Web Data Extraction from Template Pages

Sayali P. Badhan
Department of Computer Engineering,
K.K.Wagh Institute of Engineering Education and Research,
University of Pune, India
sayalibadhan@gmail.com

Sumedha K. Chumble
Department of Computer Engineering,
K.K.Wagh Institute of Engineering Education and Research,
University of Pune, India
sumedha.chumble@gmail.com

Abstract**—**The World Wide Web is a rapidly growing source of information. Deep web contains more and valuable information as compared to surface web. Many websites contain large number of pages generated using a common template. The data values used to generate these pages come from the database. Web data extraction enables us to pose complex queries over the data. Web data extraction has also been recognized as one of the important problems in information integration systems which integrate the data present on different websites in different formats. Various approaches exist for extracting the data from web pages. The system proposed in this paper is a page level web data extraction system which extracts schema and template from these template based web pages automatically. It uses a machine learning based method which compares HTML tag pairs to estimate how likely they present in web pages. It uses one of the ML techniques called J48 decision tree classifier. It also uses image comparison to assist template detection to improve the performance.

Keywords*- DOM Elements; Structured data; Tree Merging; Web Data Extraction; Wrapper Induction.*

*****

I. INTRODUCTION

Pages belonging to the same Website share the same template since they are encoded in a consistent manner across all the pages. So, these pages are generated with a predefined template by plugging data values. Template pages can also occur in surface Web. For example, commercial Websites often have a template for displaying company logos, browsing menus, and copyright announcements, such that all pages of the same Website look consistent. In addition, templates can also be used to render a list of records to show objects of the same kind. Thus, information extraction from template pages can be applied in many situations.

For template pages the extraction targets for template web pages are almost equal to the data values embedded during page generation. Thus, the key to automatic extraction depends on whether we can deduce the template automatically.

Generally, templates as a common model for all pages, occur quite fixed as opposed to data values which vary across pages. Finding such a common template requires multiple pages or a single page containing multiple records as input. When multiple pages are given, the extraction target aims at page-wide information. When single pages are given, the extraction target is usually constrained to record-wide information, which involves the addition issue of record-boundary detection. Page-level extraction tasks, although do not involve the addition problem of boundary detection, are much more complicated than record-level extraction tasks since more data are concerned in page level extraction.

Deep web contains more valuable information than surface web. Also, current websites present information on same topic in different formats. But, much efforts are required for an user to manually locate and extract useful data from websites. So, there is a need of a value-added service that integrates the information from different sources. So, to develop these information integration systems we need good tools for information extraction from web pages. A conventional approach for extracting the data from various web pages is to write the programs called as wrappers or extractors based on the prior knowledge of the format of web pages. However, programming wrappers require manual coding and is therefore labor-intensive. Also, format of websites is often subject to change. Therefore, maintaining wrapper can become expensive and impractical.

Recently, various approaches have been proposed to fully automate the task of wrapper generation in web data extraction process. The approach proposed in this paper deals with the automatic detection of template and schema of the website. An idea is to implement efficient algorithm for web data extraction for deep websites where all the pages of the same site share the common template. Before DOM tree generation the template blocks of the webpage are filtered. For peer node recognition a machine learning technique j48 decision tree algorithm is used to identify similar subtrees.

II. LITERATURE SURVEY

In this section, current work related to our work including the approaches that use DOM tree information to construct the wrapper, the approaches that use visual information to help extracting data records, and the approaches that are page-level extraction systems are described.

A. Approaches using DOM Tree Information

Many approaches for Web data extraction consider and operate on DOM tree structure. MDR [18] analyzes the child nodes under each parent node and finds generalized nodes and data regions by enumerating possible combinations of child nodes. In MDR string edit distance is used to compute the similarity between tag sequences of two generalized nodes. However, the goal of MDR is to identify data records. MDR does not align the data items in each data record. Meanwhile, due to missing and noisy information, it may find wrong combination of subtrees, DEPTA [8] uses visual gaps between data records to find out data records. It uses partial tree alignment technique to align data fields of data records. NET [10] extends DEPTA by supporting extraction of nested records. ViPER [4] uses primitive tandem repeats and visual context information for record segmentation and enhances the concept of generalized nodes. This provides a better subtree comparing method than MDR which allows consecutive data records with various lengths. In DeLa [9] sufix trees are built to detect C-Repeated patterns in web page string and its algorithm can extract the nested objects. FiVaTech [1] uses tree matching score for subtree comparison, however, the bigger goal is to find the schema and template for the whole page.

B. *Approaches using Visual Information*

Some approaches improve the task of web data extraction by using the visual information. ViNTs [2] and MSE [3] use visual content features on a browser to identify candidate content line. ViPER [4] uses visual information for global multiple sequence alignment. Although visual information is used in these approaches, for similarity calculation they still use HTML tag structure as primary information. ViDE [5] constructs a visual block tree. Its main visual features are position features, layout features, appearance features, and content features and they can be obtained from web page layout (location, size, and font).

C. *Page-level Extraction Systems*

EXLAG [6] and RoadRunner [7] are unsupervised systems for page level web data extraction. RoadRunner extracts data by comparing a pair of web pages to get the template. It works in three steps: A (Align), CM (Collapse under Mismatch), and E (Extract). It supports the backtracking mechanism if optional or iterated tags are found. EXLAG extracts data by forming and analyzing equivalence classes. In EXLAG dTokens (Differentiating Tokens) are aggregated in equivalence classes if they have same occurrence frequency in all input web pages. For template generation large and frequent equivalence classes (LFEQs) are extracted. EXALG and RoadRunner operates on HTML tags, while FiVaTech manipulate DOM trees in order to find out peer nodes (i.e. nodes with the same tag names but different roles).

III. System Architecture

In this section we present the details of the proposed approach. As shown in Fig. 1 given some web pages as an input, we filter template blocks by image comparison. In peer node recognition J48 classifier is used during tree merging. In tree merging module all input DOM trees are merged at the same time into a structure called as Fixed/Variant pattern tree. In the module for schema detection Fixed/Variant pattern tree is used to detect the template and schema of the website.

A. *Filtering Out Template Blocks in the Input DOM Trees*

Generally, deep web pages contain two types of blocks : template data blocks and data rich blocks. Template data blocks contain template data such as navigational panels, advertisements and so on in a web page. Data rich block contains the data of user’s interest in a web page. In order to improve the efficiency of the web data extraction algorithm it is necessary to filter out template data blocks from the web page. In this paper an algorithm to filter out template blocks is proposed before applying the module for tree merging.

Algorithm for filtering out template data blocks has two main assumptions. First, template data blocks are always displayed at the same location with the same content in the web pages of the same website. So, tags corresponding to a particular template block are located at the same location in the

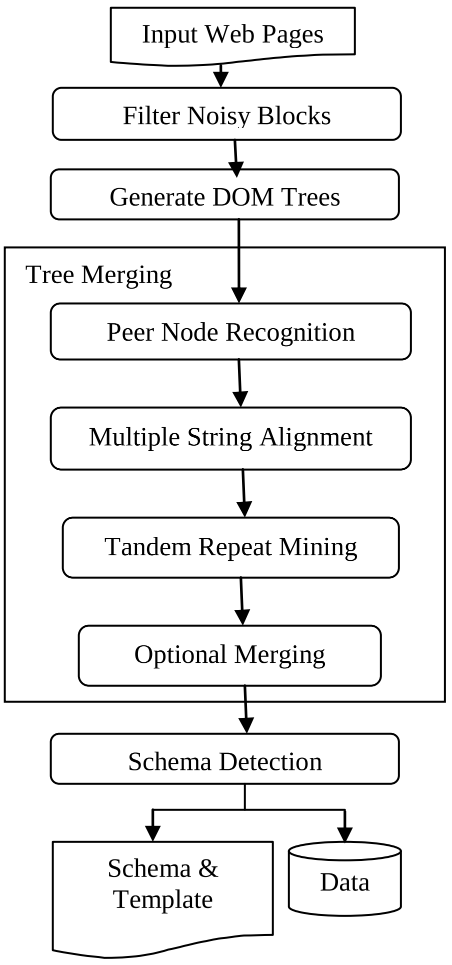

Figure 1. System Architecture

DOM trees and also they have same subtree. Second, The area for a data rich block occupies the biggest area in a whole web page. But it is observed that subtrees with the same images in a data rich area are often not template.

```
Procedure FilterTemplateBlocks(TreeNode)
biggest = the child having biggest area;
if (the area of biggest < 40%) then
    return NULL;
for each child in a TreeNode
     if (child != biggest) then
        compare and remove all equal subtrees (equal images)
        corresponding to the child tag from all DOM trees
     end if
FilterTemplateBlocks (biggest)
```

Figure 2. Algorithm for Filtering Template Blocks

B. *Tree Merging*

There are two main modules in the proposed approach: tree merging and schema detection. The tree merging module merges all the input DOM trees at the same time to form a structure called as Fixed/Variant pattern tree. This Fixed/Variant pattern tree is then used by the module of schema detection to find out the template and schema of the website. The tree merging module consists of four steps: peer node recognition, multiple string alignment, tandem repeat mining and optional merging.

```
Algorithm MultipleTreeMerge(T,P)
// T is a set of DOM trees of the same type
// P is the tag for the roots of T
   Initialize M; i = 0;
   for each tree t in T
          j = 0;
         for each child c in t
               M[j++][i] = c;
         end for
         i++;
   end for
   recognizePeerNode(M);
  childList = matrixAlignment(M);
  childList = repeatMining(childList,1);
  mergeOptional(childList);
  for each node c in childList
       if(c is a tree) then
         C=MultipleTreeMerge(recognizePeerNode(c,M), tag(c));
       else  // c is a leaf node
         C = c;
       end if
       Insert C as a child of P;
  end for
  return pattern tree P;
```

Figure 3. Algorithm for Tree Merging

In peer node recognition step, two nodes with the same tag name are compared to check whether they are peer subtrees or not. For this a decision tree algorithm is used to judge whether two input subtrees (nodes) at the same level are peer nodes (subtrees) or not. J48 Decision tree algorithm exploits both structural and visual features of web pages (DOM trees). If they are peer subtrees then they are denoted with the same symbol. In the multiple string alignment step (matrix alignment step), the system tries to align the nodes in peer matrix to get a list of aligned nodes. Also, the additional task of recognizing the variant leaf nodes corresponding to basic-typed data is performed. In the pattern mining step, the system takes the list of aligned nodes as an input and detects every repetitive pattern in this list starting with length 1. For each repetitive pattern detected, all occurrences of this pattern are deleted except for the first one to facilitate the further process. The output of this step is a modified list of nodes with no repetitive patterns. In the optional merging step, the system recognizes the nodes as optional if a node disappears in some columns of the matrix and groups nodes according to their occurrence vector.

C. *Schema Detection*

Website structure detection involves two tasks: identifying the schema and defining the template for each type constructor of this schema. The task of schema detection is to recognize tuple type, order of set type and optional data. The system traverses Fixed/Variant pattern tree P from the root downward and nodes are marked as k-order or k-tuple. The identified tuple nodes are marked by angle brackets <>. Then the schema S can be obtained by excluding all tag nodes which have no types. Once the schema is identified, the template of each type can be discovered by concatenating nodes without any type marked.

```
Function MarkTheOrder(P)
  if (P is a leaf node) then
     if (P is fixed) then return 0; else return 1;
  end if
  if ( P has only one child C) then
      if ( P is already marked as a type constructor) then
          Mark the type as 1-order;
      Return MarkTheOrder(C);
  end if
  k = 0;
  for each child C of P do
        k+ = MarkTheOrder(C);
  end for
  if (k==0) then return 0;
  if (P is a virtual node) then
     Mark P as k-order;
  else
     Mark P as k-tuple;
  end if
  return 1;
```

Figure 4. Algorithm for identifying orders of type constructors in pattern Tree

IV. CONCLUSION AND FUTURE WORK

In this paper, a classifier based approach is proposed for peer node recognition to extract the template from deep web pages automatically using DOM tree information, text contents and visual information for training features. Peer node recognition is improved by visual information and also visual information helps detecting template blocks as well as fixed blocks. Also, it is found that machine learning is suitable for all websites. The J48 decision tree is efficient because it is time saving.

The research shows that such a classifier-based approach can be used for more accurate and efficient data extraction. Using more features or multiple classifiers might be an interesting direction to study and explore.

REFERENCES

[1] M. Kayed, C.-H. Chang. "FiVaTech: Page-Level Web Data Extraction from Template Pages", IEEE TKDE, vol. 22, no. 2, pp. 249-263, Feb. 2010.

[2] H. Zhao, W. Meng, Z. Wu, V. Raghavan, C. T. Yu. "Fully automatic wrapper generation for search engines". WWW 2005: 66-75

[3] H. Zhao, W. Meng and Z. Wu, V. Raghavan, C. Yu. "Automatic Extraction of Dynamic Record Sections From Search Engine Result Pages". VLDB, pp.989-1000, 2006

[4] K. Simon, G. Lausen: "ViPER: augmenting automatic information extraction with visual perceptions". CIKM 2005: 381-388

[5] W. Liu, X.-F. Meng, W.-Y. Meng. "ViDE: A Vision-based Approach for DeepWeb Data Extraction. Transactions on Knowledge and Data Engineering, IEEE, 2007

[6] A. Arasu and H. Garcia-Molina, "Extracting structured data from Web pages," in Proceedings of the 2003 ACM SIGMOD international conference on management of data San Diego, California: ACM,2003.

[7] V. Crescenzi, G. Mecca, P. Merialdo. "RoadRunner: Towards Automatic Data Extraction from Large Web Sites," in Proceedings of the 27th International Conference on Very Large Data Bases: Morgan Kaufmann Publishers Inc., 2001

[8] Y. Zhai and B. Liu. "Web data extraction based on partial tree alignment," in Proceedings of the 14th international conference on World Wide Web Chiba, Japan: ACM, 2005.

[9] J. Wang, F. H. Lochovsky. "Data extraction and label assignment for web databases". WWW 2003: 187-196

[10] B. Liu and Y. Zhai. "NET - A System for ExtractingWeb Data from Flat and Nested Data Records." WISE Conference, 2005.