MODELING WEB DOCUMENTS AS OBJECTS FOR AUTOMATIC WEB CONTENT EXTRACTION
Object-oriented Web Data Model

Estella Annoni
IRIT, University of Toulouse, 118 Route de Narbonne, F-31062 Toulouse CEDEX 9, France
annoni@irit.fr, http://www.irit.fr/~Estella.Annoni

C. I. Ezeife*
School of Computer Science, University of Windsor, Windsor, Ontario N9B 3P4, Canada
cezeife@uwindsor.ca, http://cs.uwindsor.ca/~cezeife

Keywords: Web data model, Object-oriented mining, Automatic web data extraction.

Abstract: Traditionally, mining web page contents involves modeling their contents to discover the underlying knowledge. Data extraction proposals represent web data in a formal structure such as database structures specific to application domains. Those models fail to catch the full diversity of web data structures which can be composed of different types of contents, and can be also unstructured. In fact, with these proposals, it is not possible to focus on a given type of contents, to work on data of different structures and to mine on data of different application domains as required to mine efficiently a given content type or web documents from different domains. On top of that, since web pages are designed to be understood by users, this paper considers modeling of web document presentations expressed through HTML tag attributes as useful for an efficient web content mining. Hence, this paper provides a general framework composed of an object-oriented web data model based on HTML tags and algorithms for web content and web presentation object extraction from any given web document. From the HTML code of a web document, web objects are extracted for mining, regardless of the domain.

1 INTRODUCTION

Since, web document volume and diversity are tremendously increasing (Kosala and Blockeel, 2000), and more businesses use full web applications, web data analysis and web content mining have become important research areas. In web content mining, presenting web data in a formal structure and extracting similar records in web documents are two important tasks. Web documents generated by back-end database systems have underlying data structures where data presented on a web document has the same set of attributes, such as the Chapters web page in figure 1 which has three nested tables. Web documents manually generated are loosely-structured, even unstructured because there is no similar data structure underlying in them, such as United Nations index web page (UN, 2007). The upper part of Chapters web document (figure 1) composed of unstructured (Chapters logo) and loosely-structured complex data (Chapters menu) is not handled by previous approaches because either they view it as noise or they do not manage it. Existing work on web data extraction mostly handle only complex and structured web data. However, many of web documents are unstructured as argued by (Kosala and Blockeel, 2000). We aim at mining any web document in a unified way, so our data extraction process needs to handle simple, complex, loosely structured and unstructured web data. For example, answering the following query, "What is Chapters fake website profile?" entails mining structured data related to items in sales and also unstructured data about chapters company. So, preventing Chapters'fake business websites requires flagging those presenting records of items and composed of Chapters logo and a menu with user account in order to drag common points and define their profile.

*This research was supported by the Natural Science and Engineering Research Council (NSERC) of Canada under an operating grant (OGP-0194134) and a University of Windsor grant.

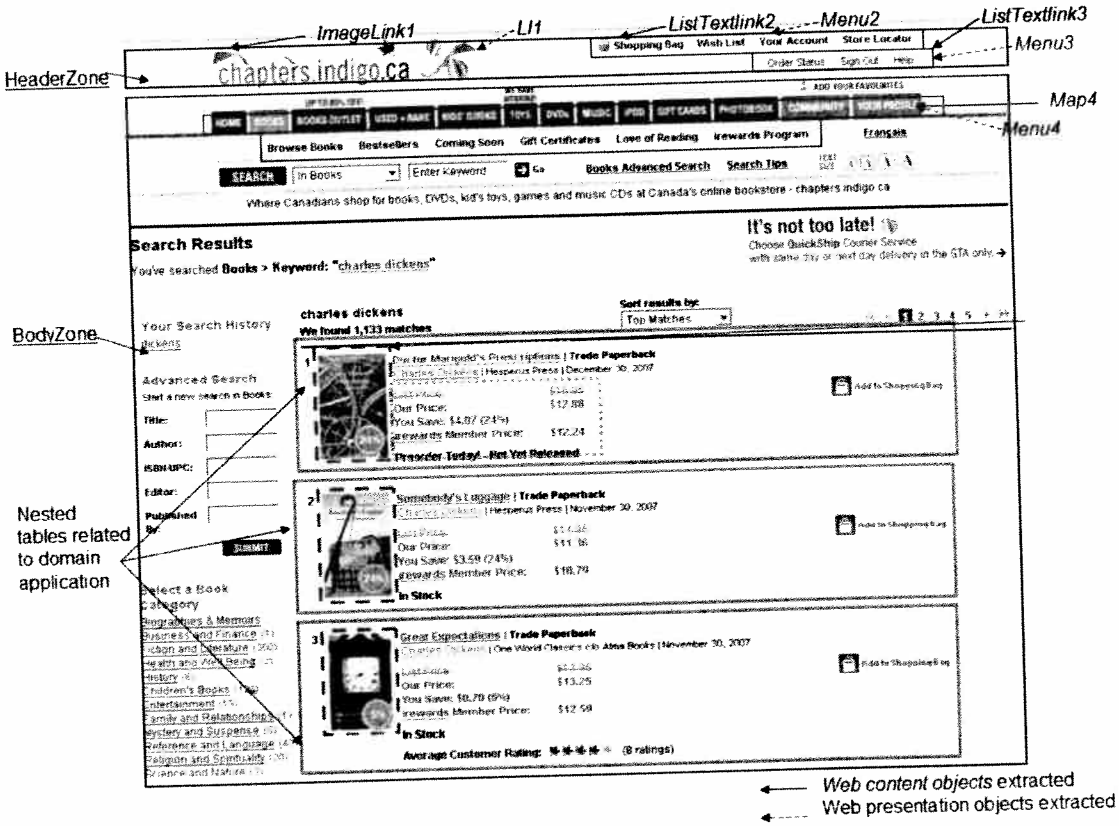

Figure 1: Web document with records in body part and unstructured data in its upper part.

1.1 Related Work

The two main structures used to model structured content are Object Exchange Model (OEM) (Abiteboul, 1997) and nested table (Liu et al., 2003). OEM represents data as a graph where objects such as books, book author. The ids of these objects are the vertices and the object string labels are the edges. The objects consist of a set of reference pairs (label, object id). Although web data are related as argued by (Liu and Chen-Chuan-Chang, 2004), none of the two main structures covers all relationships. In fact, both nested tables and OEM represent aggregation relationships, but only OEM represents reference relationships. Both of them do not represent hierarchical relationships between objects, which are also useful in order to reuse and extend mining techniques through objects. Nested tables do not support loose structures and both do not handle unstructured data because they look only for similarities through data. Hence, they are not adequate for catching the full diversity of structures within web documents (Crescenzi et al., 2001).

With both of these approaches mining a web document corresponds to either going through all the OEM graph or mining all nested tables within it. Mining web documents of different domains is not possible because the nested tables or OEM associated are comparable to each others, which is not efficient.

To mine data on structured web documents, their HTML files are mainly represented as pre-parsed documents with DOM tree by (Liu et al., 2003; Gottlob and Koch, 2004; Zhao et al., 2005) contrary to (Abiteboul, 1997; Crescenzi et al., 2001; Arasu et al., 2003; W3C, 2007) assuming them as sequences of character strings. In a DOM tree, tag and tag attributes are represented with nodes, and the nodes respect hierarchical relationships between tags. Although, users mainly refer to web data presentation to select information on web documents as argued in web segmentation work (Levering and Cutler, 2006; Li and Ezeife, 2006), none of pre-parsed based research assumes presentation cues conveyed through HTML tags, such as attribute tags "display" as our approach does. Hence, considering only content types search spaces are partially narrowed. The authors of (Zhao et al., 2005) use visual analysis of content shapes on web documents and compare with web document DOM tree, but their work does not take advantage of presentation aspects conveyed directly through the DOM tree. They consider only strict structures and assume stable data presentations since web documents handled are generated by search engines.

1.2 Contributions and Outline

Basically, mining data approaches from web documents face three main problems:
(1) Inability to focus web search on either web document presentation or content, or both.
(2) Lack of a unified framework to mine each web object regardless of their structure type (unstructured, loosely or strictly structured).
(3) Time consuming and domain-dependent because for requirements based on several domain applications, the mining process should submit as many process as number of domain applications.
This paper provides an object-oriented web data model for representing web data as web content and web presentation objects to address these problems and mine complex and structured data as well as simple and unstructured data in a unified way. Hence, our object-oriented web data model distinguishes content from presentation aspects of data (title, label, image, ...). We also define algorithms to extract these objects from any given web document of any web application in different domains.

This paper is organized in four sections. Section 2 presents the proposed object-oriented web data model which represents the first general and complete web content and presentation type classification. In section 3, algorithms for extracting web content and presentation objects are presented. An example application of these algorithms on Chapters web document (figure 1), on its structured and unstructured data is also presented in this section. Conclusions and future work are given in section 4.

2 OBJECT-ORIENTED WEB DATA MODEL

Web document segmentation work (Yu et al., 2003; Song et al., 2004) uses DOM tree, data location features, and data presentation features to distinguish information blocks from noisy ones. We need to go further through the concept of web document block to address data of simple and complex type and define web document objects. Like web segmentation algorithms of (Yu et al., 2003; Song et al., 2004), we do not evaluate all HTML tags because that is time consuming and all HTML tags are not always meaningful. For example, <h1> or <a> tags are more meaningful than <pre> used for pre-formatted text.

Thus, we propose an object-oriented web data meta model, which captures both the content and presentation views of web documents, to mine a web document as a set of objects. Our web object classes are defined based on the four concepts:
(1) Main HTML tags: non-empty tags which impact web document contents and presentations such as title, table, link, form or list tags.
(2) Location features: web users directly distinguish the meaning and the interest of data in web documents with respect to where information is located. Therefore, we pay attention to location features such as web document zone, width, height, and center.
(3) Presentation features: information which are instinctively used by human beings to distinguish contents such as style, fonts, and spaces. An important visual cue in the DOM tree is depicted by the attribute tags called "style" or "type". Moreover, the value of the feature named "display" must not be "none" or "hidden" to be considered as a rendering content.
(4) Relationships between objects: web document objects are related to each other and they share at least the same space of presentation.
This metamodel represents a web document with an object of the same name as a composition of WebZone objects because a WebZone object can appear only on one web document at a time. A **WebZone** object is a coherent zone in a web document. In a web zone, one can notice contents and presentation of these contents. A **WebElement** object is a content such as text, picture, forms, plug-in, separator, and structure related to the content view. An example of WebElement object is a weather plug-ins on personal home pages which come from the same website. A **WebRender** object is the rendering of contents on a web document. An example of WebRender is the menu shared by different web pages of the same website or the legal information announced in the lowest part of every web document of a website. From the content view, a WebZone object is a composition of WebElement objects because when a WebZone is destroyed, WebElement objects associated to it will be destroyed. For the same reason, from the presentation view, a WebZone object is a composition of WebRender objects.

In the literature, three main zones at most are visually considered on a web document regardless of the number of main subtrees of its DOM tree. These zones are called header part, body part, and foot part (Song et al., 2004). We define these parts as instances of specialized WebZone classes: HeaderZone, BodyZone, and FootZone. HeaderZone and BodyZone of our snapshot are sketched on the left part of figure 1. Now, we have presented the zones, we are going to present the objects which may exist in these zones in terms of content and presentation views.

2.1 Web Content Objects

Examples of web content types cited in existing work include text, image, metadata, video, but a complete classification of these contents does not exist. Our classification is a tree-based structure defined from sets of tags at the same level in the DOM tree, attributes of these tags, and their enclosed tags. (Levering and Cutler, 2006) classify web content according to four web content types (Image, Text, Form, Plug-in). We consider them as the basis of our classification.

Web documents shows blanks, more generally separators between the four aforementioned types. So, we add a fifth content type called SeparatorElement, which is an element separating content types from each other, so that any web application such as segmentation could be supported. Moreover, web documents generated by back-end database systems respect an underlying data structure, which is an aggregation of simple data types. A sixth content type which is an aggregation of the five previous ones is also defined to model structured data. Thus, we define sub-content types until we find the most specialized, simple content type existing on the web documents. The feature of anchorage of text and image contents is also used to take advantage of the property of navigation through web pages. Our complete classification of web content types defines six main generic types, among them four have sub-content types:

1. Text element: textual data containing or not containing page setup and it can be of two kinds:
(a) Raw text: textual data without predefined page setup and enclosed in HTML tag <span> and is of three kinds:
i. Title: textual data which is header in page setup and enclosed within HTML tags <h1> to <h6>.
ii. Label: textual data with page setup enclosed in HTML tag <label> .
iii. Paragraph: textual data with page setup defined by the enclosing HTML tag <p>.
(b) List text: textual data organized into lists and enclosed within HTML tag <li>. It has two kinds:
i. Ordered list: enclosed within HTML tag <ol>.
ii. Definition list: enclosed within an HTML tag <dl>.
2. Image element: a picture which can be of two kinds:
(a) Image: a simple picture enclosed within an HTML tag <img> which does not contain a <map> attribute.
(b) Map: a picture associated to a mapping defined by an HTML tag <img> with a <map> attribute set up (in case of a client-side mapping) or a <ismap> attribute (in case of a server-side mapping).
3. Form element: box of textual data that web users send to web servers. It could be enclosed within HTML tags <form> or <fieldset>. The <fieldset> tag allows authors to semantically group form entries. It has three kinds:
(a) Form select: box of textual data submitted by web users to web servers from a unique or multiple choice. It could be enclosed within HTML tags <form> and <select>.
(b) Form input: box of textual data that web users send to web servers order. It could be enclosed within HTML tags <form> and <input>.
(c) Form text area: box of textual data delivered to web users. It could be enclosed within HTML tags <form> and <textarea>.
4. Plug-in element: data launched by specific applications in order to provide web users with a richer interface. It has two kinds:
(a) Plug-in server: data launched onto web servers. It could be enclosed within HTML tag < −−#command exec="scriptName"−− > for CGI code, < % program % > for VB code, and <?*php* program ? > for Php code, ...
(b) Plug-in client: data launched onto the local client. It could be enclosed within HTML tags <script> and <object>.
5. Separator element: spaces between contents which emphasize them and make them instinctively meaningful for human beings such as line, blank, and empty space. They could be enclosed within HTML tags <hr> ,
.
6. Structure element: aggregation of the previous elements of simple type in order to represent web data of complex type adapted to represent web data structure such as structures composed of different content types. It is defined as a sub-tree of a node in the DOM which has three or more children sub-tree root of height 3. The string value of these tags has to be the same.

Text and images which are hyperlinks, references to web document parts, are represented as implementing an interface called "Link". Structured data type is modeled by an aggregation of WebElement objects as a self-aggregation relationship.

2.2 Web Presentation Objects

The authors of (Yu et al., 2003) mention that web documents are composed of four noisy materials, called: 1. navigation (e.g., hyperlink), 2. decoration (e.g.,

pictures), 3. interaction (e.g., forms), 4. special paragraphs (e.g., copyright) and one information material, called 5. topic (e.g., main data). These materials are defined mainly according to properties which graphically grab users' attention. These presentation properties have only one strict meaning and are specifically defined (for example images, hyperlinks, copyrights) contrary to textual contents. In this section, we make explicit this classification in order to automatically recognize these objects and we complete it to represent structured data. So, we indicate HTML tags according to the presentation object data type and we add a data type called "Record" to model structured data. The UML class diagram of presentation objects is composed of six specialized classes which are:
1. Banner: dynamic WebRender object that contains information plug-in, images linked to external websites, usually situated around edges, in the upper part. These objects could be presentation view of text, image, and plug-in contents. Thus, the HTML tags associated are <span>, <h1> to <h6>,<label>, <p>, <li>, <img>, <form>, <fielset>, < −−#command exec="scriptName"−− >, < % program % >, and <?*php* program ? >, <script>, <object>.
2. Menu: WebRender object that organizes navigation in web documents of a website or through a web document. These objects could be the presentation of text and image content. These objects do not contain keywords, such as "copyright", "about our company". Thus, the HTML tags associated are <span>, <h1> to <h6>,<label>, <p>, <li>, <img>.
3. Interaction: WebRender object that collects user information for service. These objects could be the presentation view of forms. Thus, the HTML tags associated are <form>, <fielset>.
4. LegalInformation: non-dynamic WebRender object that contains text or image related to authors, website or references. It is defined through text and a series of <a> tags referencing web documents of the same website. It might be defined by simple text containing keywords "copyright", "private", "policy", "about our company", or it can be defined through a series of <img> tags and <a> enclosing <img> tags.
5. Record: WebRender object that represents structured data related to web document interests. These objects could be structured element rendering. Thus, the HTML tags associated are a subset of the aforementioned tags such as a sub-tree of a DOM tree node which has three or more children sub-tree root of height 3. The string value of these tags have to be the same.
6. Bulk : WebRender object that contains unstructured and loosely-structured data which are not a banner, a menu, an interaction, a legalInformation or a record.
Regarding the content view and the presentation view, two hierarchies of web objects have been defined based on HTML tags.

3 AUTOMATIC WEB OBJECT EXTRACTION

Web documents are from two main types : unstructured and structured documents. They are evaluated by users in terms of contents and rendering of these contents. Using web document DOM tree and the hierarchies aforementioned, web objects can be automatically extracted. By this way, whatever the application domain, any web document can be modeled as a set of objects and tasks of pattern discovery through intra and inter web documents can be processed to detect similarities and trends. The main algorithm called OWebMiner (Object-oriented Web Miner of Content and Presentation Objects) is defined as follows :

Algorithm OWebMiner()
Input: a set of HTML files (WDHTMLFile)of web documents.
Output: a set of patterns of objects.
begin
for each WDHTMLFile
  (A) Extract web presentation objects and web content objects are sequentially extracted with respect to their hierarchical dependencies.
  (B) Store the web object hierarchies into a database table.
endFor
Mine patterns lying within objects
end.

In this paper, we develop the sub-algorithm (A). Given a web document HTML file, this sub-algorithm works in three steps. In the first step, a DOM tree of the HTML file is generated from the HTML code source requires a DOM parser (W3C, 2007). The DOM tree obtained from the HTML file of Chapters web document (Chapters.ca, 2007) is shown in figure 2. In the second step, from the DOM tree, web zones on the web document are identified. Then, during the third step, for each zone, web presentation objects and web content objects are sequentially extracted with respect to their hierarchical dependencies. We use blocks'levels to extract objects from the DOM tree and we consider both block-level tag nodes and non block-level tag nodes. A **block-level** tag is a tag that can only be child of the sub-tree "body" or another block-level tag and should be a parent of other

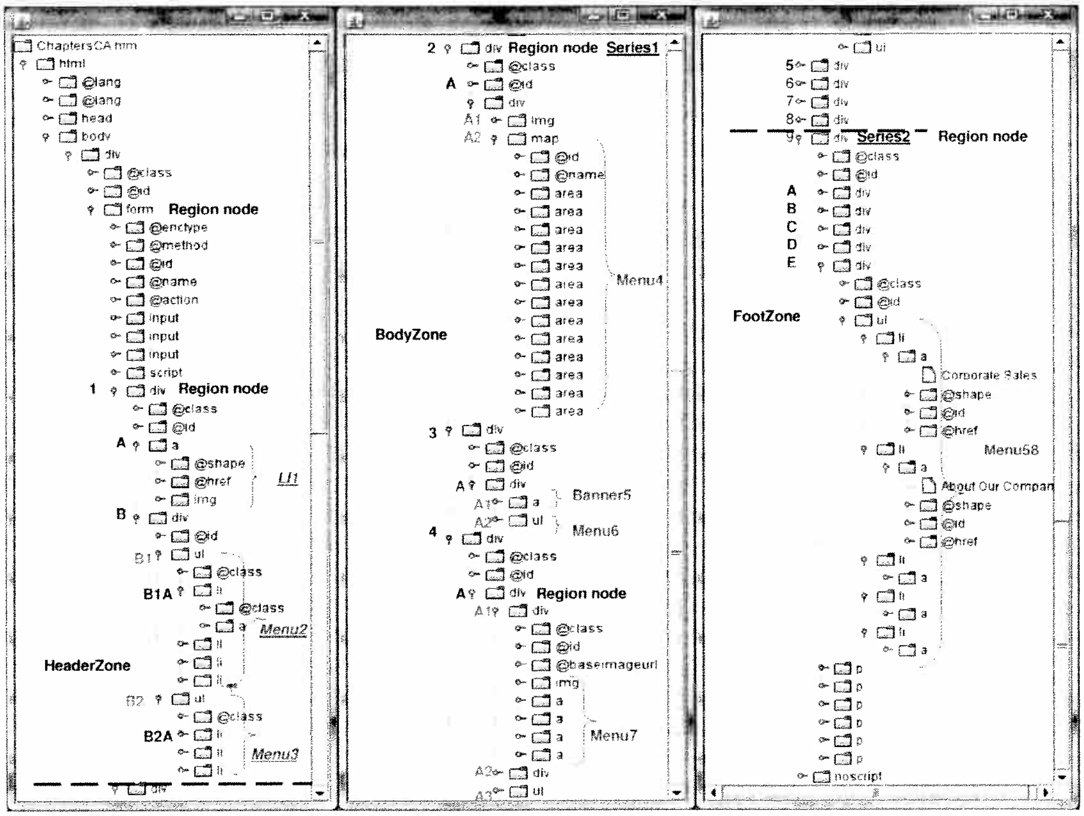

Figure 2: DOM tree of web document on figure 1.

tags, like table, division, heading, list, form, block quotation, paragraph, and address. A **non block-level** tag is an inline tag or text-level tag which is a child of block-level tag and is mainly a DOM tree leaf. For example, non block-level tags are anchor, citation, image, object, span, script. Since block-level tags mainly structure web documents and non-block level enclose web document content or presentation, two search approaches are used to explore the DOM tree. In fact, the extraction process follows a depth-first search through block-level tags until finding a non block-level tag. But, it follows a breadth-first search when a non block-level tag is parsed, its sibling set are evaluated to associate web content objects and web presentation objects to this set of tags. Through the extraction process, only tags with "display" or "type" attributes which are not equal to "hidden" or "none" are processed. This condition entails using only user meaningful tags.

3.1 Web Zone Object Extraction Algorithm

Procedure. From a web document DOM tree, extracting web zone objects requires extraction of navigation menus which divide web document into useful and meaningful zones. A web document is composed of at least one web zone object, e.g., a BodyZone, and up to 3 zones objects, e.g., a HeaderZone, a BodyZone, and a FootZone, because data interest rates are different and depend on where data are located in. Thus, we define an algorithm which receives as inputs a web document DOM tree and returns an array containing web zone objects. It uses string comparison for parsing tag sets. From web document DOM trees analyzed during our preliminary work, we notice as (Liu et al., 2003) that some nodes have a specific role. In (Liu et al., 2003), some of these specialized nodes are used to only identify data records, e.g., generalized nodes, but we also use those which help us with identifying web zones. We call them region nodes. A region node is a root node of:

- a tree which has at least more than half of its children having similar tag strings,
- a tree which has at least a height of 3,
- a tree which has at least three first children.

Contrary to (Liu et al., 2003) requiring that generalized nodes of a data region has the same length, region nodes do not require the same length because the goal is not identifying data records. Parts of web documents are separated by navigation blocks, e.g., menus. We notice that main menus existing on web documents are usually composed of at least five links (<a> HTML tag) with or without special keywords in their contents. We call these sets of tags Series. A Series is a set of five or more <a> or <area> (hypertext or zone tags) sibling nodes. A web document is composed of three zones at most, so the number of series used to separate these zones are two at most. These two specific series are called"Series1" and "Series2". They are identified according to the region node associated with the first and respectively the last series in the DOM Tree.

Series2 can also be identified as the series enclosing one of these keywords "copyright", "private", "policy", "about our company" because these menus usually contain information about the owner company. In our algorithm for web zone extraction, region nodes are firstly identified and labeled, thus Series1 and Series2 are searched to set up web zone objects. Only the subtree of "body" is meaningful for web object extraction because it contains information rendered to users contrary to the subtree associated to "head". An application of the process is defined below. From our work on web zone extraction, we notice that content and presentation objects before the first menu and included in less than half of the DOM tree belong to the web document HeaderZone. However, content and presentation data after the last menu and included in the second half of the DOM tree belongs to FootZone. So, between these two zones, content and presentation data belong to BodyZone. Hence, our hypothesis for Series searching are: 1) If the search process of Series1 goes over half of the DOM tree size, that means the web document does not have HeaderZone, Series1 is empty and BodyZone's first tag is the first region node child of the closest region node in the subtree of "body" root, 2) If the search process of Series2, from half size of the DOM tree to its end, Series2 does not exist that means the web document does not have FootZone and BodyZone's last tag is the last region node child of the closest region node in the subtree of "body" root. Series1 and Series2 are used to set up web zone objects according to rules called "ZR", which stands for Zone Rule (cf. ZR rules in Application section):

Application. Web content and web presentation object extraction algorithm applied on the web document DOM tree presented in figure 2 uses an array of web zone objects called WebZoneObjectArray composed of three cells. We use node labels of figure 2 to cite them, for example "div2" stands for <div> labeled 2 and "ul1-B1' stands for <ul> of the subtree 1 and labeled B1. During the first task, some nodes are labeled as region nodes like "form", "div1', "div4-A', and "div9" sketched on the snapshot in figure 2. Then, search of Series1 and Series2 through the DOM tree begins effectively from the node "body'. Subtrees "ul1-B1" and "ul1-B2" are parsed and they include respectively four and three <a> tags; those are not special series because they do not have at least 5 <a> or <area> tags. Parsing subtree 2-A2, a series of 13 <area> tags is detected. The closest region node of Body node is the "form" and its first region node child is <div2>. So, Series1 is found and equals to <div2>, e.g. "html/body/form/div2". Through the DOM tree, subtrees 3, 4, 5, 6, 7, 8, and 9 are parsed and several series of <a> tags are found, the last one is in subtree labeled 9-E (five <a> tags). The last region node child of "form" is <div9>, and Series2 is set and equals to <div9>.

Hence, with respect to ZR rules, web zone objects can be initialized.

- ZR1: HeaderZone.FirstTag is Series1 first sibling. It is "div1" because it is the first sibling of <div2>,
- ZR2: HeaderZone.LastTag is Series1 previous sibling. It is "div1" because it is also the previous siblings of <div2>,
- ZR3: BodyZone.FirstTag is Series 1. It is"div2",
- ZR4: BodyZone. is Series2 previous sibling. It is"div8" because it is the previous sibling of <div9>,
- ZR5: FootZone.FirstTag is Series 2. It is"div9",
- ZR6: FootZone.LastTag is Series 2 last sibling. It is"div9" because <div9> is the last sibling at this level in the DOM tree.

Then, HeaderZone.NbTag is set up and equals to 1 because HeaderZone is defined through only one subtree. BodyZone.NbTag is set up and equals to 7 because there are 7 nodes of the same level between "div2" and "div8". FootZone.NbTag is set up and equals to 1 because FootZone is defined through only one subtree.

3.2 Web Content and Presentation Object Extraction Algorithm

Procedure. In a web zone, our web object extraction process begins with web presentation objects and finishes with web content objects. We provide two algorithms called PresWebObjectScan and ContWebObjectScan. PresWebObjectScan algorithm extracts presentation objects with respect to our web presentation class hierarchy and extracts objects such as

"LegalInformation", "Menu", ... Whereas, ContWebObjectScan algorithm extracts web content objects with respect to our web content class hierarchy and extracts objects such as "TextElement", "DefinitionList", "PlugInServer", ... Web presentation objects and web content objects are extracted for each web zone object from the DOM tree. We assume that it is unlikely that an object is spread out through several zone objects because web zone objects are defined as sub-tree of the same level in the DOM tree and enclosed tags are supposed to be coherent and well closed. It is worth mentioning that several web presentation objects can rendered a web content object, and vice versa.

These algorithms are similar in many steps but differ in the way of detecting siblings through breadth-first search. More precisely, PresWebObjectScan scans siblings until a block-level node or no more sibling is left whereas ContWebObjectScan scans siblings until a dissimilar node of string tag, a block-level or no more sibling is left. ContWebObjectScan is slightly different from PresWebObjectScan but easier and due to space we present only PresWebObjectScan. The latter receives as input the DOM tree and a web zone object called WDZoneObject. Objects are extracted and flagged with a counter called "indTag" per web zone. Another counter called "nbTag" is used to keep track of the number of tags parsed at the same level as the web zone object's first tag. At the beginning of this algorithm, an array for web presentation object storage is created. Then, it goes through the DOM tree from its root to WDZoneObject first tag and parses without going over WDZoneObject number of tags "nbTag". Then, it scans recursively sub-trees of each block-level node calling ProcessPresentationSibling algorithm presented in figure 3. Through this search, it looks for non block-level nodes and as soon as one is found, its siblings are explored by a breadth-first search until a block-level or no more sibling left. So, a web presentation object is associated to these sibling tags using our web presentation objects defined in section 2.2. The counter of web presentation object is incremented after each web presentation object extraction. In fact, web presentation objects extraction process uses a depth first sear when a block-level tag is found and a breadth first search when a non block-level tag is found, then the right web presentation object is associated with the set of similar or dissimilar siblings.

Application. PresWebObjectScan algorithm is applied on the HeaderZone of our web document example, so the input are with the DOM tree presented in figure 2 and the HeaderZone object. The process begins by creating PresentationObjectArray and from the DOM tree root, "html", reaches <div1> which is HeaderZone.FirstTag. Then, it reaches "1-A" and calls ProcessPresentationSibling algorithm (call 1 of the algorithm) with "html/body/form/div1/a1-A", the DOM tree, PresentationObjectArray and indTag=1 as inputs.

ProcessPresentationSibling stores "html/body/form/div1/a1-A" into tagArray. Its unique sibling "html/body/form/div1/div" is a block-level (<div>) and the web presentation object associated to tagArray[0] which is an <a> tag with an inner <img> tag is a LegalInformation object. It is also the first element of PresentationObjectArray. The counter "indTag" is incremented to 2 and ProcessPresentationSibling (call 2 of the algorithm) is called with "html/body/form/div1/div1-B", the DOM tree, PresentationObjectArray and indTag=2 as inputs. "html/body/form/div1/div1-B/ul1-B1", is reached and since it is a block-level node, ProcessPresentationSibling (call 3 of the algorithm) is called with "html/body/form/div1/div1-B/ul1-B1/li1-B1A", the DOM tree PresentationObjectArray and indTag=2 as inputs. <li1-B1A> is a non block-level, so itself and all siblings which are non block-level are stored in tagArray. Then, the web presentation object extracted is a Menu object because tagArray is composed of four <li> tags and each of them has an inner <a> tag. This Menu is the second element of PresentationObjectArray.

The counter "indTag" is incremented to 3 and as there is no more sibling of <li1-B1A> left ProcessPresentationSibling (algorithm call 3) is over. In ProcessPresentationSibling (algorithm call 2), <ul1-B1> has a sibling which is <ul1-B2> which is a block-level node. By a depth-first search the process reaches <li1-B2A> and ProcessPresentationSibling (call 4 of the algorithm) is called with "html/body/form/div1/div1-B/ul1-B1/li1-B2A", the DOM tree PresentationObjectArray and indTag=3 as inputs. <li1-B2A> is a non block-level, so itself and all siblings which are non block-level are stored in tagArray. Then, the web presentation object extracted is a Menu object because tagArray is composed of four <li> tags and each of them has an inner <a> tag. This Menu is the third element of PresentationObjectArray. The counter "indTag" is incremented to 4 and as there is no more <li1-B2A> sibling left ProcessPresentationSibling (algorithm 4) is over. In ProcessPresentationSibling (algorithm 2), there is no more <ul1-B2> sibling, so ProcessPresentationSibling (algorithm 2) is over. ProcessPresentationSibling (algorithm 1) is also over because there is no more <a1-A> sibling left. Back in PresWebObjectScan algorithm, the number

```
Algorithm ProcessPresentationSibling (TTag, DOMTree, PresentationObjectArray, indTag)
Input: TTag is the HTML tag value which its siblings will be processed
Other variables: tagArray is an array of similar or dissimilar tag siblings of TTag with distinct names.
Comment: PresentationObjectArray is the global array storing web presentation objects. indTag is a global
index for labeling presentation objects per zone.
begin
  if TTag is not a block-level tag then
    repeat
      -Store TTag in tagArray
      -Store TTag siblings found in tagArray
    until there is no more sibling left
    for each TTagSibling in tagArray
    begin
    If TTagSibling is a block-level tag then
      -Associate an object to tagArray[TTagSibling index-1] with respect to our web presentation object
      model
      -Store this object in the PresentationObjectArray cell indexed indTag
      -indTag:= indTag+1
      -TTag is set up with TTagSibling
      -Call recursively ProcessPresentationSibling (TTag, DOMTree, PresentationObjectArray, indTag);
    endif;
    end;
    -Associate an object to tagArray with respect to our presentation web class hierarchy
    -Store this object in the PresentationObjectArray cell indexed indTag
    -indTag:= indTag+1
  else
    -TTag is set up with the next node of the DOMtree by a depth-first search
    -Call recursively ProcessPresentationSibling (TTag, DOMTree, PresentationObjectArray, indTag)
    -TTag is initialized to the next node of the DOMtree by a breadth-first search
  endif; /*All TTag siblings have been parsed*/
end;
```

Figure 3: Algorithm for sibling search in web presentation object extraction process.

of tags parsed is incremented to one that means the end of the loop because HeaderZone.NbTag=1 and PresWebObjectScan end. Hence, the result of PresWebObjectScan application on Chapters web page HeaderZone is an array of three web presentation objects, LegalInformation1, Menu1 and Menu3.

These web presentation objects of Chapters web document HeaderZone are sketched in figure 2 and the web presentation objects hierarchy obtained is sketched in figure 4. The web content objects of this zone obtained by applying ContWebObjectScan are sketched with a dashed arrow in figure 1. At this point, an advantage of our work is to detect fake Chapters web pages, only based on the presentation view. For that, we have to look for web page which

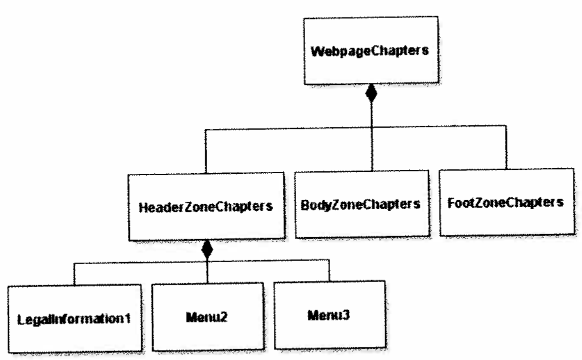

Figure 4: Hierarchy of web objects of figure 1 according to presentation view.

match with the pattern LegalInformation1 (Chapters logo) and Menu2 (menu including user account) in their HeaderZone as it is representative and relevant for every Chapters web page. By the same way, according to the content view, ImageLink object and ListTextlink objects can be mined to find the common features of Chapters fake websites. The interest of the two views rely on crossing patterns found from web content object extraction and those from web presentation object extraction to refine and adjust the profile. The existing approaches allow us to extract objects like images or data record. On top of extracting these objects, with our approach, the space search can be limited to either zones of web or view (content or presentation) of web documents.

4 CONCLUSIONS

The proposed object-oriented meta model for representing web documents as an UML diagram of objects consists of two UML class hierarchies. This model is composed of six web content classes and six web presentation classes because we assume that presentation of web contents impacts user understanding. It represents fine levels of contents (such as title), high level of contents (such as text), unstructured presentation data (such as banner), loosely-structured presentation data (such as menu) and strictly-structured data (such as record). The data extraction algorithm processes any given web document. It pays attention to content and presentation aspects of data on web documents mentioned in related work and it generalizes previous work because it goes further in data block, record on web documents. This new object-oriented web data model accesses web documents either by objects with fine granularity or by record level. Our model is suitable for handling web documents of any application domain and either web documents generated by database system which contains mainly structured data or web document generated by human beings which contain unstructured data. Our model also allows us to narrow the search space in terms of either content or presentation, or both, and also in terms of more precise zones of web documents. Representing a web document as a list of objects is a framework for other web applications because even separators are modeled between web content objects (that is useful for web segmentation work) and any data types of web documents. Manual applications of the proposed technique on a number of web pages generated detailed web object hierarchies and their accompanying databases for mining. We are working on a complete automation of the proposed algorithms to instantly generate objects on any given set of web pages, mining various object level association rules patterns and sequential patterns with further experimentations. These rules aim at identifying similarities and trends between set of objects intra and inter web documents could be discover easily.

REFERENCES

Abiteboul, S. (1997). Querying semi-structured data. In Afrati, F. N. and Kolaitis, P. G., editors, *Database Theory - ICDT '97, Greece, January 8-10, 1997,*, volume 1186 of *LNCS*, pages 1–18. Springer.

Arasu, A., Garcia-Molina, H., and University, S. (2003). Extracting structured data from web pages. In *SIGMOD '03 international conference on Management of data*, pages 337–348, New York, NY, USA. ACM.

Chapters.ca (2007). Chapters canada website (november 2007).

Crescenzi, V., Mecca, G., and Merialdo, P. (2001). Roadrunner: Towards automatic data extraction from large web sites. In *27th International Conference on Very Large DataBases*, pages 109–118.

Gottlob, G. and Koch, C. (2004). Logic-based web information extraction. *SIGMOD Rec.*, 33(2):87–94.

Kosala, R. and Blockeel, H. (2000). Web mining research: a survey. *SIGKDD Explor. Newsl.*, 2(1):1–15.

Levering, R. and Cutler, M. (2006). The portrait of a common html web page. In *DocEng '06 ACM symposium on Document engineering*, pages 198–204, New York, NY, USA. ACM.

Li, J. and Ezeife, C. I. (2006). Cleaning web pages for effective web content mining. In *DEXA*, pages 560–571.

Liu, B. and Chen-Chuan-Chang, K. (2004). Editorial: special issue on web content mining. *SIGKDD Explor. Newsl.*, 6(2):1–4.

Liu, B., Grossman, R., and Zhai, Y. (2003). Mining data records in web pages. In *KDD '03*, pages 601–606, New York, NY, USA. ACM.

Song, R., Liu, H., Wen, J.-R., and Ma, W.-Y. (2004). Learning block importance models for web pages. In *WWW'04*, pages 203–211, New York, NY, USA. ACM.

UN (2007). United nations english index web page (november 2007).

W3C, W. W. W. C. (2007). Document object model standard. http://www.w3.org/DOM/.

Yu, S., Cai, D., Wen, J.-R., and Ma, W.-Y. (2003). Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *WWW'03*, pages 11–18, New York, NY, USA. ACM.

Zhao, H., Meng, W., Wu, Z., Raghavan, V., and ement Yu, C. (2005). Fully automatic wrapper generation for search engines. In *WWW '05*, pages 66–75, New York, NY, USA. ACM.