Information Extraction from Web pages

Róbert Novotný
Institute of Computer Science
University of P. J. Šafárik
Košice, Slovakia
Email: robert.novotny@upjs.sk

Peter Vojtáš
Department of Software Engineering
Charles University
Prague, Czech Republic
peter.vojtas@mff.cuni.cz

Dušan Maruščák
Department of Software Engineering
Charles University
Prague, Czech Republic
dusan.maruscak@yahoo.co.uk

***Abstract*—We present a chain of techniques for extraction of object attribute data from web pages which contain either multiple object data or detailed data about a single object. We discover data regions containing multiple data records, which will be extracted with help of extraction ontology. Furthermore, we present an additional algorithm for detail-page extraction based on the comparison of two HTML subtrees.**

***Keywords*-page annotation, web information extraction, collaborative**

I. Introduction

Vendors and companies traditionally present their products on the web in the form of structured pages which contain summarized information about one or more products. Such pages can be targeted by the semiautomatic information extraction, where the product data are extracted into instances of data objects. These instances are is usually processed by middleware in order to achieve such tasks as competitor tracking, market intelligence or tracking of pricing information.

There are multiple approaches and implementations which try to achieve the goal of information extraction. Most often, they represent a tradeoff between degree of automation of page extraction (usually with lower precision) and the amount of user effort needed to train data extractor for special type of pages (increasing precision).

In our method, we shall focus on the web pages which contain several structured data records. This is usually the case of vendor web pages which contain information on products and services.

The web extraction from structured web pages is covered by many semiautomatic solutions like Lixto [2] or Stalker [10], which are generally based on the machine learning techniques. Another alternatives are systems which try to leverage the document structure and deduce element sequences containing data records, e. g. IEPAD [4] or MDR [9].

In this paper, we present a system for object attribute values extraction from two kinds of web pages: master pages containing structured data about multiple objects and detail pages, which contain data about single product. We propose an approach which is a combination of three techniques. Firstly, we briefly present a pair of MDR-like algorithms for data record extraction which are suitable for master pages. Then we present an algorithm for detail-page extraction based on the comparison of sources of two web pages.

These approaches do not require any user effort, aside from the definition of extraction ontology required for the master-page extraction.

II. Master Page Extraction

A master page contains data about multiple products in a concise manner – usually in the form of tabular or list form. Data are extracted in multiple phases, which will be briefly explored in the following subsections. (Detailed process can be found in [1].)

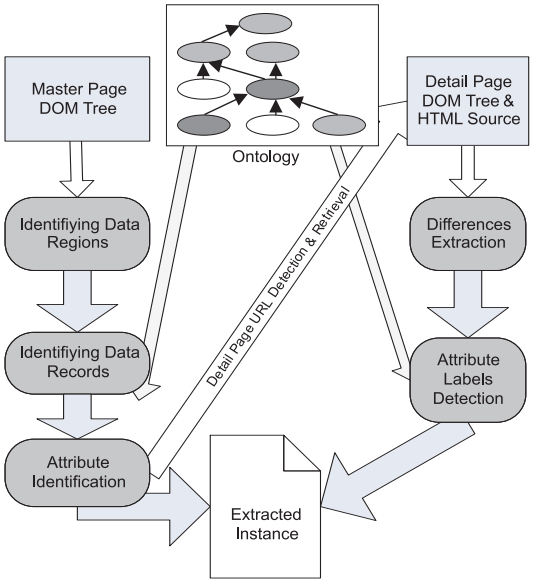

Figure 1. An Overview of Phases

III. Identification of Data Regions

The main observation of this phase is that data records are contained in larger units – data regions (while assuming that a data region contains at least two data records). We parse a web page in the standard Document Object Model (DOM) representation. Then we classify nodes by their weight,

978-0-7695-3801-3/09 $26.00 © 2009 IEEE
DOI 10.1109/WI-IAT.2009.245

where 0-weighted nodes will be pruned from the DOM tree. The higher weight corresponds to higher probability that the node contain text-nodes in their subtrees.

An example of DOM nodes with their weight assignments can be seen in the left part of Figure 2. An integer uniquely identifies each node and the black circle denotes its weight. It can be observed that only about one third of level nodes are considered for further processing (zero-weighted nodes are left out). This significantly reduces space to be searched. The pruned layer of tree can be seen in the top right part of the figure.

The actual search for data regions is based on the breadth-first search. We try to discover repeated similar sequences of subtrees under the nodes with weight 1 or 2. At first, we start with comparison of source text similarity below different nodes, then pairs of consecutive nodes, triples of nodes, etc. The similarity is computed according to the Levenshtein edit distance. Similar subtrees mean that a data region candidate has been discovered and we can search it for data records in the next phase.

IV. Identification of Data Records

In this phase we take a data region and try to discover the data records, where each data record contains information about a single object. There are two possibilities: the "easy" one, when the records are contiguous in the HTML source and each of them forms a proper subtree. The other case is represented by visually contiguous, but HTML-scattered values. In this case, we either estimate the number of data records (by removing non-text tags and pairing up the individual HTML tag groups) or postpone the data record identification into next phase.

V. Attribute Values Identification from Master Page

Having successfully identified data regions and data records it is necessary to retrieve the actual attribute values of discovered records. We take each discovered data record and tokenize the text nodes of the corresponding DOM trees by treating HTML tags, special characters and conjunctions as separators. The tokenization leads to the approximate determination of attribute values. However we have to match the tokens to the attribute values which is obtained with the help of the ontology. The leverage of ontology allows us to detect the structure of data records, the cardinality of extracted attribute values and possibly the relationships between them. An extraction ontology, which can specify attribute labels, regular expression, keywords or possible values enumeration is used to refine unnecessary values and improve the extraction precision.

The ontology has a substantial effect on the application domain used in the extraction process. Adding more annotated properties to ontology (e. g. keywords like ATA, SATA) or attribute labels (hard drive, HDD) increases the success rate. However there may arise domains and attributes, which cannot be easily extracted. For example an attribute representing student's first name can not be extracted in a simple way, since it has the infinite domain, which cannot be exactly described by the annotated properties. Another problematic attributes are those, which can be both atomic and nonatomic. In this case we can identify all found tokens and estimate the number of records by number of found attributes and the order of their discovery. If the attributes are in the correct order, we can extract the actual records. However, this approach cannot deal with missing attributes, since we would have n attributes and $n-1$ values. This can be solved by extracting the attribute values from the detail pages, which contain exactly one data record.

The experiments have shown that attribute labels are occuring in approx. one third of tested web documents. However, the occurence of attribute labels can be increased even more. Almost all of the documents contained the link to the product detail pages, which contains more and detailed information, often with the attribute labels. By extracting data from such pages we can increase the efficiency of the extraction process (see the next section).

VI. Attribute Value Extraction from Detail Pages

A detail page is a web document which displays all known information about single product. While the primary (summary) page usually presents a concise information about multiple products, the detail page typically presents additional attributes or the existing attributes in more structured form. Therefore using this kind of pages can help us to improve the success rate of information extraction about products.

A simple observation of detail pages would show the fact, that they are visually very similar. This similarity arises from the fact, that the underlying implementation of web portals is often based on the template engines (see [12]). A typical template of the template engine represents a skeleton of HTML page with variables. These variables are in the process of template evaluation merged with data model, which contains (besides other things) the attributes of the product.

The dissimilarity between two web pages (more precisely, between two DOM trees) is usually occuring on the three types of places:

- the differences between structures of DOM subtrees – this represents a situation, when one page contains a subtree, which does not occur in the another one (typically when one product contains attribute, which does not occur in another one),
- the differences between HTML attributes – this represents differences between HTML attributes or their values,

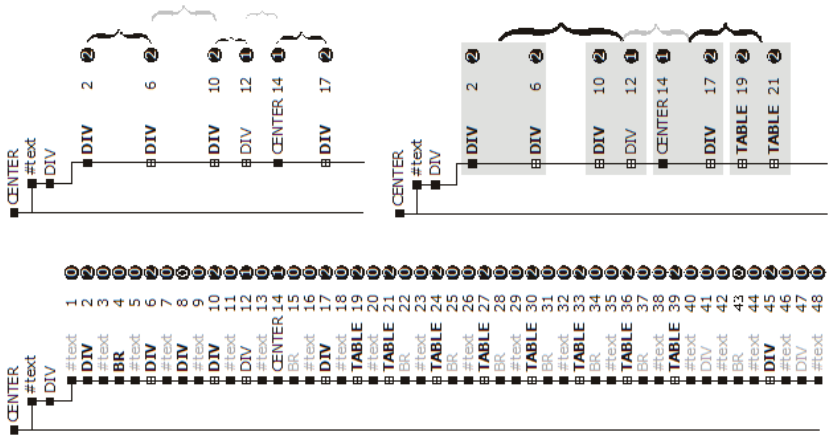

Figure 2. A Sample DOM Model (left), a pruned DOM model with node pairs comparison (top right) and a pruned DOM model with node triples comparisons.

- the differences between text nodes, which represents different values in HTML tags.

We have focused on the third type of difference spots. The pseudocode of attribute extraction is as follows:

- Pick two detail pages, retrieve their HTML source and perform its normalisation (pretty printing, indenting).
- Apply the *diff* algorithm ([5]) and retrieve difference spots.
- Find the nearest HTML element, which contains the difference spot in its subtree. If the difference spot is in the child text node, declare the whole next node as an extraction candidate. If the spot is in HTML attribute, ignore it (since HTML attributes do not usually contain suitable object attributes). If the difference spot spans multiple HTML nodes, consider this difference to be a DOM subtree difference and ignore it (or, pass it to another method of extraction).

This process results in a set of possible candidates for attribute values. However, we have to match these values to the actual attributes. This can be achieved in two ways: either by attribute labels detection or by using the approach which uses ontology in a similar way that is mentioned in section V.

VII. Attribute Labels Detection

Attribute labels detection is based on the observation that the attribute labels are explicitly denoted with some kind of separator (usually a colon). For each element containing a difference spot we can find a nearest left text node and detect whether it contains the separator. If so, we can match the given value with the appropriate attribute.

It the element does not contain a separator, we can still consult the ontology, particularly the name annotation property, if it is equal with enumerated labels.

Moreover, we plan to implement the approach in which we consult the retrieved labels with either WordNet or a suitable word hierarchy (e. g. Google Topic Maps).

Besides implemented methods, we have examined other approaches to attribute labels detection. Two are based on the visual emphasis of HTML elements. We can expect that the attribute labels which are in the vicinity of attribute values are often presented in some kind of accentuated form (typically in bold text), what is typically achieved via CSS rules. We can possibly calculate the relative visual weight of each element containing text nodes, which is in the neighbourhood of the difference spot and take one with the largest weight and consider it to be a label.

VIII. Experiments and Conclusion

We have performed experiments on the application domain representing details about notebooks. We have taken 30 web documents from 23 different major notebook retailers in the Czech Republic and Slovakia. These web documents (all of them in Czech or Slovak) were chosen to be taken from different portals having various structure and visual representation. However, we have focused on the "master" web pages containing multiple data regions with data records with optional links to the detail pages.

In this domain we are interested in following attributes: manufacturer, processor type, RAM size, hard drive capacity, display resolution, optical drive type, presence or absence of WiFi and price. The second, third, fourth and sixth attribute can be described by regular expressions and the attribute labels. Keyword specification can be used in processor type and WiFi. Manufacturer has been described by explicit enumeration, but this causes no difficulties, since the list of manufacturers is relatively small (we have used 10 manufacturers).

On the test data, we have reached a very high soundness / correctness, which was above 95%. In case of structured data

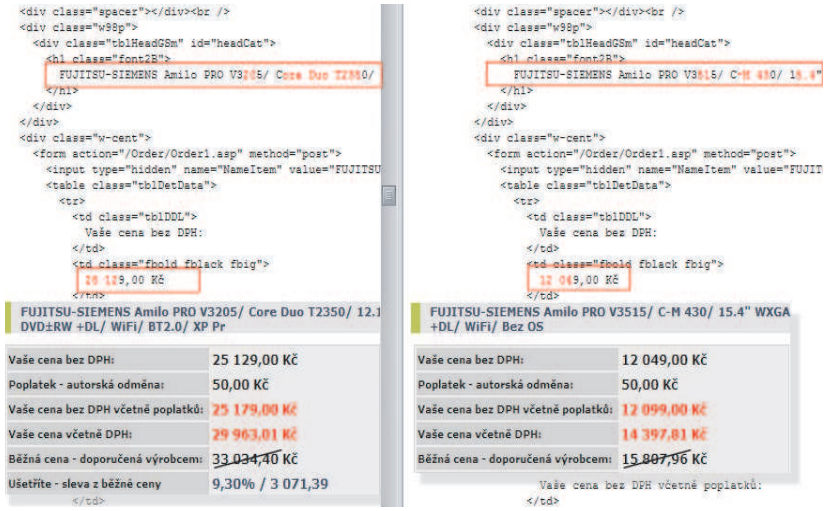

Figure 3. An example of difference spots

we have experienced the only encountered misdiscovered data regions were those which contained nondomain data. However, as we have said, these data were eliminated in the Attribute Identification phase.

Table I
OPTIMAL VALUES FOR EFFICIENT EXTRACTION

Domain	K	s	$l_{\min}$	$l_{\max}$
Notebooks	1	0.1	800	100
Cars	1	0.2	300	80
Hotels	2	0.3	800	200

Moreover, we have experimentally found the optimal values for the most efficient extraction in the first two phases (see Table I) in additional two domains. These values represent Levenshtein similarity (s), minimal and maximal length of attribute value text ($l_{\min}$ and $l_{\max}$) and the maximum number of nodes used in the repeating sequences search (K).

REFERENCES

[1] D. Maruscak, R. Novotny, P. Vojtas, "Unsupervised Structured Web Data and Attribute Value Extraction," Proceedings of Znalosti 2009.

[2] R. Baumgartner, S. Flesca, G. Gottlob, "Visual Web Information Extraction with Lixto," in Proceedings of the 27th VLDB Conference, 2001. pp. 119-128.

[3] V. Crescenzi, G. Mecca, P. Merialdo, "RoadRunner: Towards Automatic Data Extraction from Large Web Sites," in Proceedings of the 27th VLDB Conference, 2001.

[4] C-H. Chang, S-L. Lui, "IEPAD: Information extraction based on pattern discovery". WWW-10, 2001.

[5] DiffUtil [online] Cited on May 25th, 2007. http://www.gnu.org/software/diffutils/

[6] Document Object Model (DOM) Level 2 Core Specification. http://www.w3.org/TR/DOM-Level-2-Core/

[7] D. W. Embley, D. M. Campbell, R. D. Smith, S. W. Liddle, "Ontology-Based Extraction and Structuring of Information from Data-Rich Unstructured Documents," in CIKM '98: Proceedings of the Seventh International Conference on Information and Knowledge Management, 1998, 1-58113-061-9, pp. 52–59.

[8] N. Kushmerick, "Wrapper induction: efficiency and expressiveness," in Artificial Intelligence, 118:15-68, 2000

[9] B. Liu, R. Grossman, Y. Zhai, "Mining Data Records in Web Pages," in Proc S IGKDD.03, August 24-27, 2003, Washington, DC, USA.

[10] I. Muslea, S. Minton, C. Knoblock, "A hierarchical approach to wrapper induction," in AGENTS '99: Proceedings of the third annual conference on Autonomous Agents, 1999. pp. 190–197.

[11] OWL Web Ontology Language Reference. Cited on Jul 28rd, 2007. http://www.w3.org/TR/owl-ref/

[12] Smarty: Template Engine. [online] Cited on Jul 28th, 2007. http://smarty.php.net