Journal of Universal Computer Science, vol. 20, no. 2 (2014), 169-192
submitted: 5/4/13, accepted: 24/11/13, appeared: 1/2/14 © J.UCS

Unsupervised Structured Data Extraction from Template-generated Web Pages

Tomas Grigalis
(Vilnius Gediminas Technical University, Vilnius, Lithuania
tomas.grigalis@vgtu.lt)

Antanas Čenys
(Vilnius Gediminas Technical University, Vilnius, Lithuania
antanas.cenys@vgtu.lt)

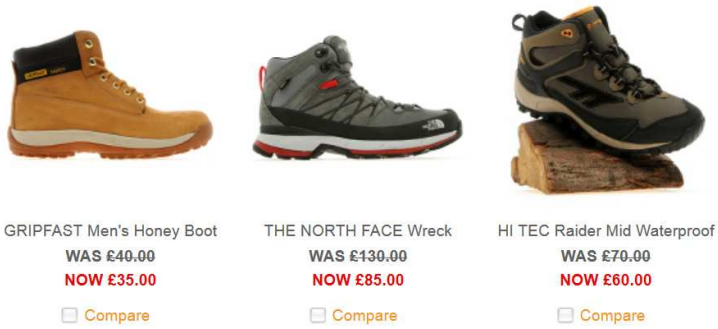
Abstract: This paper studies structured data extraction from template-generated Web pages. Such pages contain most of structured data on the Web. Extracted structured data can be later integrated and reused in very big range of applications, such as price comparison portals, business intelligence tools, various mashups and etc. It encourages industry and academics to seek automatic solutions. To tackle the problem of automatic structured Web data extraction we present a new approach – structured data extraction based on clustering visually similar Web page elements. Our method called ClustVX combines visual and pure HTML features of Web page to cluster visually similar Web page elements and then extract structured Web data. ClustVX can extract structured data from Web pages where more than one data record is present. With extensive experimental evaluation on three benchmark datasets we demonstrate that ClustVX achieves better results than other state-of-the-art automatic structured Web data extraction methods.

Keywords: data extraction, wrapper induction, structured web data, Deep Web
Categories: H.0, H.2.8, H.3.3, H.3.5

1 Introduction

We are witnessing ever increasing amount of structured data available on the Web and, in the same turn, the diversity of the visual structures in which the data is stored [Madhavan et al. 2007]. The prime examples of such structured data is the Deep Web, referring to content on the Web that is stored in databases and served by querying HTML forms [Madhavan et al. 2007]. Another example is tables in Web pages, which contain highly structured data [Cafarella et al. 2008; Elmeleegy et al. 2009]. There is an enormous potential in combining and re-using this data in creative ways [Cafarella and Halevy 2009; Madhavan and Halevy 2009], such as meta-searching, price comparison shopping, business intelligence tools, etc. However, the presence of vast heterogeneous sources and different visual presentation styles of structured Web data pose key challenges to Web search today [Madhavan et al. 2007]: Web pages with structured data are easily understandable by humans, but automatically extracting the same data by computers at Web scale is a very difficult task [Baumgartner and Flesca 2001; Cafarella and Halevy 2009; Cai et al. 2003].

Many solutions are proposed to extract structured Web data. They typically search for repeating patterns in a Web page by calculating the similarity between HTML tag tree nodes [Álvarez et al. 2008; Jindal and Bing 2010; Kayed and Chang 2010; Su et al. 2011; Zhai 2005]. However, such methods do not show consistent results on different benchmark datasets and are prone to errors when dealing with contemporary WEB 2.0 pages. That happens, because modern Web browsers have very high tolerance for an invalid HTML code and thus a lot of Web pages do not obey the W3C HTML specifications. Incorrect HTML code leads to error in constructing HTML tree, which, in turn, hinders structured data extraction process.



a) An example of three boots as products (Data Records) in a Web page

Xstring:	htmlbodydivdivdivspanfonta-Arial,rgb(229,9,9);700
NOW £35.00	/html/body/div[3] /div[1] /span/font/a
NOW £85.00	/html/body/div[3] /div[2] /span/font/a
NOW £60.00	/html/body/div[3] /div[3] /span/font/a

b) A cluster with visually similar price elements

Image1	GRIPFAST Men's Honey Boot	WAS £40.00	NOW £35.00	Compare
Image2	THE NORTH FACE Wreck	WAS £130.00	NOW £85.00	Compare
Image2	HI TEC Raider Mid Waterproof	WAS £70.00	NOW £60.00	Compare

c) Desired extraction results

Figure 1: An example of structured Web data extraction in ClustVX system

Moreover, HTML tree was initially introduced for more readable HTML presentation in the browser rather than description of the semantic data structures in the Web pages [Cai et al. 2003]. Widespread invalid HTML code especially hinder Web data extraction from sophisticated WEB 2.0 pages, where HTML tree is often dynamically modified by various JavaScript codes, new data is added by asynchronous requests to Web server and elements are positioned with Cascading Style Sheets (CSS). The underlying structure of most modern Web pages is complicated more than ever before and has very weak ties to their visually rendered versions displayed on modern Web browsers [Liu et al. 2010]. It is becoming very difficult to successfully extract

structured data from Web pages by just analysing raw HTML code, which is retrieved directly from a Web server. Although there are already some works [Liu et al. 2010; Zhai and Liu 2006; Zhao et al. 2005; Nie and Wen 2007] that utilize the visual features of Web pages to extract data, but these methods often lack proper experimental evaluation on publicly available benchmark datasets or have some limitations which we address in details in related work Section.

In this paper we introduce a conceptually different approach, called ClustVX (Clustering Visually similar XPaths). It first renders given Web page in a contemporary Web browser, then clusters visually and structurally similar repeating Web page elements to identify the underlying structure of embedded structured data object (*Data Records*) and its attributes (*Data Items*). Evaluation of ClustVX on three benchmark datasets demonstrates that the proposed new method is consistent across all of them and outperforms other state-of-the-art approaches.

ClustVX is based on two fundamental observations. First, vast amount of information on the Web is presented using fixed templates and filled with data retrieved from underlying databases [Cafarella et al. 2011]. For example, Fig. 1a shows three Data Records with structured data describing three boots in an online store. The three Data Records are listed according to some unknown to us visual style and the data comes from an underlying database. All three Data Records are structurally similar and are placed in one region of a Web page (*Data Region*) [Zhai and Liu 2006]. Since all these Data Records are placed in one place, each of them has almost the same XPath, i.e. the tag path from the root node in HTML tree to the particular Web page element.

Second, Web page designers optimize visual presentation of structured Data Records for the human reader. So although the templates and underlying data differ from site to site, humans understand them easily by analysing repeating visual patterns on a given Web page [Miao et al. 2009]. The data that has the same semantic meaning is often visualized using the same style [Liu et al. 2010]. Therefore humans, viewing a Web page, are able to comprehend its unique structure quickly and effortlessly and distinguish each Data Record and its unique attributes, such as photos, titles, prices and etc. For example in Fig.1a current prices are red and bold, title is grey, text "compare" is orange and etc.

ClustVX exploits both observations by representing each Web page element with a combination of its XPath and rendered visual features such as font size, font colour and etc. Getting visual features of Web page elements is not a trivial task. It is not enough to simply extract *style* or *class* attributes from HTML code. The information describing the visual style of a Web page element can be stored in many different locations, such as in Cascading Style Sheet (CSS) files, in attributes, in parent elements and etc. Furthermore visual appearance of an element can be modified on-the-fly by JavaScript code. Correct Web page visual rendering can be only achieved by modern Web browsers and their complex rendering engines. ClustVX system employs a modern Mozilla Firefox Web browser to visually render Web pages and to retrieve resulting rendered visual features of Web page elements. For each visible Web page element we encode XPath and visual data into a string called Xstring. Clustering Xstrings allows us to identify visually similar elements, which are located in the same region of a Web page and in turn have same semantic meaning. See

Fig.1b where price elements are clustered together according to their Xstring. Subsequent data extraction leads to a machine readable structured data (see Fig.1c).

Structured data extraction process in ClustVX system is also based on two structural observations about Data Record representation in a Web page. First, a group of Data Records are usually rendered in a contiguous region of a Web page [Zhai and Liu 2006] and are visually similar. Second, a group of Data Records are formed by some child sub trees and at some level have same parent node [Zhai and Liu 2006]. Thus, by calculating longest common prefix of XPaths from each cluster of visually similar Web page elements, we can find the exact locations of Data Records groups (Data Regions) in a page. For a simple example, consider the Fig.1b, where XPaths of clustered price elements are located. First, we find the longest common prefix *(/html/body/div[3]*) of these clustered XPaths. The prefix leads us to the particular region of a Web page, where Data Records are located. Then, the longest common suffix *(/span/font/a*) is Data Items' path in the Data Record. The XPath substring between prefix and suffix (*/div[*]*) is used to segment Data Region into Data Records. All clusters that have the same longest common XPath prefix present one particular Data Region. If there are many Data Regions in one page, ClustVX locates them all.

The rest of the paper is organized as follows: we review the related work in Section 2. In Section 3 we present the architecture of ClustVX system. In Section 4 we present the visual and structural features of Web pages that are used in data extraction. We also review the clustering process of ClustVX system. Then we show how structured data is extracted from a Web page. Section 5 of this paper is used for experimental evaluation of our proposed technique. Section 6 provides a link to a demo implementation of proposed ClustVX system. The last Section 7 is conclusions.

2 Related Work

Structured data extraction systems can be broadly divided into *automatic* and *manual* categories. Supervised learning approaches like Wien [Kushmerick et al. 1997], SoftMealy [Britain et al. 1998], Olera [Chang and Kuo 2004], or LiXto [Baumgartner and Flesca 2001] require some *manual* human effort to derive the extraction rules, while *automatic* data extraction systems like RoadRunner [Crescenzi 2001], DEPTA [Zhai and Liu 2006], VIDE [Liu et al. 2010], VINTS [Zhao et al. 2005] work automatically and need no manual intervention to extract data.

Supervised learning approaches usually builds data extraction wrappers, which, in other words, are programs or strictly defined logical rules used to extract data from Web pages. Supervised learning approaches needs manually labeled sample pages to learn its rules [Kushmerick et al. 1997; Chang and Kuo 2004; Baumgartner and Flesca 2001; Britain et al. 1998; Chang 2001; Laender, B Ribeiro-Neto, et al. 2002]. The main two disadvantages of supervised learning approaches are time consuming manual labeling process and matching already inducted wrappers to constant change of Web sites [Gulhane et al. 2011; Kushmerick et al. 1997; Raposo et al. 2007]. Even one Website may have many different Web page templates. To manually label all of these templates a lot of human input is required. So the supervised learning approaches are usually not applicable to Web-scale structured data extraction and are used to extract data from several selected Web sites of high interest.

The obvious advantage of automatic data extraction systems [Zhai and Liu 2006; Liu et al. 2010; Zhao et al. 2005; Crescenzi 2001; Álvarez et al. 2008; Liu 2005; Jindal and Bing 2010; Kayed and Chang 2010; Simon 2005; Hong et al. 2010; Su et al. 2011; Liu and Grossman 2003] is that they are able to extract structured data from different Web pages without human intervention. In this work we focus on the latter as we believe that only fully automatic structured data extraction systems can be applied for Web-scale data extraction (for a more extensive review of manual structured data extraction approaches please see surveys by [Chang et al. 2006; Laender, BA Ribeiro-Neto, et al. 2002; Lam and Gong 2005]).

One widely adopted technique to automatically detect and extract Data Records is to search for repetitive patterns in HTML source code by calculating the similarity of HTML tree nodes. Variations of simple tree matching algorithm [Yang 1991] are employed for this task. Bing Liu et al. was first to exploit simple tree matching algorithm for many automatic structured data extraction systems called DEPTA [Zhai and Liu 2006], NET [Liu 2005], and G-STM [Jindal and Bing 2010]. The latest state-of-the-art G-STM system integrates the grammar based approach and tree matching to produce a brand new algorithm, which is a more principled approach to extract structured Web data from a Web page [Jindal and Bing 2010]. G-STM generalizes a tree matching algorithm [Zhang and Shasha 1989] and introduces special grammar generation method so that it can identify and deal with lists inside Data Records. It worth mentioning, that ClustVX system can also handle lists inside Data Records. However these two systems work in a totally different way: G-STM is tree patterns comparison based, while ClustVX is clustering based. Furthermore, ClustVX relies on both rendered visual and pure HTML features of Web pages to extract Data Records, while G-STM only uses HTML tree.

Some other automatic structured data extraction systems called FiVaTech [Kayed and Chang 2010], CTVS [Su et al. 2011] (and its predecessor DeLa [Wang and Lochovsky 2003]), Viper [Simon 2005], WISH [Hong et al. 2010] work very similarly to G-STM and depend on the same tree matching technique [Yang 1991]. Due to space limitations we do not discuss in details these very similar systems.

M. Alvarez et al. propose another structured data extraction method [Álvarez et al. 2008] which is not based on simple tree matching like all above systems. Instead of comparing nodes in HTML tree directly, the authors first convert them to a string and then calculate string edit distance to determine the similarity. Their method constitutes three main steps. First, the method begins by finding the dominant Data Region in a Web page. Then, it performs a clustering process to limit the number of candidate record divisions in the dominant Data Region. Each candidate record list will propose a particular division of the Data Region into records. After that, their system chooses the one having highest similarity according to string edit distance calculations. The similarity between records in a Data Region is determined by comparing two sequences of consecutive sibling sub trees in the HTML tree of a page in each of the record. Finally, a multiple string alignment algorithm is used to extract the attribute values of each Data Record. This means that after selecting an initial Data Record as a starting string, each additionally selected Data Record is aligned to the so far obtained string, until all Data Records are aligned to the string. The same Data Items alignment technique is also used in DEPTA [Zhai and Liu 2006] system.

Contrary to the above systems, which search for repeated patterns purely in HTML code, VINTS [Zhao et al. 2005] system utilizes both visual content features and HTML tree structure regularities of Web page to detect structured data. This system also employs a clustering technique: to identify the main Data Region containing Data Records VINTS first identifies content line separators (visual feature) and uses them to segment result page into visual block. Then the blocks that are consecutive and visually similar are clustered into one group. The clustering is based on visual similarity, which means that two blocks are visually similar if their type distance, shape distance and position distance are all below certain hardcoded thresholds. After that, VINTS uses heuristics to determine which of blocks contain Data Records. Although visually aided clustering is used in VINTS it differs from the one employed in ClustVX. First of all, the clustering in VINTS is used to identify dominating Data Region, while we use it for structured data extraction. Furthermore, VINTS clusters visual blocks, while we cluster Data Items. And the most apparent difference is the extraction capabilities of these two systems: VINTS can extract data from only one Data Region in a page, while our proposed ClustVX system can handle many Data Regions per page.

A more recent automatic structured data extraction system VIDE [Liu et al. 2010] tries to not to depend on HTML tree at all and instead uses purely visual features of a Web page. Using patented VIPS [Cai et al. 2003] algorithm it builds a visual containment tree of a Web page and uses it instead of pure HTML tree. However if there are some unloaded images or missing style information in a Web page VIPS may fail to build correct visual containment tree which leads to data extraction problems [Liu et al. 2010].

Some authors [Furche et al. 2011; Walther 2012] have also addressed the problem of rendering modern Web pages in a browser and only then extracting data. To render a Web page they employ frameworks originally used for Web application testing (HTMLUnit, Selenium, FireWatir, and etc). Such frameworks control modern Web browsers, such as Mozilla Firefox, Chrome, or Internet Explorer. Similarly to the systems described in the corresponding publications the ClustVX system also utilizes a Web application testing framework to control the Mozilla Firefox browser.

All the reviewed systems extract structured data from Web pages where the data is structured [Cafarella et al. 2011] by HTML markup and visual styling, especially when Web pages are automatically generated with templates and populated from underlying databases. The structured nature of the targeted data sets *Web data extraction* apart from *information extraction* where entities and their relations are extracted from natural text [Furche et al. 2012]. Information extraction systems [Etzioni et al. 2011; Suchanek et al. 2007] usually employ various natural language processing techniques and search free text for facts. A fact is represented as a tuple consisting of entities (real world objects) and their relationship. For instance, given the sentence, "McCain fought hard against Obama, but finally lost the election," an information extraction system should extract two tuples, (*McCain, fought against, Obama*), and (*McCain, lost, the election*) [Etzioni et al. 2011]. This way millions of facts are extracted from Web-scale text corpus and are put into knowledge bases. However, these systems are not yet applicable to extract structured data with multiple attributes, where the structure is defined at a database level.

3 The Architecture of Proposed ClustVX System

The general architecture of proposed ClustVX system is presented in Fig. 2. Web browser is used to download HTML code from a Web server and fully render Web page in a browser window (in current implementation of ClustVX system we use Mozilla Firefox Web browser). Web browser downloads all required additional data, such as cascading style sheets (CSS), JavaScript code and etc. In the browser all retrieved JavaScript code is executed, CSS is used to visually style Web page elements. All this process visually renders Web page and prepares it for browsing. After Web page is rendered at browser level we artificially execute additional JavaScript code to retrieve visual features and only then pass resulting HTML code to ClustVX system. Please see Fig. 3 where detailed process flow diagram is presented.

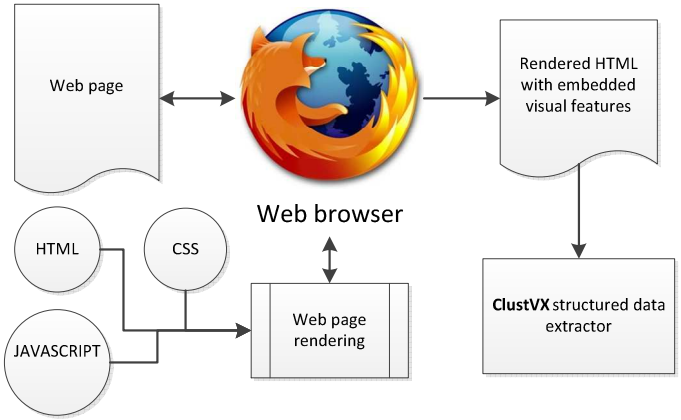


Figure 2: ClustVX system architecture

As we further see from Fig. 3, after rendering Web page we proceed to Web page pre-processing stage where a JavaScript code is injected into the Web page to select rendered visual features, such as font size, font style, font colour and etc. The extracted visual features of each HTML element node are then embedded into the element node as attributes. Unenclosed text nodes are put under artificially created elements. In this stage we also remove all HTML text formatting elements, such as , <bold>, <italic> and etc. The HTML tree with embedded visual data, enclosed text tags and removed text formatting element nodes is passed to *Xstrings* generation stage. Xstrings are modified XPath location strings with added visual data.

Clustering stage clusters Xstrings into separate clusters. Data Records extractor uses from clustering stage inherited information and identifies Data Regions available on the Web page. Each Data Region gets rules (XPaths) to extract Data Records, which are inside Data Region. Then Data Items extractor extracts and aligns all Data Items within each Data Record.

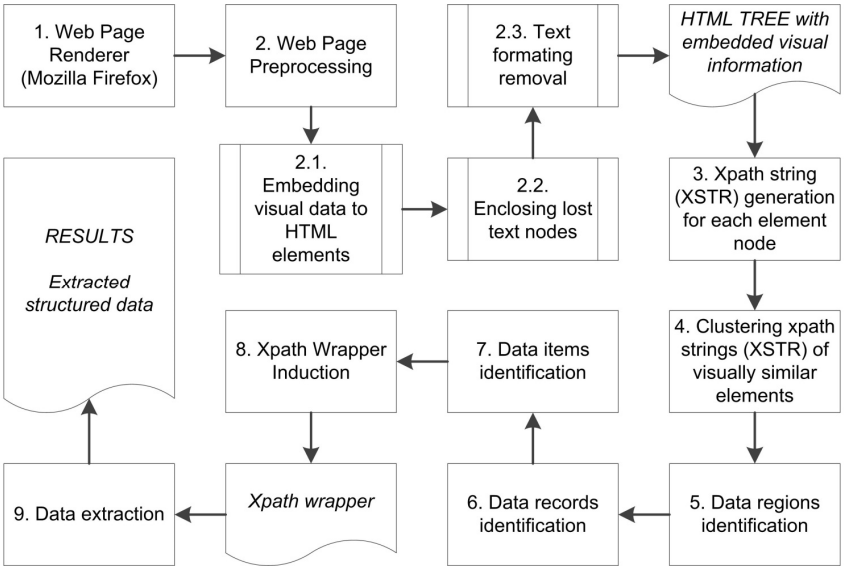


Figure 3: Process flow of structured data extraction with ClustVX

The results of the data extraction process are visualized in a HTML table. During the Data Records extraction and Data Items alignment phases XPath wrapper is automatically induced. XPath wrapper can be later reused to directly extract structured data from the Web page without a need to repeat all the stages. Detailed explanation of each extraction step is available in the following Sections of this paper.

4 Structured Data Extraction

The objective of structured Web data extraction is to extract all Data Records from a given Web page. We assume that the Web page is dynamically generated and hence an underlying template exists. Data Items are usually located in Data Records, which all together make a Data Region. According to [Liu et al. 2010], an ideal structured data extractor should achieve the following: 1) all Data Records in the Data Region are extracted and 2) for each extracted Data Record, no Data Item is missed and no incorrect Data Item is included.

In this Section we in detail explain the process of structured Web data extraction with ClustVX system. We begin by reviewing structural and visual features of Web pages that are used in structured data extraction process. The process itself consists of many separate steps. First of all HTML of a Web pages is pre-processed to enclose unenclosed text tokens and to embed visual information to each HTML element (see Section 4.2.). Then each visually visible text element of a Web page is clustered to clusters according to its visual similarity and XPath strings (see Section 4.3 and 4.4). By manipulating XPaths in clusters ClustVX system locates Data Regions, segments Data Records. A *visual weight* for each Data Region is then calculated to determine

its importance in a page. And only then Data Records are extracted (see Sections 4.5 – 4.7). The final steps of structured Web data extraction process in ClustVX system are wrapper induction and Data Items extraction (see Section 6.6).

4.1 Structural and Visual Features of Web Pages

It is becoming very difficult to access and extract structured data from Web 2.0 pages, where many parts of page content are generated dynamically with JavaScript code. This leads to a necessity to first of all render a Web page in a contemporary Web browser which executes all embedded JavaScript codes, applies cascading style sheets and etc. Only then we can leverage visual features of a Web page, such as font colour and font size for structured Web data extraction. In this Section the main visual and structural features of a Web page are presented. We exploit these features to extract structured Web data.

4.1.1 Structural Features

Each Web page can be presented as a tree structure which is a fundamental data structure in XML documents. See Fig. 4 for an example. Thus a very useful approach to extract structured data from Hypertext Markup Language (HTML) documents is to employ Extensible Markup Language (XML) technologies to translate HTML to valid XML code. In this approach, HTML documents are first normalized into Extensible HTML (XHMTL) and then then processed by XML applications [Myllymaki and Jackson 2002].

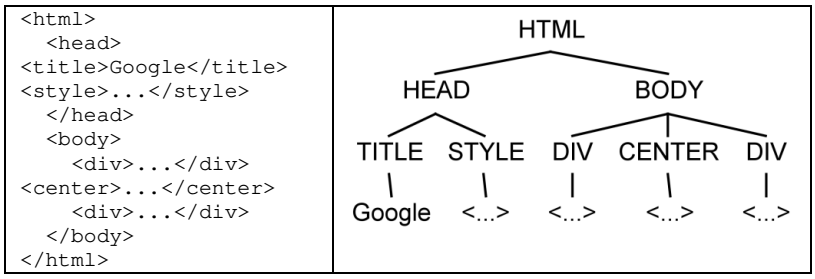


Figure 4: HTML source code on left shown as a tree structure on right

XPath is a language used to navigated XML tree structure. The primary purpose of XPath is to access parts of an XML document. XPath operates on the abstract, logical structure of an XML document, rather than its underlying syntax (source code). XPath gets its name from its use of a path notation as in URLs for navigating through the hierarchical structure of an XML document. XPath sees an XML document as a tree of nodes (see Fig. 4). The modelled HTML tree is widely used in structured data extraction algorithms [Chang 2001; Chang and Kuo 2004; Dalvi and Bohannon 2009; Myllymaki and Jackson 2002; Bohannon et al. 2012], where usually dynamic programming is employed to find similar branches of the tree. Similarity is determined by comparing every tree nodes with each another. Contrary to this comparison based approaches for searching similar trees, ClustVX employs clustering

of location paths (XPaths) which in turn are enhanced with visual features of Web page elements.

A location path (XPath) is one of most important kind of expressions in XPath language. An XPath locates and selects a set of nodes relative to the context node (usually a root node of the tree). The result of evaluating an XPath is the node-set containing the nodes selected by the XPath. Every XPath can be expressed using a straightforward but rather verbose syntax. There are also a number of syntactic abbreviations that allow common cases to be expressed concisely [Clark et al. 1999].

4.1.2 Visual Features

HTML code together with images and visual style information is rendered in a Web browser. The whole rendering process consists of computation of style data, frames construction, constant reflowing to represent changes or adding newly downloaded information. The result is a Web page as we see it in a Web browser window. It has been demonstrated that computed visual information can improve data extraction process [Liu et al. 2010]. In this Section we describe the two main visual features of a fully rendered Web page, which helps ClustVX to extract data: Web page layout and Web page element font features.

A text element in a Web page can be styled in many different ways. For example, it is possible to set individual size, colour, weight, strikethrough for each text string. The change of any of the font features, such as *size, face, colour, strikethrough, weight, italic, underline,* modifies the visual presentation style of the text element. Visual font features are often used to differentiate one semantic type of text from another. These features help ClustVX system to efficiently group semantically similar text elements to same clusters.

Each Web page element, such as let it be an image, a text string, or a white space, after Web page rendering is placed to some location. The whole Web page can be seen as a coordinate system where x axis describes horizontal position and y axis describes vertical position. The most left top corner of a Web page is a starting point, where x and y are equal to zero. The more an element is down in the page the bigger y value it has. And similarly, the more an element is to the right in the page the bigger x value it has. According to W3 specification [Kesteren 2011], each Web page element is bounded by a rectangular box which is called bounding rectangle. The left top corner of a bounding rectangle is positioned at an exact coordinate (x, y) in a Web page. So the left, top, right and bottom properties are describing the bounding rectangle, in pixels, with the top-left relative to the top-left of the page view. Combined view of all rectangles makes a Web page view. The ClustVX system exploits these visual attributes to determine visual position and size of each element.

4.2 Web Page Pre-Processing

After a Web page has been rendered in a Mozilla Firefox browser, but just before the extraction process is started, the Web page is modified to enhance Web data extraction and to speed up whole process. The modification includes: a) embedding visually significant data to the attributes of each Web page element; b) enhancing HTML tree structure by enclosing lost text nodes within new parent node; c) removing any text formatting. Those steps are described in the following Sections.

4.2.1 Embedding Visual Data to HTML Elements

The technique embedding visual data into HTML code is employed to enable processing Web page source code without constant API calls to a browser and, at the same time, retaining accessibility to all visually important information. It is done in this way: while a Web page in rendered in a browser window all important visual information is extracted using browsers API calls and JavaScript code. The extracted information is then embedded into each corresponding HTML element as an additional attribute. These attributes has no effects on visual appearance of a Web page. Fig. 5 shows source code of HTML element with embedded visual information to "*left*", "*top*", "*width*", "*height*", and "*fontdata*" tags. These visual features are later used in structured data extraction stage of ClustVX system. Visual appearance features, such as *font type, font size, font colour* and visual position features of rectangle boxes, such as *left, top, width, height*, are embedded into HTML code.

```
<span
left="631.5"
top="1477.5"
width="195"
height="17"
fontdata="Arial,Verdana,Helvetica,sans-serif;rgb(0,
0, 0);10px;normal;400;none"
class="artbox_1col_finance">
```

Figure 5: An example of visual data embedding into attributes

```
<p>
  <b>
    Some unenclosed text
    <br>
    E-Mail Address:
    <a
href="mailto:K6@hm.com">
    <i> K6@hm.com </i>
    </a>
    <br>
    Contact By: E-Mail
    <br>
    Ad Number: 101205
  </b>
</p>
```

```
<p>
  <b>
    <span>Some unenclosed text</span>
    <br>
    <span>E-Mail Address:</span>
    <a href="mailto: K6@hm.com ">
    <i> K6@hm.com </i>
    </a>
    <br>
    <span>Contact By: E-Mail</span>
    <br>
    <span>Ad Number: 101205</span>
  </b>
</p>
```

Figure 6: An example of enclosing lost text nodes

4.2.2 Enclosing Lost Text Nodes

Some Web pages have text strings (text nodes), which has no individual HTML element as a parent. Those nodes are often put under the same parent together with other HTML elements and other text strings. Because ClustVX system identifies the exact position of each HTML element by using XPath, each text string must have its own HTML element and no other sibling under the same XPath should exist. So in

Web page pre-processing stage we search for each unenclosed text strings in HTML tree and enclose them with tag. Fig. 6 demonstrates an example of text strings in HTML tree, which do not have individual enclosing tag and will be put under the element. The unenclosed text nodes are showed in bold on left. On right side of Fig. 6 we see the resulting HTML source code after the enclosure process. The process added additional enclosing tags shown in red colour.

4.2.3 Removing Text Formatting Element Nodes

HTML elements that are used to visually style text, such as <*em*> to emphasize, <*u*> to underline and etc. often do not help to identify structural nature of encoded text data. Contrary, we have observed that sometimes formatted text nodes can impede data extraction process. Often text formatting nodes divide semantically contiguous text into two or more text nodes in HTML tree. As seen in Fig. 7, formatting element node (<*b*> - bold) divides a description of a book into three text nodes. It is semantically contiguous text and should not be divided. This kind of division impedes the data extraction process, so we simple remove those formatting element nodes and form a new contiguous text node.

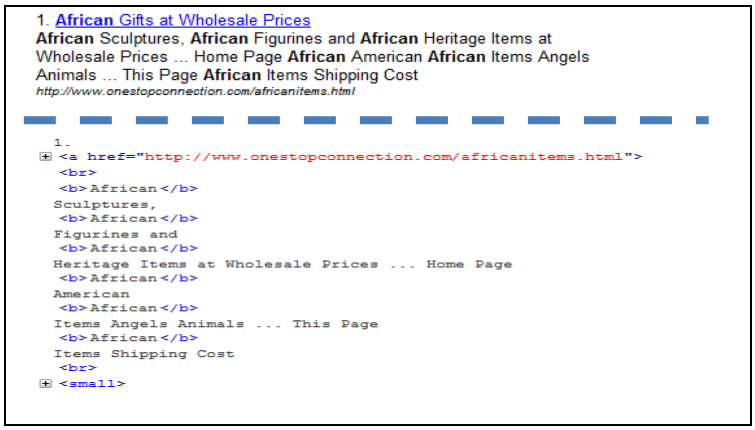


Figure 7: An example of text formatting tags dividing semantically contiguous text

4.3 Generating XPath Strings with Visual Data (Xstrings)

In this stage ClustVX algorithm iterates each HTML element. For each visible Web page element we generate an absolute location path (XPath). Furthermore, we extract all embedded visual data, which is stored in the attributes of that element: font size, font style, font colour, font weight, font strikethrough. For example, let's say we have obtained font data and XPath of the element. The next step is to remove from XPath the indexes, separation slashes and leave only tag names. Then we join tag names into a string together with font data. We call resulting string an *Xstring*. We will use the Xstrings in the next clustering stage of the ClustVX algorithm. See Table 1 for an example with Xstring.

***XPath**:*	/html[1]/body[1]/div[2]/**div[1]**/div[1]/div [5]/p[2]/a[1]
***Font Data**:*	Arial,Verdana,Helvetica,sans-serif;rgb(0, 0, 0);10px;normal;400;none
***XSTRING as a result**:*	*htmlbodydivdivdivdivpa-Arial,Verdana,Helvetica,sans-serif;rgb(0, 0, 0);10px;normal;400;none*

Table 1: An example of Xstring formation

4.4 Clustering Visually Similar Web Page Elements

To ease understanding of structured data presented in a Web page, Web page template designers often arrange the Data Records and the Data Items with visual regularity to meet the reading habits of human beings. Data Items of the same semantic in different Data Records are similar on layout and font [Liu et al. 2010]. By exploiting this observation the clustering stage of ClustVX Web data extraction system puts visually similar Web page elements into the same clusters.

The similarity of elements is computed by comparing tag paths and font data, i.e. the Xstrings. To cluster according to similarly, as in VIDE [Liu et al. 2010], a single one pass clustering algorithm is employed: we take any first HTML element from a HTML tree and look at its visual appearance data (Xstring). If there is no cluster representing such Xstring we simple create a new cluster and put the element there. If element has the same visual properties as elements from other cluster, we put that element there. In other words, if two elements have same Xstring string, they are put into the single cluster. The same thing is done with all remaining elements from HTML tree.

4.5 Data Records Identification and Extraction

Data Record extraction in ClustVX system is based on two observations about Data Records presentation in Web pages: 1) A group of Data Records are usually rendered in a contiguous region of a Web page [Zhai and Liu 2006] and are visually similar; 2) A group of Data Records are formed by some child sub trees and at some level have same parent node [Zhai and Liu 2006].

There exist two ways of embedding Data Records into HTML tree. For example consider that a single Data Record in a Web page consists of three node tags *<A>, <DIV>* and *<SPAN>*. *<A>* is the first node in a Data Record and <SPAN> is a last. Fig. 8 demonstrates two ways, how a list of mentioned Data Records can be presented in a Web page:

a) Each Data Record belongs to only one parent node *(<DIV>$_1$, <DIV>$_2$,* and *<DIV>$_3$*).
b) A set of nodes forms a Data Record and they all are placed under the same parent. Here only one *
* tag node separates the three Data Records.

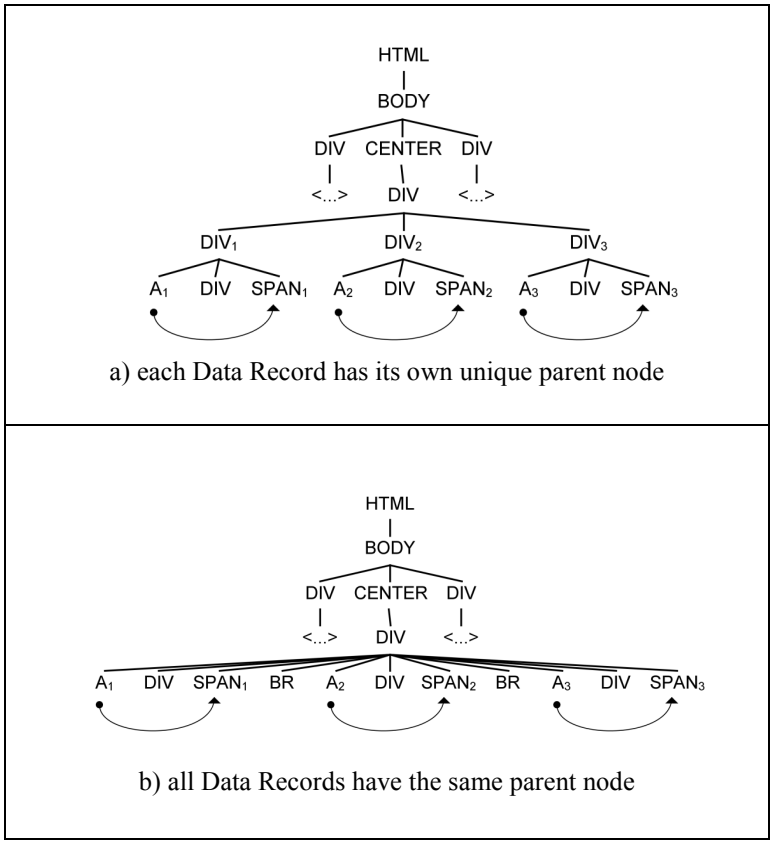


Figure 8: Two types of structural Data Records representation in HTML tree

While these two types of Data Records visually may be rendered the same, but as we see in Fig. 8, the parent and children relationships in the HTML tree are totally different. This difference has an profound implication for Data Records identification and separation: a) if there exist one parent for each Data Record, then we identify the boundary of Data Record just by looking at the boundary of the parent; b) if Data Record consists of a set of nodes, and all nodes of all Data Records in particular Data Region are under the same parent node, we first need to find the boundary of each Data Record and combine all nodes which belongs to that Data Record into the same group. The detailed explanation of the process is described in following Section.

4.5.1 Finding Data Regions, Data Records and Data Items

Each clustered HTML element has a unique XPath string, which points to the exact location of the element in the HTML tree. Using this information it is very easy to find Data Region, Data Records and Data Items in a Web page. Just by calculating longest common tag path prefix of all elements in particular cluster, we find the tag path of a Data Region.

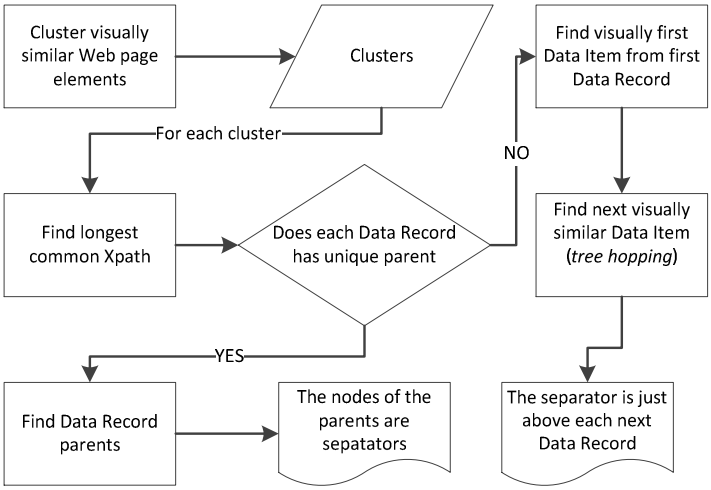


Figure 9: The diagram depicting Data Records segmentation process

Depending on the type of Data Record, the two heuristics (for a visualization see Fig. 9) are used to segment Data Records: 1) If Data Records are of a type A as seen in Fig. 8a, that is – each Data Record in a Data Region is under its own parent, then Data Records tag path will be the first tag path whose number differs in each clustered Data Item. Sometimes Data Record tag path can consist of two or more tag names. 2) If Data Records are of type B as seen in Fig. 8b, that is – each Data Record consists of a set of nodes and all Data Records and their sets of nodes are under the same parent, than different approach is used, which we call HTML tree hopping technique.

The tree hopping technique works in this way: ClustVX identifies the first (visually) Data Item in the first record of Data Region. In Fig. 10a the first Data Item is "1: Amarillo Globe-News: Headlines". Then all other Data Items, which are visually located between the identified visually first one and the second, forms a Data Record. In Fig. 10a the start of a second Data Record is "2: On The Edge Of Common Sense". So these three Data Items "1: Amarillo Globe-News: Headlines", "http://www.amarillonet.com/1/index.html", and "(Score 84, Size 18K, Last modified Nov-07-03)" belong to the Data Record 1. As we see, in the corresponding HTML source code in Fig. 10b, each Data Record is separated by *"

"* separator and all Data Records are under the same parent node.

Thus by manipulating and combining tag paths of the HTML elements in clusters we find tag paths of all Data Regions, Data Records and Data Items.

4.5.2 Finding Visual Weights of Data Regions

Since there can by many Data Regions in a Web page, it is handy to somehow rank these regions according to their importance. One straightforward way to do it is to calculate the visually occupying place of each Data Region (*visual weight*) in a rendered Web page. We presume, that the more visual place the Data Region occupies in a rendered Web page, the more important it is. Furthermore, the amount of Data

Records and Data Items in that region also contributes to its visual weight. During the visual weight calculation we square the number of Data items to significantly improve their importance.

Search found 25 documents that match your criteria:

1: Amarillo Globe-News: Headlines
http://www.amarillonet.com/1/index.html
(Score 84, Size 18K, Last modified Nov-07-03)

2: On The Edge Of Common Sense
http://www.amarillonet.com/2.index.html
(Score 82, Size 31K, Last modified Nov-07-03)

(a) rendered view in a browser

Data Record 1

```
<font size="2" face="arial,helvetica,sans-serif">
        <b>1: Amarillo Globe-News: Headlines</b>
  </font>
  <br>   
  <font size="1" face="arial,helvetica,sans-serif">
         <a
href="http://www.amarillonet.com/1/index.html">
                http://www.amarillonet.com/1/index.html
        </a>
  </font>
  <br><i>(Score 84, Size 18K, Last modified Nov-07-
03)</i>
```

Separator

```
  <br><br>
```

Data Record 2

```
  <font size="2" face="arial,helvetica,sans-serif">
        <b>2: On The Edge Of Common Sense</b>
  </font>
  <br>   
  <font size="1" face="arial,helvetica,sans-serif">
        <a href="http://www.amarillonet.com/2/index">
                http://www.amarillonet.com/2.index.html
        </a>
  </font>
  <br><i>(Score 82, Size 31K, Last modified Nov-07-
03)</i>
  <br><br>
```

(b) HTML source code

Figure 10: Data Records are under the same parent and are separated by <br> tag

Sometimes Data Regions contains only a few Data Records which are spread very widely on a page. Such Data Regions contain small amount of data and are not typically important for extraction. An example of widely spread Data Records is pagination numbers which are locate above and below paginated data. In that case, even though the total occupying visual area is significantly big, but it is very widely spread and contains a small amount of textual data. To decrease the visual weight of widely spread Data Regions and to increase the visual weight of regular Data Records with many Data Items we square the latter number. So in ClustVX system we use the following formula to calculate visual weight of particular Data Region:

*Visual Weight = (average area of one Data Record) * (number of Data Items)^2*

4.5.3 Extracting Data Records

Data Record extraction process is very straightforward: since we already know the tag paths of each Data Region in a Web page and the tag paths of Data Records, the only thing we need to do is to traverse the HTML tree and collect relevant information.

4.6 Data Items Identification and Extraction

As described in previous Section on this paper, by analysing tag paths in every cluster of visually similar Web page elements we derive tag paths of Data Regions, Data Records and Data Items.

To extract Data Items we first need to locate Data Region and to find each Data Record. Inside each Data Record we extract Data Items. All these steps are done using tag paths. This is very convenient way of extracting data from Web pages, since in HTML tree, which represents the structure of HTML code, each tag path leads to a particular location.

The tag paths of Data Items in Data Records are also used in alignment process. We presume that Data Items in each Data Records have the same tag paths. During the extraction process Data Items are aligned according to their tag paths in a Data Record.

4.7 Wrapper Induction

Wrapper induction step automatically derives extraction rules, which can be later reused to extract structured data from the same template Web page. Usually, these extraction rules are saved as XPath sets per Data Region. In case of type B Data Region (as seen in Fig. 8b), we also save the XPath of visually first Data Record item. Fig. 12 shows a sample wrapper for one Data Region. The first XPath is the address of Data Region, second XPath is the relative XPath of each Data Record and the last three XPaths are Data Items, which can be found in each Data Record.

```
"/html[1]/body[1]/form[1]/table[1]/tbody[1]/tr[2]/td[1]/div[1]"
: {
      "/div[*]" : [
         "/div[1]/div[2]/div[2]/span[2]",
         "/div[1]/div[4]/p[1]/span[1]/a[1]/span[2]",
         "/div[1]/div[2]/div[1]/span[1]",
]}
```

Figure 12: An example of XPath wrapper

5 Experimental Evaluation

This Section describes the extensive experimental evaluation of our proposed ClustVX system with three benchmark datasets containing in total 6264 Data Records. Given a Web page from a dataset we process it with data extraction system

and measure the extraction results. Here are three benchmark datasets that we use to evaluate structured Web data extraction systems:

1. ClustVX dataset[1]
2. ViNTs dataset 2 [Zhao et al. 2005]
3. M. Alvarez et al. dataset [Álvarez et al. 2008]

These datasets contain search results of the Deep Web, that is, pages generated using templates filled with data coming from databases. Usually these databases are accessed when user submits a Web form (for example, to find all books that in title contain word „data"), and Web server retrieves requested data and generates a Web page. See Table 2 for detailed characteristics of these datasets. The total number of Data Records is calculated by analysing only one page per Web site. In case there are many Web pages per site in a dataset, we take only the first one from alphabetic. If first page contains no Data Records, then the second page is taken.

Dataset:	**ClustVX**	**VINTS-2**	**M. Alvarez et al.**
Sites	10	102	200
Pages per site	3	11	1
Average records per page	22	24	18
Total records (1st page per site)	218	2489	3557

Table 2: Benchmark datasets containing Deep Web pages

The experimental results are compared with state-of-the-art automatic structured data extraction systems called G-STM [Jindal and Bing 2010], DEPTA [Zhai and Liu 2006], FIVATECH [Kayed and Chang 2010], MDR [Liu and Grossman 2003], ViNTs [Zhao et al. 2005], and the method proposed in [Álvarez et al. 2008]. Some of these systems are not publicly available to download. In such case we use the reported evaluation results from the original publications.

Since some of the Web pages have malformed HTML source codes, as it was done in RoadRunner [Crescenzi 2001] and DEPTA [Zhai and Liu 2006] experiments, we use the Tidy program [Paehl 2012] to clean the source code of malformed pages.

5.1 Evaluation Metrics

We use the precision, recall and f-score measures (which are widely used to evaluate information retrieval system) to evaluate the effectiveness of our system for extracting structured Web data. For Web data extraction, the recall and precision are computed based on the total number of correct Data Records found in all pages and the actual number of Data Records in these pages. See the following formulas for calculating precision, recall and f-score:

- **Precision** = (| {Correctly extracted Data Records} ∩ {Extracted Data Records} |) / (| {Extracted Data Records} |);
- **Recall** = (| {Extracted Data Records} ∩ {Actual Data Records} |) / (| {Actual Data Records} |);
- **F-score** = (2 * Precision * Recall) / (Precision + Recall)

[1] Available to download at http://clustvx.no-ip.org/dataset.zip

As in FIVATECH [Kayed and Chang 2010] experimentation, we consider Data Record to be successfully extracted if more that 60% of its Data Items are extracted correctly and each of Data Records in a Data Region is correctly segmented.

5.2 Experimental Results with Dataset 1

The first ClustVX dataset is compiled by us (see Table 3 for details). We chose ten technologically sophisticated modern Web sites, where much of content is dynamically modified with JavaScript code and style is set by Cascading Style Sheets (CSS). Since many automatic Web data extraction systems take as an input a raw HTML file they would fail to get the final HTML of a modern Web page where the execution of JavaScript code and live HTML modifications are done while rendering a Web page in a browser. In contrast to raw HTML handling the proposed ClustVX systems uses a modern Web browser to fully render a HTML file and take the final version of the source code. To conduct a proper comparison between those systems that do not render HTML files and the proposed ClustVX system we decided to: a) fully render in the browser each Web site from the dataset; b) save the resulting rendered version of the HTML source code; c) provide the rendered version of the HTML code as an input for automatic structured data extraction systems.

			ViNTs		**FiVaTech**		**MDR**		**ClustVX**	
	Website	**TRUE TOTAL**	*total*	*TP*	*total*	*TP*	*total*	*TP*	*total*	*TP*
1	currys.co.uk	20	19	19	20	20	0	0	20	20
2	google.com	10	0	0	7	7	0	0	10	10
3	argos.co.uk	50	50	50	3	3	1	0	50	50
4	bestbuy.com	15	0	0	15	15	0	0	15	15
5	clothingattesco.com	20	3	0	20	20	0	0	20	20
6	samsung.com	15	9	0	3	3	0	0	15	15
7	adidas.com	24	24	24	24	24	0	0	24	24
8	amazon.com	24	0	0	24	24	0	0	24	24
9	barnesandnoble.com	30	30	30	0	0	8	0	29	29
10	bing.com	10	8	0	13	10	11	10	10	10
	total:	**218**	**143**	**123**	**129**	**126**	**20**	**10**	**217**	**217**

Table 3: Experimental results on ClustVX dataset in absolute numbers

We compare the effectiveness of our proposed ClustVX system to three publically available to download or access automatic Web data extraction systems called MDR [Liu and Grossman 2003], ViNTs [Zhao et al. 2005], and FIVATECH [Kayed and Chang 2010]. The results in terms of absolute numbers are available in Table 3. We list the total number of Data Records available to extract in each Web page. The total numbers of actually extracted Data records by each system are marked as "total". Next to it we also list the True Positive (TP) extraction results, i.e. the number of correctly extracted Data Records by each system. The Table 4 lists the results on the same dataset presented in precision, recall and f-score.

System / Measure	ViNTs	FiVaTech	MDR	ClustVX
Precision	86.0%	97.7%	50.0%	100.0%
Recall	65.6%	59.2%	9.2%	99.5%
F-score	74.4%	73.7%	15.5%	99.8%

Table 4: Experimental results on ClustVX dataset in precision, recall and f-score

The results on this dataset reveal that out proposed ClustVX system achieves perfect precision and nearly perfect recall, 100.0% and 99.5% respectively. These results are better than those achieved with other methods. The lowest results in terms of f-score are achieved by MDR systems. We believe it is because the MDR system is the oldest systems of the four tested. During the creation of MDR the Web pages were technologically simpler and the system seems to be unable to cope effectively with modern Web pages.

5.3 Experimental Results with Dataset 2

The structured Data Records extraction results on VINTS-2 dataset reveal, as we see in Table 6, that ClustVX system again achieves very high precision and recall, 98.6 % and 98.5% respectively. These results are better than those reported with G-STM [Jindal and Bing 2010] and DEPTA [Zhai and Liu 2006] systems. The main difficulties which hindered data extraction for ClustVX system are related to malformed HTML source code, which cannot be fixed even with Tidy program. Another difficulty was structurally complex lists of Data Records. Errors mainly occur when ClustVX system cannot correctly segment Data Records. Incorrect segmentation leads to extraction problems, such as when not all Data Items per Data Record are extracted, or Data Items belonging to the same Data Record are assigned into many different Data Records.

System / Measure	ClustVX	G-STM	DEPTA
Reported actual records	2489	N/A	N/A
Extracted records	2452	N/A	N/A
Correctly extracted records	2417	N/A	N/A
Precision	98.6%	98.5%	95.1%
Recall	98.5%	96.7%	83.9%
F-score	98.5%	97.6%	89.1%

Table 6: Experimental Results on VINTS-2 dataset

5.4 Experimental Results with Dataset 3

Here, in Table 5, are the results with [Álvarez et al. 2008] dataset. This dataset in terms of Data Records number and different style Web sites is the biggest of all three. The extraction results of ClustVX system are compared to the reported results of the system proposed by Alvarez et al. in [Álvarez et al. 2008]. Same problems as we encountered with VINTS-2 dataset are present here also. Particularly, the incorrect segmentation of structured Data Records leads to extraction errors. However, even with these problems our ClustVX system achieves better results than the method proposed by M. Alvarez et al. We achieve 98.2% precision and 99.7% recall, while [Álvarez et al. 2008] achieve 97.9% and 98.3% respectively.

System:	ClustVX	Alvarez et al.
Reported actual records	3557	3557
Extracted records	3546	3570
Correctly extracted records	3482	3496
Precision	98.2%	97.9%
Recall	99.7%	98.3%
F-Score	98.9%	98.1%

Table 5: Experimental Results on [Álvarez et al. 2008] et al. dataset

6 Demo

For demonstration purposes the ClustVX system has been implemented using PERL scripting language. The online demo is available at http://clustvx.no-ip.org.

7 Conclusions

In this paper we introduced a novel method for extracting structured Data Records from template-generated web pages. The method, called ClustVX, is based on clustering visually similar web page elements. It first renders given web page in a contemporary web browser, then clusters visually and structurally similar repeating web page elements to identify the underlying structure of embedded structured Data Records. ClustVX can extract structured data from web pages where more than one Data Record is present.

Visual and structural features of technologically sophisticated modern web pages can be accessed by loading web pages into a modern web browser and then using browser's API to retrieve the features. During a web page loading process a web browser also executes all JavaScript code, runs any necessary asynchronous requests (AJAX), retrieves additional data, such as style sheet files (CSS), and finally presents a rendered version of the web page. The final rendered version of a web page can then be used in the structured data extraction process.

The XPath language is suitable for generating data extracting wrappers. A set of XPath expressions, executed in a right order, can successfully extract structured data records from template-generated web pages. The generated wrappers can be later

reused by many data extracting methods, since the XPath language is an open standard and is widely used. Thus the proposed method can be easily integrated into other systems.

Extensive experimental evaluation, including more than six thousands Data Records, revealed that the proposed method consistently outperforms other state-of-the-art techniques. The main problems which occurred while extracting structured Web data with our proposed ClustVX technique are related to malformed HTML source code, extreme similarity (visual and structural) between Data Records and non-records, and incorrect segmentation of Data Regions.

Acknowledgements

We would like to thank Dr. Juozas Gordevičius and Dr. Lukas Radvilavičius for giving us insightful suggestions and comments.

References

[Álvarez et al. 2008] ÁLVAREZ, M., PAN, A., RAPOSO, J., BELLAS, F. AND CACHEDA, F.: "Extracting lists of Data Records from semi-structured Web pages"; Data & Knowledge Engineering, 64, 2, (2008), 491–509.

[Baumgartner and Flesca 2001] BAUMGARTNER, R. AND FLESCA, S.: "Visual Web information extraction with Lixto". International Conference on Very Large Data Bases (VLDB). (2001).

[Britain et al. 1998] BRITAIN, G., HSU, C., DUNGS, M., SCIENCE, I. AND SCIENCE, C.: "Generating finite state transducers for semi-structured data extraction from the web"; Information Systems, (1998), 521–538.

[Cafarella and Halevy 2009] CAFARELLA, M.J. AND HALEVY, A.: "Data Integration for the Relational Web". In International Conference on Very Large Data Bases (VLDB), (2009).

[Cafarella et al. 2008] CAFARELLA, M.J., HALEVY, A., WANG, Z.D. AND WU, E.: "WebTables : Exploring the Power of Tables on the Web". In International Conference on Very Large Data Bases (VLDB). (2008).

[Cafarella et al. 2011] CAFARELLA, M.J. et al. “Structured data on the web”; Communications of the ACM, (2011), 54, 2, 72–79.

[Cai et al. 2003] CAI, D., YU, S. AND WEN, JI-RONG: "VIPS : a Vision-based Page Segmentation Algorithm"; Technical Report, Microsoft, MSR-TR-200, (2003).

[Chang et al. 2006] CHANG, CH, KAYED, M AND GIRGIS, R.: "A survey of Web information extraction systems"; IEEE Transactions on Knowledge and Data Engineering, 18, 10, (2006), 1411–1428.

[Chang 2001] CHANG, CHIA-HUI: "IEPAD : Information Extraction Based on Pattern Discovery". In The World Wide Web Conference. (2001), 681–688.

[Chang and Kuo 2004] CHANG, CHIA-HUI AND KUO, S.-C.: "OLERA : Semisupervised Web-Data Extraction"; IEEE Intelligent Systems, (2004).

[Clark et al. 1999] CLARK, J., DEROSE, S. AND CORP, I.: "XML Path Language (XPath)"; http://www.w3.org/TR/XPath/, November 1999, (1999).

[Crescenzi 2001] CRESCENZI, V.: "RoadRunner: Towards Automatic Data Extraction from Large Web Sites"; International Conference on Very Large Data Bases (VLDB), (2001).

[Dalvi and Bohannon 2009] DALVI, NILESH AND BOHANNON, P.: "Robust Web extraction: an approach based on a probabilistic tree-edit model"; ACM SIGMOD international conference on Management of data, (2009).

[Elmeleegy et al. 2011] ELMELEEGY, H., MADHAVAN, J. AND HALEVY, A.: "Harvesting relational tables from lists on the Web"; International Conference on Very Large Data Bases (VLDB), 20, 2, (2009), 209–226.

[Etzioni et al. 2011] ETZIONI, O. et al. „Open Information Extraction: The Second Generation“; International Joint Conference on Artificial Intelligence, (2011), 3–10.

[Furche et al. 2011] FURCHE, Tim, et al. “Oxpath: A language for scalable, memory-efficient data extraction from web applications”; Proc. of the VLDB Endowment, (2011) 1016-1027.

[Furche et al. 2012] FURCHE, T. et al. “AMBER: Automatic Supervision for Multi-Attribute Extraction”; arXiv preprint arXiv:1210, (2012).

[Gulhane et al. 2011] GULHANE, P., MADAAN, A. AND MEHTA, R.: "Web-scale information extraction with vertex". In IEEE International Conference on Data Engineering (ICDE). (2011).

[Hong et al. 2010] HONG, J.L., SIEW, E.-G. AND EGERTON, S.: "Information extraction for search engines using fast heuristic techniques"; Data & Knowledge Engineering, 69, 2, (2010), 169–196.

[Yang 1991] YANG, W.U.U.: "Identifying Syntactic Differences Between Two Programs"; Software - Practise and Experience, 21, JULY, (1991), 739–755.

[Jindal and Bing 2010] JINDAL, N. AND BING, L.: "A Generalized Tree Matching Algorithm Considering Nested Lists for Web Data Extraction"; The SIAM International Conference on Data Mining, (2010), 930–941.

[Kayed and Chang 2010] KAYED, MOHAMMED AND CHANG, CHIA-HUI: "FiVaTech : Page-Level Web Data Extraction from Template Pages"; IEEE Transactions on Knowledge and Data Engineering, 22, 2, (2010), 249–263.

[Kesteren 2011] KESTEREN, A. VAN: "CSSOM View Module"; http://www.w3.org/TR/cssom-view, August 2011, (2011).

[Kushmerick et al. 1997] KUSHMERICK, N., WELD, D. AND DOORENBOS, R.: "Wrapper induction for information extraction". In International Joint Conference on Artificial Intelligence (IJCAI). (1997).

[Laender, B Ribeiro-Neto, et al. 2002] LAENDER, A. AND RIBEIRO-NETO, B: "DEByE - Date extraction by example"; Data & Knowledge Engineering, 40, 2, (2002).

[Laender, BA Ribeiro-Neto, et al. 2002] LAENDER, A., RIBEIRO-NETO, BA AND SILVA, A. DA: "A brief survey of Web data extraction tools"; ACM SIGMOD international conference on Management of data, (2002).

[Lam and Gong 2005] LAM, M.I. AND GONG, Z.: "Web information extraction". In IEEE International Conference on Information Acquisition. (2005).

[Liu 2005] LIU, B.: "NET − A System for Extracting Web Data from Flat and Nested Data Records"; International Conference on Web Information System Engineering, (2005), 487–495.

[Liu and Grossman 2003] LIU, B. AND GROSSMAN, R.: "Mining Data Records in Web pages"; ACM SIGKDD international conference on Knowledge disc. and data mining, (2003).

[Liu et al. 2010] LIU, W. et. al. "ViDE : A Vision-Based Approach for Deep Web Data Extraction"; IEEE Transactions on Knowledge and Data Engineering, 22, 3, (2010), 447–460.

[Madhavan and Halevy 2009] MADHAVAN, J. AND HALEVY, A.: "Harnessing the Deep Web : Present and Future". In Biennial Conf. on Innovative Data Sys. Res. (CIDR). (2009).

[Madhavan et al. 2007] MADHAVAN, J., JEFFERY, S.R., COHEN, S., ET AL.: "Web-scale Data Integration : You can only afford to Pay As You Go". In Biennial Conference on Innovative Data Systems Research (CIDR). (2007), 342–350.

[Miao et al. 2009] MIAO, G., TATEMURA, J. AND HSIUNG, W.: "Extracting Data Records from the Web using tag path clustering"; The World Wide Web Conference, (2009), 981–990.

[Myllymaki and Jackson 2002] MYLLYMAKI, J. AND JACKSON, J.: "IBM Research Report Robust Web Data Extraction with XML Path Expressions"; Technical Report, IBM, (2002).

[Nie and Wen 2007] NIE, Z. AND WEN, JR: "Object-level vertical search". In Biennial Conference on Innovative Data Systems Research (CIDR). (2007).

[Paehl 2012] PAEHL, D.: "HTML Tidy Library Project Table of Contents"; http://tidy.sourceforge.net/, (2012).

[Raposo et al. 2007] RAPOSO, J., PAN, A., ÁLVAREZ, M. AND HIDALGO, J.: "Automatically maintaining wrappers for semi-structured Web sources"; Data & Knowledge Engineering, 61, 2, (2007), 331–358.

[Simon 2005] SIMON, K.: "ViPER: augmenting automatic information extraction with visual perceptions"; ACM int. conf. on Information and knowledge management (CIKM), (2005).

[Su et al. 2011] SU, W., WANG, J., LOCHOVSKY, F.H. AND LIU, Y.: "Combining Tag and Value Similarity for Data Extraction and Alignment"; IEEE Transactions on Knowledge and Data Engineering, 99, (2011), 1–15.

[Suchanek et al. 2007] SUCHANEK, F. et al. „Yago: a core of semantic knowledge“; Proceedings of WWW, (2007), 697–706.

[Walther 2012] WALTHER, M. “Unsupervised extraction of product information from semi-structured sources”; IEEE 13th International Symposium on Computational Intelligence and Informatics, (2012), 257-262.

[Wang and Lochovsky 2003] WANG, J. AND LOCHOVSKY, F.H.: "Data Extraction and Label Assignment for Web Databases"; The World Wide Web Conference, (2003), 187–196.

[Zhai 2005] ZHAI, Y.: "Web data extraction based on partial tree alignment"; The World Wide Web Conference, (2005), 76–85.

[Zhai and Liu 2006] ZHAI, Y. AND LIU, B.: "Structured Data Extraction from the Web Based on Partial Tree Alignment"; IEEE Transactions on Knowledge and Data Engineering, 18, 12, (2006), 1614–1628.

[Zhao et al. 2005] ZHAO, H., MENG, W., WU, Z. AND RAGHAVAN, V.: "Fully automatic wrapper generation for search engines"; The World Wide Web Conference, (2005).

[Zhang and Shasha 1989] ZHANG, K. SHASHA, D.: ”Simple Fast Algorithms for the Editing Distance between Trees and Related Problems”; SIAM Journal on Computing, (1989), 18, 6, 1245–1262.