Automatic Wrapper Generation Using Tree Matching and Partial Tree Alignment

Yanhong Zhai and Bing Liu
Department of Computer Science
University of Illinois at Chicago
851 S. Morgan Street, Chicago, IL, 60607
yzhai, liub@cs.uic.edu

Abstract

This paper is concerned with the problem of structured data extraction from Web pages. The objective of the research is to automatically segment data records in a page, extract data items/fields from these records and store the extracted data in a database. In this paper, we first introduce the extraction problem, and then discuss the main existing approaches and their limitations. After that, we introduce a novel technique (called DEPTA) to automatically perform Web data extraction. The method consists of three steps: (1) identifying data records with similar patterns in a page, (2) aligning and extracting data items from the identified data records and (3) generating tree-based regular expressions to facilitate later extraction from other similar pages. The key innovation is the proposal of a new multiple tree alignment algorithm called *partial tree alignment*, which was found to be particularly suitable for Web data extraction. This paper is based on our work published in KDD-03 and WWW-05.

Introduction

Structured data in Web pages usually contain important information. Such data are often retrieved from underlying databases and displayed in Web pages using fixed templates. In this paper, we call these structured data objects *data records*. Extracting data records enables one to integrate data/information from multiple Web sites and pages to provide value-added services, e.g., comparative shopping, meta-querying and search. Data extraction is a reverse engineering task. That is, given the HTML encoded data (i.e., Web pages), the extraction system recovers the original data model (schema) and extracts data from the encoded data records.

There are two main types of data rich pages on the Web.

1. List pages: Each of such pages contains lists of objects. Fig. 1 shows such a page. From a layout point of view, we can see two data regions (one horizontal and one vertical). Within each region, the data records are formatted using the same template. The templates used in the two regions are different.
2. Detail pages: Such a page focuses on a single object, as shown in Fig. 2.

In Fig. 1, the description of each product is called a data record. Notice that the data records in this page are all flat with no nesting. Fig. 3(a) contains some nested data records, which makes the problem harder and also more interesting. The first product "Cabinet Organizers by Copco" has two sizes, 9-in. and 12-in. with different prices. These two organizers are not at the same level as "Cabinet Organizers by Copco".

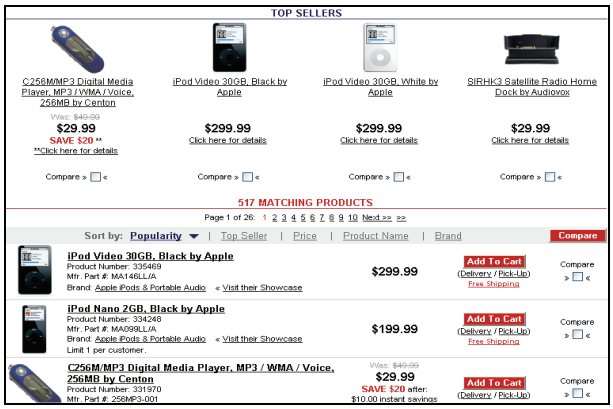

Fig. 1. A segment of a list page with 2 data regions

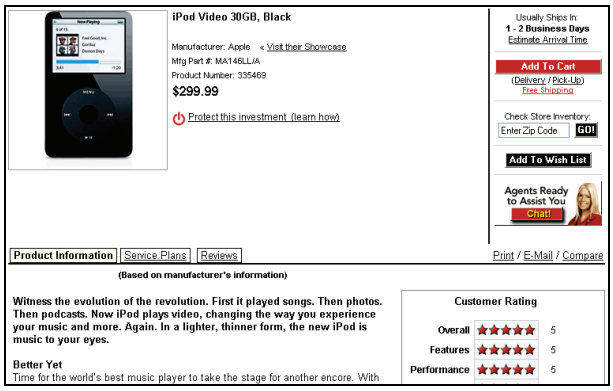

Fig. 2. A segment of a detail page

Our work focuses on data extraction from list pages. The objective is threefold: (1) automatically identify data records in a page, (2) align data items in a data table such as the one given in Fig 3(b), and (3) construct the templates (regular patterns) from identified data records. The problem can be generalized as detecting similar patterns in a Web document

(a). An example page segment with nested data

image 1	Cabinet Organizers by Copco	9-in.	Round Turntable: White	*****	$4.95
image 1	Cabinet Organizers by Copco	12-in.	Round Turntable: White	*****	$7.95
image 2	Cabinet Organizers	14.75x9	Cabinet Organizer (Non-skid): White	*****	$7.95
image 3	Cabinet Organizers	22x6	Cookware Lid Rack	****	$19.95

(b). Extraction results. "image 1" and "Cabinet Organizers by Copco" are repeated for the first two rows due to the nest-

and generate a tree-based regular expression from these patterns.

In this paper, we first describe several existing approaches and their main limitations. We then discuss our work in this area published in WWW-05 (Zhai and Liu 2005) and KDD-03 (Liu, Grossman and Zhai 2003).

Existing Solutions and Limitations

Existing techniques for the structured data extraction problem can be classified into three main categories: 1) Wrapper programming languages and visual wrapper generation platforms, 2) wrapper induction, and 3) automatic data extraction.

The first approach provides some specialized pattern specification languages to help the user construct data extraction programs. Visual wrapper generation platforms use such languages and hide their complexities under easy-to-use graphical wizards and interactive processes. Systems that follow this approach include WICCAP (Zhao and Wee 1994), Wargo (Raposo, Pan and Alvarez et al. 2002), Lixto (Baumgartner, Flesca and Gottlob 2001), DEBye (Laender, Neto and Silva 2002), etc. Clearly, this approach cannot scale to a large number of sites or pages.

Wrapper Induction

The second approach is wrapper induction, which uses supervised learning to learn data extraction rules from a set of manually labeled positive and negative examples. The learned rules are then applied to extract target data from other pages using the same template. For example, the Stalker system (Muslea, Minton and Knoblock 1999) learns two rules to identify the beginning and the end of a target item, which are called the *start rule* and the *end rule*. Given a set of training examples *E*, the learning algorithm tries to generate extraction rules that can extract all the target items (also called positive items) without extracting any other items (called negative items). Learning is done based on the machine learning method, such as sequential covering (Muslea, Minton and Knoblock 1999).

Example wrapper induction systems include WIEN (Kushmerick 2000), Softmealy (Hsu and Dung 1998), Stalker (Muslea, Minton and Knoblock 1999), WL^2 (Cohen, Hurst and Jensen 2002), Thresher (Hogue and Karger 2005.), etc.

Manual labeling of data for wrapper induction is, however, labor intensive and time consuming. To avoid unnecessary labeling, *active learning* is proposed as an approach to help identify informative unlabeled examples.

The main shortcomings of this approach are as follows:

1. It is unsuitable for a large number of sites due to the manual labeling effort as for different sites or even different pages in the same site the manual labeling process needs to be repeated.
2. Wrapper maintenance is very costly. The Web is a dynamic environment where sites change constantly. If a site changes, existing extraction rules for the site become invalid, which introduces two problems.

- If the site changes, does the wrapper know the change? This is called the *wrapper verification problem*.
- If the change is correctly detected, how to automatically repair the wrapper? This is called the *wrapper repair problem*.

Both these problems are very challenging. There is still no satisfactory solution so far.

Due to these problems, automatic extraction has been studied by researchers in recent years. Automatic extraction is possible because data records in a Web site are usually encoded using a limited number of fixed templates. It is possible to find these templates by mining repeated patterns in multiple data records.

Automatic Wrapper Generation

As we mentioned in the introduction, there are two types of data rich pages, detail pages and list pages. We discuss extraction from them separately below.

Wrapper generation based on detail pages: In (Crescenzi, Mecca and Merialdo 2001), a matching algorithm is proposed to infer union-free regular expressions from multiple pages with the same template. The resulting regular expression is then used to extract data from other similar pages. The full algorithm has an exponential time complexity. To limit the search, heuristic pruning techniques are employed, which compromise the expressive power of the inferred grammar. This approach was improved in (Arasu and Molina 2003), which presents a polynomial time algorithm by using several heuristics. Both methods need multiple input pages with a common schema/template and assume that these pages are given.

It is clear that for extraction from detail pages, multiple pages are required to find patterns because each page only focuses on a single object. However, this approach has the following difficulties:

- Although a detail page focuses on a single object, the page may contain a large amount of "noise", i.e., irrelevant information on the boundary of the page. Without target items to extract, finding patterns to extract every piece of information is not only time consuming, but may also be ineffective due to irregular formatting of the "noisy" information, which can make pattern generation difficult.
- Finding a set of detail pages generated by a common template as input pages to the techniques is not a trivial task itself.

We note that these techniques can also be used to extract data from list pages, but they still require multiple input list pages containing similar lists of objects. It has been shown that for list pages (Liu, Grossman and Zhai 2003), multiple input pages are not necessary.

Extraction from list pages: For list pages, a single page is sufficient because it is possible to find patterns from multiple data records in a list to do the extraction.

In (Embley, Jiang and Ng 1999), a study was made to automatically identify data record boundaries based on a set of heuristic rules and domain ontologies. A domain ontology is costly to build (about 2-person weeks for a given Web site) (Embley, Jiang and Ng 1999). In (Buttler, Liu and Pu 2001), additional heuristics are proposed to perform the task without using domain ontologies.

In (Chang and Lui 2001), a method (called IEPAD) is proposed to find patterns from the HTML tag string of a page, and then use the patterns to extract data items. The method uses the Patricia tree (Gonnet and Yates 1991.) and sequence alignment to find patterns that allow inexact matches. However, the algorithm generates many spurious patterns and users have to manually select the correct pattern for extraction. Following the work in (Chang and Lui 2001), a system for wrapper generation and label assignment is proposed in (Wang and Lochovsky 2003). In this work, a candidate wrapper is generated based on a single page by finding repeated patterns in the HTML string, and then multiple similar pages are used to determine a generalized wrapper based on multiple candidate wrappers. This work thus needs multiple pages with a common schema and template as the input.

Another method for data extraction is proposed in (Lerman, Getoor and Minton et al. 2004). Its main idea is to utilize the redundant information in list pages and detail pages to aid information extraction. The method also needs multiple similar pages to infer a template to find tables in list pages, and then finds records from the tables. It is not applicable to pages where data records in list pages have no links to detail pages.

The main existing approaches are based on string matching and multiple string alignments to find patterns. However, the approaches were shown not very accurate. We proposed a tree-based matching algorithm and a new multiple tree alignment technique called *partial tree alignment* to perform the task, which has been shown to be much more effective for Web data extraction (Zhai and Liu 2005).

Our Work

Our approach uses both visual (rendering) information and tree alignment. To perform the data extraction task, our proposed technique works in three steps:

1. Given a page, the method first identifies the data records with similar patterns via a top-down tree distance measure, calculated by the *Enhanced Simple Tree Matching* algorithm (Zhai and Liu 2005).
2. A novel *partial tree alignment* method is proposed to align and to extract corresponding data items from the discovered data records and put the data items in a database table.
3. Generate tree-based regular patterns for later extraction.

Data Record Identification

The identification of data records in a page is based on two important observations:

1. A group of data records that contains descriptions of a set of similar objects are typically rendered in a contiguous region of a page and are formatted using similar HTML tags. Such a region is called a *data record region* (or *data region* in short). We can use a tree matching approach to compare different sub-trees to find those similar ones, which may represent similar data records. The problem with this approach is that the computation is prohibitive because a data record can start from any tag and end at any tag. The next observation helps to deal with this problem.
2. A set of similar data records are formed by some child sub-trees of the same parent node. This observation makes it possible to design a very efficient algorithm based on tree mapping to identify data records because it limits the tags at which a data record may start and end.

Given a Web page, the identification works in 3 steps:

Step 1: Building a DOM tree of the page using visual (rendering) information. Utilizing visual information instead of following the traditional way of analyzing the nested HTML tag structures leads to more robust DOM tree construction due to the coordination between the parsing and rendering engines of a browser. A page can be rendered properly even its HTML tags are ill-formatted. In this way, as long as a page can be rendered correctly, its DOM tree can be built correctly.

Step 2: Mining data regions in the page using the DOM tree. A data region is an area in the page that contains a list of similar data records. Instead of mining data records directly, which is hard, the algorithm mines data regions first and then finds data records within them. The top-down tree distance is adopted as the distance measure and we introduce an algorithm named *Enhanced Simple Tree Matching* for computing top-down distance between two trees. In general, HTML tag labels, visual information and textual contents enclosed in HTML elements are all used in deciding whether two nodes can match or not.

Step 3: Identifying data records from each data region.

Data Extraction Using Partial Tree Alignment

To extract data items from each data record, the key task is how to match corresponding data items from all data records. We proposed the partial tree alignment technique for this purpose, which consists of two steps:

1. Produce one rooted DOM tree for each data record.
2. Align the DOM trees of all data records in each data region using *partial tree alignment*.

The partial tree alignment approach aligns multiple DOM trees by progressively growing a seed tree. The seed tree, denoted by T_s, is initially picked to be the tree with the maximum number of data fields. Then for each tree T_i (i s), the algorithm tries to find for each node in T_i a matching node in T_s. When a match is found for node $T_i[j]$, a link is created from $T_i[j]$ to $T_s[k]$ to indicate its match in the seed tree. If no match can be found for node $T_i[j]$, the algorithm attempts to expand the seed tree by inserting $T_i[j]$ into T_s. However, the algorithm only inserts $T_i[j]$ into T_s if a location for the insertion can be *uniquely* determined in T_s. The expanded seed tree T_s is then used in subsequent matching (see (Zhai and Liu 2005) for details).

Extraction Pattern Generalization

After the alignments of items are performed, we can generate a grammar for data extraction, which is a task of grammar induction. In general, grammar induction needs a finite set of positive and negative examples to generate the grammar. However, in our case there are only positive examples. Fortunately, structured data in Web pages are usually highly regular which enables us to design an algorithm to produce "simple" regular expressions based on positive examples only.

Empirical Evaluation

We evaluated DEPTA (Data Extraction based on Partial Tree Alignment) using 200 Web pages from 142 sites.

We show in Table 1 the performance of DEPTA on data record extraction and data item alignment. For data record extraction, the recall and precision are computed based on the total number of correctly identified data records and the actual number of data records in all pages. For data item extraction, the precision and recall computation has considered all the incorrectly extracted or missing data records introduced in step 1 of DEPTA. For a data record, an incorrect extraction means that only part of the content of a data record is extracted, or information outside of the data record boundary is extracted and enclosed in it. For a data item, incorrect alignment means items of the same attribute are placed into different columns, or items of different attributes are placed into the same column.

Table1: Experimental result

	Data Record Extraction	Data Item Extraction
Recall	96.18%	98.08%
Precision	95.25%	93.26%

Conclusions

In this work, we discussed the problem of structured data extraction from Web pages, followed by a description of existing approaches and their deficiencies. We then introduced our new algorithm for the task which consists of three steps: (1) identifying data records by detecting similar patterns in a DOM tree based on a top-down tree distance measure, (2) aligning corresponding data items from multiple data records using a partial tree alignment algorithm, and (3) generating tree-based regular expressions for later extraction.

References

Arasu A.; and Molina H. 2003. *Extracting structured data from web pages*. SIGMOD'03.

Baumgartner R.; Flesca S.; and Gottlob G. 2001. *Visual web information extraction with lixto*. VLDB'01.

Buttler D.; Liu L; and Pu C. 2001. *A fully automated object extraction system for the world wide web*. In ICDCS'01.

Chang C.; and Lui S. 2001. *IEPAD: Information extraction based on pattern discovery*. In WWW'01.

Cohen W.; Hurst M.; and Jensen L. 2002. *A flexible learning system for wrapping tables and lists in html documents*. In WWW'02, pages 232–241, New York, NY, USA.

Crescenzi V.; Mecca G.; and Merialdo P. 2001. *Roadrunner: Towards automatic data extraction from large web sites*. In VLDB'01.

Embley D.; Jiang Y.; and Ng Y. 1999. *Record-boundary discovery in web documents*. In SIGMOD'99.

Gonnet G.; and Yates R. 1991. *Handbook of Algorithms and Data Structures in Pascal and C*.

Hogue A.; and Karger D. 2005. *Thresher: Automating the unwrapping of semantic content from the world wide web*. In WWW'05.

Hsu C.; and Dung M. 1998. *Generating finite-state transducers for semi-structured data extraction from the web*. Inf. Syst., 23(9):521–538.

Kushmerick N. 2000. *Wrapper induction: efficiency and expressiveness. Artificial Intelligence*, 118:15–68.

Laender A.; Neto B.; and Silva A. , 2002. *Debye - date extraction by example*. Data Knowl. Eng., 40(2).

Lerman K.; Getoor L.; Minton S.; and Knoblock C. 2004. *Using the structure of web sites for automatic segmentation of tables*. In SIGMOD'04.

Liu B.; Grossman R.; and Zhai Y. 2003. *Mining data records in web pages*. In KDD'03.

Muslea I.; Minton S.; and Knoblock C. 1999. *A hierarchical approach to wrapper induction*. In AGENTS'99.

Raposo J.; Pan A.; Alvarez M.; Hidalgo J.; and Vina A. 2002. *The wargo system: Semi-automatic wrapper generation in presence of complex data access modes*. In 13th International Workshop on Database and Expert Systems Applications.

Wang J.; and Lochovsky F. 2003. *Data extraction and label assignment for web databases*. In WWW'03.

Zhai Y.; and Liu B. 2005. *Web data extraction based on partial tree alignment*. In WWW '05.

Zhao, H.; Meng, W.; Wu, Z.; Raghavan, V.; and Yu, C. 2005. *Fully automatic wrapper generation for search engines*. In WWW '05.

Zhao L.; and Wee N. 1994. *WICCAP: From Semi-structured Data to Structured Data*. In ECBS '94.