4M1-IOS-3c-2

FiVaTech2: A Supervised Approach to Role Differentiation for Web Data Extraction From Template Pages

Chia-Hui Chang[*1] Chih-Hao Chang[*1] Mohammed Kayed[*2]

[*1] National Central University, Jhongli, Taiwan [*2] Math. Department, Faculty of Science Beni-Suef University, Egypt

A huge amount of consolidated information on the World Wide Web are embedded in HTML pages as they are generated dynamically from databases through some search form. This paper proposes a page-level web data extraction system FiVaTech2 that extracts schema and templates from these template-based web pages automatically. The proposed system, FiVaTech2, is an extension to our previously page-level web data extraction system FiVaTech. FiVaTech2 uses a machine learning (ML) based method which compares HTML tag pairs to estimate how likely they present in the web pages. We use one of the ML techniques called J48 decision tree classifier and also use image comparison to assist templates detection. Each HTML tag in the web page has several features that can be divided into the three types: visual information, DOM tree information, and HTML tag contents. Our experiments show an encouraging result for the test pages when combinations of the three types of tag features are used. Also, our experiments show that FiVaTech2 performs better and has higher efficiency than FiVaTech.

1. Introduction

The explosive growth and the popularity of the World Wide Web make it the repository of information for all people around the world. Although this growing of the World Wide Web, it doesn't guarantee effectively accessing to those information. Most of the web sites don't provide the API to be mashed up or the RSS to be subscribed, so users of these sites have to do a lot of browsing or copy&paste in order to get the required information. This cost of browsing is becoming noticeable with the Deep Web (deepnet or invisible web), which contains magnitudes more and valuable information than the surface Web. Web pages in the Deep Web are generated dynamically from databases and often present in some template showing consistent view of their search result. Since these web pages are not meant to be processed by programs, developing a wrapper is useful for value-added services and other information integration systems.

There are extensive studies of information extraction (IE) from World Wide Web. Chang et al. [2] surveyed these works and classified them based on the automation degree into four different classes: manually constructed [9], [10], supervised [11], [12], [13], [14], semisupervised [1], [15], and unsupervised systems [3],[16], [17], [18], [19]. Manually constructed systems require programmers to deduct the extraction rules but are costly and difficult to scale up. Supervised systems require less user skills to label sample pages for these systems to induce the extraction rules. Semi-supervised Systems do not require users to label any sample pages but require post-processing from the users to choose the pattern and indicate the data to be extracted, while unsupervised systems automatically generate the wrappers without any user interventions and receive a lot of attentions.

Contact: Chia-Hui Chang, National Central University, No. 300, Jhongda Rd., Jhongli City, Taiwan, Tel: +886-3-4227151 ext 35302, Fax: +886-3-4222681, Email: chia@csie.ncu.edu.tw

Most technologies for unsupervised web data extraction analyze HTML source code or DOM (Document Object Model) tree of the input to find common template and their schema. For example, FiVaTech [3] is an unsupervised web data extraction system, which uses tree template to model the generation of dynamic web pages. In order to deduce the schema and templates for each individual Deep Web site, Fivatech applies tree matching for peer node recognition, tree alignment for missing and optional nodes recognition, and mining techniques for repetitive pattern detection. Since commercial web pages uses a lot decorating HTML tags or CSS for presentation, many studies have exploited visual information for web data extraction. Example of such studies are ViNTs [4], MSE [5], ViPER [6], [7], and ViDE [8]. In ViNTs and MSE, visual information is used to capture the content lines. ViPER uses visual information to identify potential repetitive patterns and ViDE obtain the visual block tree to extract data records and data items from the web pages.

While
unsupervised information extraction use annotation-free input pages for wrapper induction, we can still make use of trained models for differentiating tag roles for unsupervised wrapper induction. In this paper, we improve FiVaTech by developing a new version called FiVaTech2 which exploits visual information and the subtree attributes of the DOM tree for peer node recognition. The visual information (including IsLeaf, Left, Top, Height, Width, Parent, Page, Value, Children, ChildHeight, and ClassAttr, etc.) is obtained from the web browser (Microsoft Internet Explorer). The training data are prepared from eight web sites (ACM, Amazon, Bing, Columbia university library, DBLP, Delicious, eBay, Google, IEEE explorer, UCLA library, American Airline, Altavista, Buy.com, CNet download, Netflix) to build a classifier for peer node recognition.

That is, instead of simple tree matching algorithm between two subtrees in FiVaTech [3], FiVaTech2 uses decision tree classifier to identify peer nodes between two DOM trees.

The rest of the paper is organized as follows. Section 2 reviews the related works. Section 3 introduces our earlier approach FiVaTech. Section 4 provides the detail of our new approach FiVaTech2. Section 5 describes the experiments. Section 6 concludes our work.

2. Related works

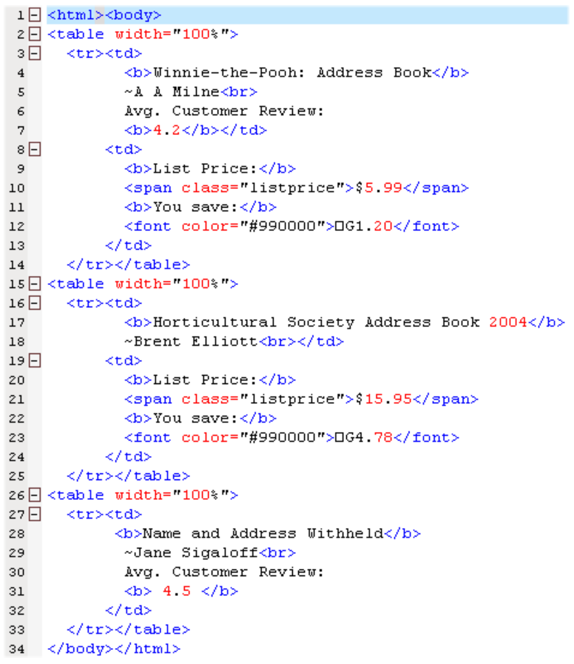
In this section, we describe current work related to ours including the approaches that use DOM tree information to construct the wrapper, the approaches that use visual information to help extracting data records, and the approaches that are page-level extraction systems.

2.1 Approaches using DOM tree information

Many approaches on Web information extraction operate on DOM tree structure. MDR [18] analyzes the child nodes under each parent node to find generalized nodes and data regions by enumerating possible combinations of child nodes. It uses string edit distance to compute the similarity between tag sequences of two generalized nodes. However, the goal of MDR is to identify data records and does not align the data items in each data record. Meanwhile, due to missing and noise information, it may find wrong combination of subtrees, i.e. incorrectly boundary of a data record. DEPTA [19] deals with the problem by using visual gaps between data records. Besides, it incorporates partial tree alignment technique to align data fields of multiple data records. NET [21] extends DEPTA by supporting nested records extraction. ViPER [6] uses primitive tandem repeats and visual context information for record segmentation to enhances the generalized nodes concept. This provides a better subtree comparing method than MDR that allows consecutive data records with various lengths. DeLa [20] builds suffix trees to detect C-Repeated patterns in web page string and its algorithm is able to extract the nested objects. FiVaTech [3] also relies on tree matching score for subtree comparison, however, the bigger goal is to find the schema and template for the whole page rather than record-level schema.

2.2 Approaches using visual information

Some approaches improve and extend these works by using the visual information to extract web data. ViNTs [4] and MSE [5] use visual content features on a browser to identify candidate content line. ViPER [6] uses visual information for global multiple sequence alignment. Although visual information is used, these approaches still use HTML tag structure as primary information for similarity calculation. The major players that rely on visual information would be ViDE [8] and WDRE [7]. ViDE obtains the visual presentation and transform it into a visual block tree. Its main visual features include position features, layout features, appearance features, and content features which can be obtained from web page layout (location, size, and font). In general these approaches use the tree edit distance



```
<html><body>
<table width="100%">
  <tr><td>
        <b>Winnie-the-Pooh: Address Book</b>
        ~A A Milne<br>
        Avg. Customer Review:
        <b>4.2</b></td>
      <td>
        <b>List Price:</b>
        <span class="listprice">$5.99</span>
        <b>You save:</b>
        <font color="#990000">□G1.20</font>
      </td>
  </tr></table>
<table width="100%">
  <tr><td>
        <b>Horticultural Society Address Book 2004</b>
        ~Brent Elliott<br></td>
      <td>
        <b>List Price:</b>
        <span class="listprice">$15.95</span>
        <b>You save:</b>
        <font color="#990000">□G4.78</font>
      </td>
  </tr></table>
<table width="100%">
  <tr><td>
       <b>Name and Address Withheld</b>
        ~Jane Sigaloff<br>
        Avg. Customer Review:
        <b> 4.5 </b>
      </td>
  </tr></table>
</body></html>
```

Figure 1: An example HTML page

to find the similarity of the HTML tag tree nodes as well as visual information cue.

2.3 Page-level Extraction Systems

EXLAG [16] and RoadRunner [17] are page-level unsupervised web data extraction systems. RoadRunner extracts data by comparing a pair of web pages to induce the template. It includes three steps: A (Align), CM (Collapse under Mismatch), and E (Extract). It provides backtracking mechanism if optional or iterated tags are found. EXLAG extracts data by analyzing equivalence classes. dTokens (Differentiating Tokens) are aggregated in equivalence classes if they have the same occurrence frequency on all input web pages. Large and frequently equivalence classes (LFEQs) are extracted for template generation. While EXALG and RoadRunner operates on HTML tags, FivaTech manipulate DOM trees in order to better differentiate roles of subtrees with the same tag names.

3. FiVaTech

In this section, we briefly describe FiVaTech, a page-level Web data extraction system from template pages. Before we start, we shall introduce the page generation model which encode data instances with tree template to form a page.

3.1 Problem definitions

All data instances of a web site shall conform to a common schema which can be defined as follows.

Definition 1: (Structured data) A data schema can be of the following types:

- A basic type β represents a string of tokens which are basic units of text.

- If τ_1,τ_2,. . . ,τ_k are types, then their order list ¡τ_1,τ_2,. . . ,τ_k¿ also forms a type τ. We say the type τ is constructed from the types τ_1,τ_2,. . . ,τ_k using a type constructor of order k. An instance of the k-order τ is of the form ¡$x_1, x_2, ..., x_k$¿ where $x_1, x_2, ..., x_k$ are instances of types τ_1,τ_2,. . . ,τ_k respectively. The type τ is called

  1. a tuple, denoted by ¡k¿$_\tau$, if the cardinality (the number of instances) is 1 for every instantiation.
  2. an option, denoted by $(k)?_\tau$, if the cardinality is either 0 or 1 for every instantiation.
  3. a set, denoted by $\{k\}_\tau$, if the cardinality is greater than 1 for some instantiation.
  4. a disjunction, denoted by $(\tau_1|\tau_2|...|\tau_k)_\tau$, if all $\tau_i(i = 1, ..., k)$ are options and the cardinality sum of the k options: τ_1 to τ_k equal to 1 for every instantiation of τ.

The process of embedding a data instance into the template can be viewed as an encoding process, while wrapper induction is an reverse engineering which decodes from the pages to its template and data schema. Thus, the problem of wrapper induction can be formulated as follows.

Definition 2: (Wrapper Induction) Let $\lambda(\mathrm{T}_\Omega, \mathrm{D})$ denotes the generation model of some Web pages at time t, where T_Ω denotes the templates for schema Ω and D denotes its extracted data. The problem of *page-level* wrapper verification is to decide whether a new Web page P at time t' has been changed from its generation model $\lambda(\mathrm{T}_\Omega, \mathrm{D})$. We call this problem a *record-level* wrapper verification if Ω is simply a set of k-tuples.

In this paper, we assume that all of the peer nodes must be in the same DOM tree level which is not true for all Web sites.

we adopt FivaTech, a page-level, unsupervised wrapper induction approach which merges the input DOM trees into a pattern tree. A pattern tree removes duplicate occurring patterns of set types and contains one representation for each data types (tuple, option, disjunction, etc). As an example, Figure 2 shows a pattern tree which contains the merged DOM tree and detected schema, where we have a set, two tuples, two options and five basics.

3.2 FiVaTech Tree Merging

There are two main phases in FiVaTech. The first phase is merging input DOM trees to construct the fixed/variant tree, and the second phase is detecting the schema and the template of a Web site based on the constructed pattern tree.

FiVaTech merges similar subtrees on the same level from the inputted DOM trees. The system considered only subtrees on the same level to simplify the merging problem. The tree merging algorithm includes four steps: peer node recognition, matrix alignment, pattern mining, and optional node detection.

1. In the peer node recognition step, a modified tree edit distance is designed to calculate a matching score for two nodes with the same tag[*1] name such that set data with various occurrences still have high similarity.

2. In the second step, the child nodes will fill up a matrix such that all peer child nodes (similar subtrees measured in the previous step) will be denoted with the same symbol. The matrix alignment algorithm then traverses the matrix to obtain an aligned peer matrix.

3. In pattern mining step, the algorithm discovers repetitive patterns and merge them to deduce the aligned list.

4. Finally, in the last step, optional node merging, the algorithm detects optional nodes based on the occurrence vectors. If a set of adjacent nodes have the same occurrence vectors, they will be grouped as optional. If a set of adjacent optional nodes have complement occurrence vectors, they will be grouped as disjunction.

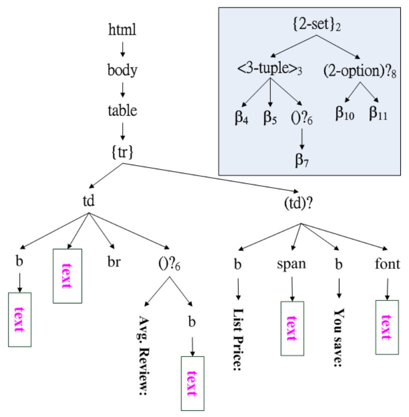


Figure 2: A merged pattern tree and the corresponding schema representation for Figure 1

3.3 Schema Detection

In this phase, FiVaTech detects the structure of the web site which includes identifying the schema and defining the template for each type constructor of this schema. By the end of the previous phase, basic type, set type and optional type are already identified. The remaining task for schema detection in this phase is to recognize tuple type as well as the order of the set type and the optional data.

FiVaTech traverses the fixed-variant pattern tree from the root downward and marks nodes as k-order (if the node is already marked as some data type) or k-tuple. For nodes with only one child and not marked as set or optional types, there is no need to mark it as 1-tuple (otherwise, there will be too many 1-tuples in the schema), thus the system simply traverses down the path to discover other type nodes. For nodes with more than one branch (child), the system will mark them as k-order if k children contain a data type.

[*1] Two nodes with different names are inherently not peer.

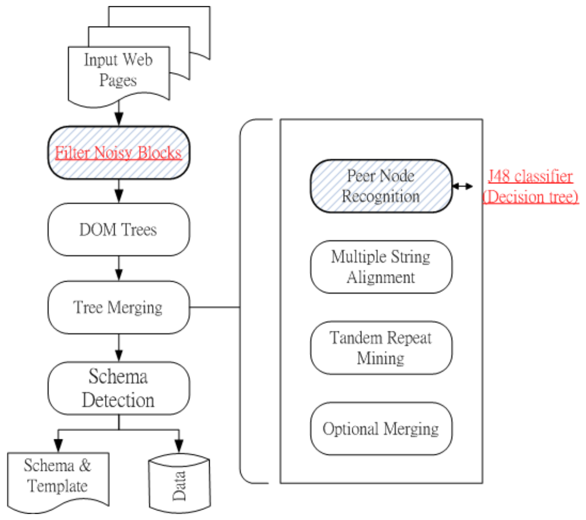
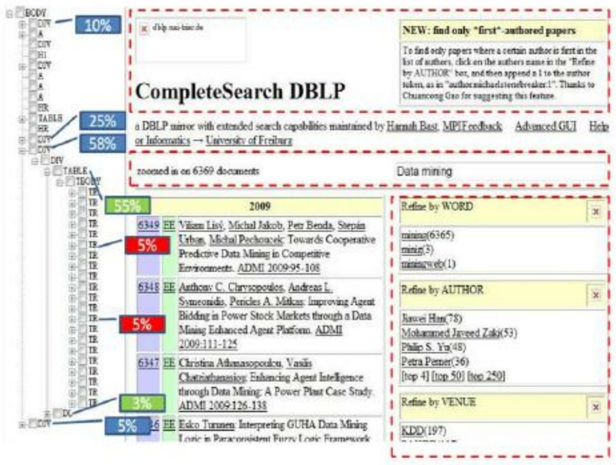


Figure 3: FiVaTech2 Architecture



dblp.uni-trier.de

CompleteSearch DBLP

NEW: find only "first"-authored papers

To find only papers where a certain author is first in the list of authors, click on the authors name in the "Refine by AUTHOR" box, and then append a 1 to the author token, as in "author:michaelstonebreaker:1". Thanks to Chuancong Gao for suggesting this feature.

a DBLP mirror with extended search capabilities maintained by Hannah Bast, MPI Feedback Advanced GUI Help
or Informatics → University of Freiburg

zoomed in on 6369 documents

Data mining

2009

6349 EE Viliam Lisý, Michal Jakob, Petr Benda, Stepán Urban, Michal Pechoucek: Towards Cooperative Predictive Data Mining in Competitive Environments. ADMI 2009:95-108

6348 EE Anthony C. Chrysopoulos, Andreas L. Symeonidis, Pericles A. Mitkas: Improving Agent Bidding in Power Stock Markets through a Data Mining Enhanced Agent Platform. ADMI 2009:111-125

6347 EE Christina Athanasopoulou, Vasilis Chatziathanasiou: Enhancing Agent Intelligence through Data Mining: A Power Plant Case Study. ADMI 2009:126-138

6346 EE Esko Turunen: Interpreting GUHA Data Mining Logic in Paraconsistent Fuzzy Logic Framework

Refine by WORD

mining(6365)
minig(3)
miningweb(1)

Refine by AUTHOR

Jiawei Han(78)
Mohammed Javeed Zaki(53)
Philip S. Yu(48)
Petra Perner(36)
[top 4] [top 50] [top 250]

Refine by VENUE

KDD(197)

Figure 4: A Web page from DBLP and its DOM tree.

Finally, the schema tree S can then be obtained by excluding all of the tag nodes that have no types.

Note that FiVaTech uses tree edit distance to measure the similarity among nodes with the same tag in the same level of the inputted Web pages, which exploits only structural information to measure the similarity. Since visual information is recommended and important for similarity measure, we propose a classifier-based approach for peer node recognition.

4. FiVaTech2

In this section, we present the details of our approach which is an improvement to FiVaTech. As shown in Figure 3, our approach includes three modifications. Given some DOM trees as input, we filter template blocks by image comparrison. Third, peer node recognition is enhanced by a J48 classifier during tree merging.

```
1 Procedure filteringTemplateBlocks (TreeNode)
2   biggest = the child with the biggest area are;
3   if ( the area of biggest < 40% ) then return NULL;
4   for each child in TreeNode
5     if (child != biggest) then
6       Compare and remove all equal images (subtrees)
        corresponding to the child tag from all DOM trees
7     endif
8   filteringTemplateBlocks (biggest)
```

Figure 5: The filteringTemplateBlocks algorithm

4.1 Filtering Out Template Blocks in the Inputed DOM Trees

The Deep Web usually contains two types of blocks in the generated Web pages: template data blocks and data rich blocks. Template data blocks are the frames/sections of the Web pages that contain template data such as advertisements, navigational panels and so on. Data rich blocks are the frames/sections of the Web pages that contain relevant data of interest to the user. Although template blocks can be detected by FiVaTech through recursive comparison of peer nodes from root, the process could be quite time consuming. To improve the efficiency, we propose an image-based step to filter out template blocks before applying the peer nodes recognition step for tree merging.

Our algorithm filteringTemplateBlocks has two main assumptions. First, template blocks for various pages of a Web site are displayed with the same content. Not only does the renderred images look the same, the tags that correspond to such template blocks also co-located in the same path of the DOM trees. Second, the area for a data rich block often occupies the biggest area in the whole Web page. Based on the first assumption, we can remove template subtrees in the preprocessing step. However, subtrees with the same images in a data rich block are usually not template.

As shown in Figure 5, the algorithm recursively traverses one of the inputted DOM trees from the root downward and checks for the existence of some child node c with percentageArea(c) > 40% where percentageArea(c) is defined as the percentage of the image area corresponds to node c to the whole area of the displayed page):

$$percentageArea(c) = \frac{nodeArea(c)}{nodeArea(< Body >)}\%.$$

If all child nodes have percentageArea less than 40%, the algorithm stops (line 1) or the algorithm identifies the child node with the biggest percentageArea value (line 2). The algorithm then keeps the biggest node and all remaining children nodes that have no identical subtrees in the other DOM trees (line 3-6).

As an example, we apply our algorithm with the DOM tree in Figure 4 to describe how it proceeds. The algorithm starts at the root node (¡body¿) and identifies the fifth ¡DIV¿ node with 58% percentageArea to be the biggest child. The algorithm then seaches for each node the identical subtrees in other inputted DOM trees and remove the subtrees if every DOM tree has identical image. For the biggest node, the algorithm recursively calls itself to process next level and find the only child node at this level (¡DIV¿ at level 2) as the biggest node and so on. The algorithm then stops at the node ¡TBODY¿ at level 4 because all of its children have percentageArea values less than 40%.

4.2 Peer Nodes Recognition

In this section, we describe the revised algorithm to identify peer nodes (similar subtrees) in the inputted DOM trees. The main idea is to use the decision tree algorithm to judge whether two input subtrees are peer nodes or not. During the peer node recognition step, we apply decision tree algorithm to compare two nodes (subtrees) at the same level in the input DOM trees.

In this paper, we use the J48 algorithm which is available with the Weka package. In order to train the J48 classifier model, we exploit both structural and visual features of Web pages (DOM trees). The visual features is obtained from Web browsers (e.g., Microsoft Internet Explorer) by calling the APIs of the browser. Each node of the DOM tree has a corresponding rectangular image displayed on the browser. The two-dimensional coordination (left,top) is used to represent the location of each image. In addition to the (left,top) point, we also have the width and height of the image and the following features.

- PageID: an unique identifier denoting where the subtree originates.
- Parent: the parent node in the DOM tree.
- TextContent: the text contents within the subtree.
- NoChildren: number of child nodes in the subtree.
- ChildHeight: The depth of the subtree of the current node.
- LeafNode: whether the node has children or not (Boolean feature).
- ClassAttr: The class name of the HTML tag.
- Path: The numbered path of the DOM tree node. The partial path includes the tag name and its parent tag name. Each tag name is followed by a number representing iting location in the parent node (from left to right starting at 1). For example, the path of the node ¡Table¿ in Figure 4 has path ¡DIV5¿¡DIV1¿.
- Token: the number of tokens in leaf text content.
- Digit: the number of digits in leaf text content.
- Letter: the number of letters in leaf text content.
- UpperCase: the number of capitalized letter in leaf text content.
- LowerCases: the number of lower cas

The features used to train the J48 decision tree with two nodes node1 and node2 are shown in Table 1. The appropriate values of the attributes are based on our empirical results. For example, the value "Both" of the attribute Node1IsLeaf means that both node1 and node2 are leaves. As another example, the value "Sim" of the attribute SameChildren means that the percentage of the size difference of the two nodes is less than 10%. Also, the value "Empty" means that the contents of the two nodes are empty. Our experiments show that these values are useful and give good results.

5. Experiments

We conduct two experiments to measure the performance of the J48 algorithm on peer node recognition and overall performance of FiVaTech2 for wrapper induction, respectively. We collect HTML pages from 15 web sites for the first experiment on peer node recognition and another 15 websites for wrapper induction.

5.1 Performance of Peer Node Recognition

5.1.1 Learning Curve with Various Training Data Size

First, we show the performance of peer node recognition when various number of training sites are used. We fix 5 web sites as the testing set and measure the performance of the models trained from n (1..10) web sites. For each n, we prepare ten data sets S_n^i (i=1..10), each with n web sites chosen in a round robin way. We then conduct ten rounds to build prediction classifiers for each data set and average the accuracy of each model to measure the performance of n training sites. That is,

$$A_n = \frac{\sum_{i=1}^{10} Acc(S_n^i)}{10}.$$

For example, to calculate the accuracy A_2 for two training web sites taken from the 10 training sites $\{s_1, ..., s_{10}\}$, we calculate the accuracy $Acc(S_2^i)$ for each one of the 10 sets $\{s_1, s_2\}$, $\{s_2, s_3\}$, $\ldots$, $\{s_{10}, s_1\}$, and then calculate the average as A_2. As shown in Figure 6, the accuracy on the 10 training sites increases from 0.89 to 0.98 with more training data (a typical learning curve for machine learning algorithm). However, the accuracy on the 5 testing sites achieves at 0.87 with 6 training sites and improves no more thereafter.

We can see a similar performance in terms of training node pairs. The given 10 training sites with 0.87 million node pairs (instances) are used to sample n examples as training set for various n from 20..100, 200..1000, 2000..10000, 20000..100000, 200000..800000. Again, we evaluate the accuracy for each n by averaging the accuracy of ten rounds. Figure 7 shows the accuracy of the training set (0.87 million instances) and testing set (0.4 million instances) at various number of training instances. The accuracy achieves 0.87 when 3000 node pairs are used as training set and remains around this value.

Table 1: Features for Peer Node Recognition

Attributes	Values	Description
Visual Information:		
SameLeft	Yes,No	Whether the “Left” of the two images correspond to the two nodes are equal (Yes) or not (No).
SameTop	Yes,No	Whether the “Top” of the two images corresponds to the two nodes are equal (Yes) or not (No).
SameHeight	Yes,No	Whether the heights of the two images correspond to the two nodes are equal (Yes) or not (No).
SameWidth	Yes,No	Whether the widths of the two images correspond to the two nodes are equal (Yes) or not (No).
DOM Tree Information:		
Node1IsLeaf	Yes,No,Both	Whether Node1 is a leaf (Yes) or not (No). “Both” means both Node1 and Node2 are leaves.
Node2IsLeaf	Yes,No,Both	Whether Node2 is a leaf (Yes) or not (No). “Both” means both Node1 and Node2 are leaves.
SameParent	Yes,No	Whether the two nodes have the same parent tags (Yes) or not (No).
SamePage	Yes,No	Whether the two nodes in the same page (Yes) or not (No).
SameChildren	Yes,No,Sim	Whether the total number of nodes in the two subtrees are equal (Yes). If the difference is less than 10%, we assign a value of “Sim”, otherwise we assign a value of “No”.
SameChildHeight	Yes,No	Whether the two nodes have the same depth (Yes) or not (No).
SameClassAttr	Yes,No,Empty	Whether the two tag nodes have the same class attributes (Yes) or not (No). “Empty” value is used when any one has no class attributes.
SamePath	Yes,No	Whether the two nodes have the same paths (Yes) or not (No).
TreeEditDistance	1%, 10%, 20%, ..., 100%	The discretized percentage of tree edit distance of two nodes
HTML Tag Contents:		
SameContent	DateTimeFormat, EmailFormat, UrlFormat, IPFormat, CurrencyFormat, DecimalFormat, PhoneFormat, StringEditDistance (1%, 10%, 20%, ...,100%), No	When two nodes have the same content category of “DateTimeFormat”, “EmailFormat”, “UrlFormat”, “IPFormat”, “CurrencyFormat”, “DecimalFormat” “PhoneFormat”, the corresponding category value is used. If the content category is string type, the percentage of the string edit distance are discretized. If the content categories are not the same, we assign a value of “No”.
DiffTokens	Yes, No, Empty, 1%, 10%, 20%, ..., 100%	Whether the two nodes have the same number of Tokens (Yes) or discretized percentage of different token.
DiffDigits	Yes, No, Empty, 1%, 10%, 20%, ..., 100%	Whether the two nodes have the same contents (Yes) or not (No). “Empty” means both are empty. The others are the percentage of different digit character number of two nodes.
DiffLetters	Yes, No, Empty, 1%, 10%, 20%,...,100%	Whether the two nodes have the same contents (Yes) or not (No). “Empty” means both are empty. The others are the percentage of different letter character number of two nodes.
DiffUpperCase	Yes, No, Empty, 1%, 10%, 20%, ..., 100%	Whether the two nodes have the same contents (Yes) or not (No). “Empty” means both are empty. The others are the percentage of different upper case character number of two nodes.
DiffLowerCase	Yes, No, Empty, 1%, 10%, 20%, ..., 100%	Whether the two nodes have the same contents (Yes) or not (No). “Empty” means both are empty. The others are the percentage of different lower case character number of two nodes.

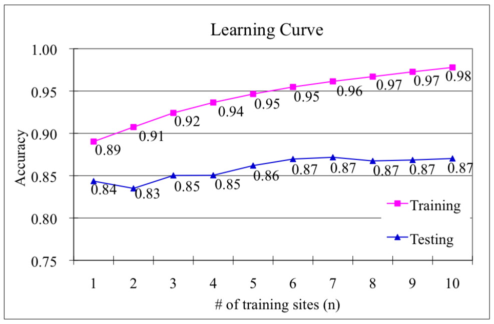


Figure 6: The accuracy of different number of training sites

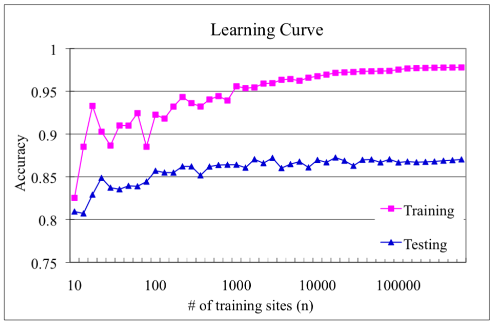


Figure 7: The accuracy of various training node pairs

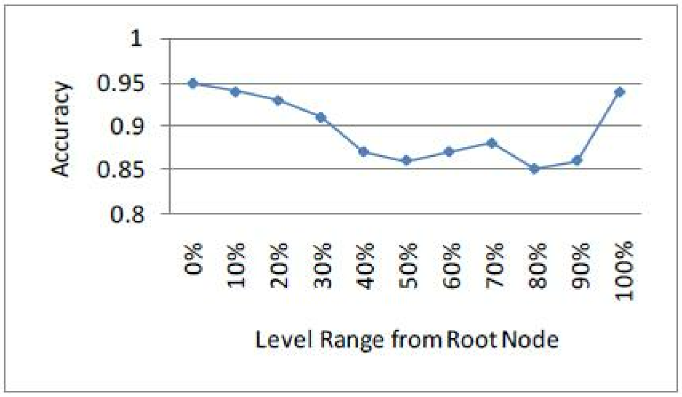


Figure 8: Accuracy at different DOM tree levels

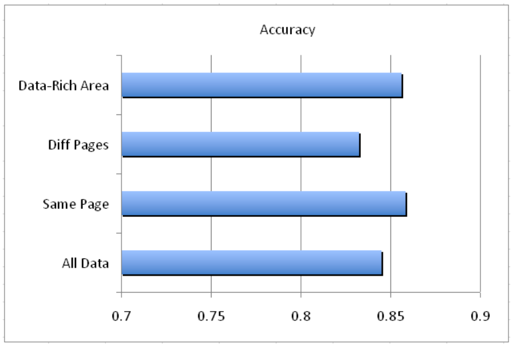


Figure 9: The accuracy of different data regions

5.1.2 Performance for Various Depth and Data Regions

We further divide the test examples into subsets to see if the performance is influenced by various levels in the DOM tree or whether the node pairs come from the same page or whether they come from data rich regions. First, we examine the accuracy of peer node recognition from various levels. Figure 8 shows the accuracy of the J48 classification at each level in the DOM tree (normalized between level 0 to 100%). As shown in the figure, the first and deepest level have higher accuracy than the middle level.

Next, we also examine the accuracy of peer recognition on node pairs from different pages as well as node pairs from data-rich regions. As shown in Figure 9, the performance of peer node recognition is higher if the node pairs come the same pages or data-rich area.

- Single Page: exclude node pairs which come from different pages
- Diff Pages: exclude node pairs which come from the same page
- Data-rich Area: exclude node pairs which are in template area

5.2 Performance on Wrapper Induction

In the second experiment, we compare the schema generated by FiVaTech2 and that generated by FiVaTech. We use additional 15 web sites in this experiment (different than the 15 sites used in the above experiment) and deduce their schema manually. As shown in Table 2, we calculate the number of basic type (denoted by B_m), optional type (denoted by O_m), and set type (denoted by S_m), then we verify the number of basic items extracted correctly (denoted by C_1 and C_2) from the two systems. We use recall=Ec/Nt and precision=Ec/Et to measure the performance of these systems, where Ec is the total number of correctly extracted basic items, Et is the total number of basic items which are extracted by the system, and Nt is the total number of basic items to be extracted. Out of 83 manually extracted basic types, FiVaTech2 correctly extracts 78 and FiVaTech correctly extracts 79. However FiVaTech detects 116 basic items (i.e., FiVaTech extracts 33 incorrect basic items), while FiVaTech2 only detects 89 basic items (i.e., FiVaTech2 extracts 6 incorrect basic items). The experimental results show that FiVaTech2 has better precision but worse recall than FiVaTech.

In terms of efficiency, we measure the average time spent on wrapper induction by FiVaTech2 and FiVaTech on the 15 web sites. As shown in Table 3, on a computer Intel Core 2 Duo T7300 with 4GB ram, the time consumed by FiVaTech2 is about 20 seconds on average (shown in the second column) and the time consumed by FiVaTech is 38 seconds on average (shown in the third column). The last column shows the percentage of the time taken by FiVaTech2 to the time taken by FiVaTech. As shown in the table, FiVaTech2 is more efficient than FiVaTech (less than half of the time taken by FiVaTech).

Table 2: Performance of comparison between FiVaTech2 and FiVaTech

	B_m	O_m	S_m	A_1	O_1	S_1	C_1	A_2	O_2	S_2	C_2
Site	Manual			FiVaTech2				FiVaTech			
GenSource	7	0	2	8	0	2	6	6	0	1	6
NACM	8	1	1	9	4	1	8	13	0	2	8
NAMI	6	1	1	8	4	2	6	6	2	1	6
OneStop	5	1	1	10	6	1	5	28	0	0	5
ShelbyVille	3	1	1	3	1	1	3	3	0	1	3
Overture	6	0	1	6	0	2	6	6	0	2	6
Budget.state	8	1	1	8	4	2	8	15	2	2	8
Queensland	6	0	1	6	3	2	6	6	0	2	6
Wacotrib	4	1	1	5	2	1	4	4	2	1	4
Atmoz	7	1	3	7	5	5	7	6	1	3	6
Execgroup	4	0	1	3	1	2	3	4	0	3	4
Dog.com	4	0	1	4	0	1	4	5	0	2	4
Lgiftbazaar	3	0	1	3	0	1	3	3	0	1	3
TheGoodWeb	7	1	2	4	4	2	4	6	2	2	6
SEHSC	5	0	1	5	0	1	5	5	0	1	4
Total	83	8	19	89	34	26	78	116	9	20	79
Recall							93.97%				95.18%
Precision							87.64%				68.10%

Finally, we analyze the factors that affect the performance of FiVaTech2. We divide the labeled instances into three different groups including "Visual", "Dom", and "Content" information. "Visual" information includes the attributes "SameLeft", "SameTop", "SameHeight", and "SameWidth"; "Dom Tree" information includes the attributes "Node1IsLeaf", "Node2IsLeaf", "SameParent", "SamePage", "SameChildren", "SameChildHeight", "SameClassAttr", "SamePath", and "TreeEditDistance"; while "Content" information includes "SameValue", "DiffTokens", "DiffDigits", "DiffLetters", "DiffUpperCase", and "DiffLowerCase". We then compare the extraction performance of data records using various combinations. Table 4 shows the performance of FiVaTech2 when various combinations of these three types of features are used. We use recall=Ec/Nt and precision=Ec/Et to measure the performance of our system, where Ec is the total number of correctly extracted data records, Et is the total number of records extracted, and Nt is the total number of data records contained in all web pages of a data set. As shown in the table, we can observe that using of DOM attributes give better results. Meanwhile, HTML tag content has only values with the leaf nodes while all internal nodes have no contents. The visual information is not sufficient for text node, so the F-measure is not the best. The overall attributes have the best F-measure in the experiment.

6. Conclusion and Future Works

In this paper, we propose a classifier based approach for peer node recognition to extract template from the deep web pages automatically. The J48 decision tree uses both DOM tree information and text contents as well as visual

Table 3: A time-based comparison between FiVaTech2 and FiVaTech

Time	FiVaTech2	FiVaTech	Ratio
GenSource	3	14	21.4%
NACM	9	15	60.0%
NAMI	13	42	30.9%
OneStop	3	35	8.6%
ShelbyVille	8	40	20.0%
Overture	1	9	11.1%
Budget.state	6	3	200.0%
Queensland	4	16	25.0%
Wacotrib	3	15	20.0%
Atmoz	17	60	28.3%
Execgroup	60	105	58.1%
Dog.com	2	7	28.6%
Lgiftbazaar	2	5	40.0%
TheGoodWeb	3	6	50.0%
SEHSC	130	160	81.3%
Average	20	38	45.6%

Table 4: Performance comparison of FiVaTech2 at different types of features

Attributes	Recall	Precision	F-measure
Visual	0.83	0.88	0.874
Dom	0.88	0.89	0.913
Content	0.83	0.81	0.826
Visual +Dom	0.90	0.91	0.951
Dom+Content	0.88	0.90	0.928
Visual+Content	0.84	0.88	0.897
All	0.89	0.90	0.962

information for training features. The visual information improves the peer node recognition and helps detect templates as well as fixed blocks. We found that the training size of 6 web sites can achieve 0.87 accuracy for peer node recognition. The experiments provides the evidence that such a classifier-based approach can be used for more accurate and efficient data extraction. However, 0.87 accuracy is not a satisfactory performance for peer node recognition. More features or multiple classifiers might be an interesting direction to explore.

References

[1] C.-H. Chang, C.-N. Hsu, S.-C. Lui. “IEPAD:Information extraction based on pattern discovery. WWW-10,pp.223-231, 2001

[2] C.-H. Chang, M. Kayed, M. R. Girgis, K. F. Shaalan. “A Survey of Web Information Extraction System. IEEE TKDE(SCI, EI),Vol. 18,No.10,pp.1411-1428, 2006

[3] 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.

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

[5] 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

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

[7] W. Liu, X.-F. Meng, W.-Y. Meng. “Vision-Based Web data records extraction. In: Proc. of the 9th SIGMOD Int’l Workshop on Web and Databases (WebDB 2006). Chicago: ACM Press, 2006.

[8] W. Liu, X.-F. Meng, W.-Y. Meng. “ViDE: A Vision-based Approach for Deep Web Data Extraction. Transactions on Knowledge and Data Engineering, IEEE, 2007

[9] J. Hammer, J. McHugh, and H. Garcia-Molina, “Semistructured Data: The TSIMMIS Experience,” Proc. First East-European Symp. Advances in Databases and iformation Systems (ADBIS), pp. 1-8,1997.

[10] L. Liu, C. Pu, and W. Han. “XWRAP: an XML-enabled wrapper construction system for Web information sources,” in Data Engineering, 2000. Proceedings. 16th International Conference on,2000, pp. 611-621.

[11] D. Freitag. “Information Extraction from HTML: Application of a General Learning Approach,” 1998.

[12] H. F. L. Alberto, R.-N. Berthier, and S. d. S. Altigran. “DEByE - Date extraction by example,” Data Knowl. Eng., vol. 40, pp. 121-154, 2002.

[13] N. Kushmerick. “Wrapper induction for information extraction,” University of Washington, 1997, p. 246.

[14] ME. Califf, RJ. Mooney, ”Relational Learning of Pattern-Match Rules for Information Extraction,” University of Texas at Austin 1998.

[15] C.-H. Chang and S.-C Kuo, ”Olera: semisupervised Web-data extraction with visual support,” Intelligent Systems, vol. 19, pp. 56-64, 2004.

[16] 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.

[17] 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

[18] B. Liu, R. Grossman. Y. Zhai. “Mining data records from Web pages.” KDD-03, 2003.

[19] 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.

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

[21] B. Liu and Y. Zhai. “NET – A System for Extracting Web Data from Flat and Nested Data Records.” WISE Conference, 2005.