, e.g. uniforming different schemas for pages belonging to sites in the same area of interest. Finally, further analysis is needed in order to significantly improve browsing and searching.

References

[1] P.M.G. Apers, *Identifying internet-related database reasearch*, 2nd Intl. East-West Database Workshop, 1994.
[2] P.Buneman, *Semistructured data*, Workshop on Management of Semistructured Data, 1997.
[3] S. Abiteboul, *Querying Semi-structured Data*, Proc. of ICDT, 1997.
[4] S. Abiteboul et al., *Data on the Web*, Morgan Kaufmann, 2000.
[5] S. Nestrorov et al., *Extracting schema from semistructured data*, Proc. of ACM SIGMOD, 1998.
[6] G. Huck et al., *Jedi: extracting and synthesizing information form the web*, Proc of 3rd IFCIS Intl CoopIS, 1998.
[7] H.G. Molina et al., *The TSIMMIS project: integration of heterogeneus information sources*, Proc. of the processing society of japan, 1997.
[8] A. Longheu, V. Carchiolo, M. Malgeri, *Structuring the web*, Accepted for Takma 2000 workshop http://www.dexa.org
[9] B. Adelberg, *NoDoSe*: *A tool for semi-automatically extracting structured and semistructured data from text documents*, Proc. of ACM SIGMOD, 1998.
[10] J. Hammer et al., *Extracting semistructured information from the web*, Workshop on Management of semistr. data, 1997.
[11] D. Smith, M. Lopez, *Information Extraction for semi-structured documents*, Proc. of Workshop on management of Semistructured data, 1997.
[12] P.Atzeni et al., *To weave the web*, Proc. of the 23rd VLDB Conference, 1997
[13] P. Fernandez et al., *Catching the Boat with Strudel: Experiences with a Web-Site Management System*
[14] P. Fraternali, Autoweb – http://www.elet.polimi.it/users/dei/sections/compeng /piero.fraternali/autoweb/
[15] S. Ceri et al., *Design Principles for Data-intensive Web Sites* – Proc. Of ACM SIGMOD, 1999
[16] Yahoo!, *http://www.yahoo.com*
[17] CNN, *http://www.cnn.com*
[18] Document Object Model , *http://www.w3.org/DOM*
[19] R. Baeza-Yates, B. Ribeiro-Neto, *Modern Information Retrievial*, ACM Press, 1999
[20] C. Parisi, A. Longheu, *Ristrutturazione dei siti web: un modello semantico per l'accesso alle informazioni*, Tech Internal Report No. DIIT00/Ah74
[21] Y. Maarek et al., *Webcutter: a system for dynamic and tailorable site mapping*, proc. Of 6th WWW conference, 1997
[22] RDF Recommendation, *http://www.w3.org/TR/REC-rdf-syntax*
[23] XML Namespaces, *http://www.w3.org/TR/REC-xml-names*
[24] XML Schemas, *http://www.w3.org/XML/Schema.html*