Towards a Spatial Instance Learning Method for Deep Web Pages

Ermelinda Oro and Massimo Ruffolo

Institute of High Performance Computing and Networking of the Italian CNR
Via. P. Bucci, 41/C, 87036, Rende CS, Italy
{oro,ruffolo}@icar.cnr.it

Abstract. A large part of information available on the Web is hidden to conventional research engines because Web pages containing such information are dynamically generated as answers to query submitted by search form filled in by keywords. Such pages are referred as Deep Web pages and contain huge amount of relevant information for different application domain. For these reasons there is a constant high interest in efficiently extracting data from Deep Web data sources. In this paper we present a spatial instance learning method from Deep Web pages that exploits both the spatial arrangement and the visual features of data records and data items/fields produced by layout engines of web browsers. The proposed method is independent from the Deep Web pages encoding and from the presentation layout of data records. Furthermore, it allows for recognizing data records in Deep Web pages having multiple data regions. In the paper the effectiveness of the proposed method is proven by experiments carried out on a dataset of 100 Web pages randomly selected from most known Deep Web sites. Results obtained by using the proposed method show that the method has a very high precision and recall and that system works much better than MDR and ViNTS approaches applied to the same dataset.

Keywords: Web Information Extraction, Deep Web, Instance Learning, Web Wrapping.

1 Introduction

The Deep Web is the part of the Internet that is not accessible by conventional search engines. Deep Web pages are dynamically generated from databases in response to queries submitted via search forms filled in by keywords. The Deep Web continue to grow as organizations and companies make available their large amounts of data by providing Web-access facilities to their databases. Consequently, there is a constant high interest in efficiently extracting data from Deep Web data sources.

A large body of work on approaches for extracting data from Deep Web sources is already available in literature. Existing approaches, for the scopes of this paper, can be classified in two main groups: (i) approaches that mainly

P. Perner (Ed.): ICDM 2011, LNAI 6870, pp. 270–285, 2011.

use the internal representation of Deep Web pages [6,10,7,16,17,13], and (ii) approaches that exploit the visual appearance of Deep Web pages [18,8]. Approaches in both groups are still limited in many aspects. In particular, approaches based on the internal structure of Deep Web pages suffer of the complexity of today Web pages encodings. In fact, they need to be updated for facing the adoption of new standards and tags. In particular, the growing adoption of scripts and CSS style sheets, for presenting data to human users, makes Web pagers more complex than ever. Approaches that exploit the visual appearance of Web pages do not use the spatial arrangement and the visual features of data records and data items produced by layout engines of Web browsers completely and directly. So they exploit partially visual cues created by Web designers in order to help human users to make sense of Web pages contents.

In this paper we present a novel spatial instance learning method for Deep Web pages that exploits both the spatial arrangement and the visual features of data records and data items/fields produced by layout engines of web browsers. The proposed approach is founded on:

- The Positional Document Object Model (PDOM) that represents the spatial arrangement and the visual features of data records and data items produced by layout engines of Web browser for presenting Deep Web pages.
- A spatial similarity measure that computes visual similarity between PDOM nodes by using spatial model called rectangular cardinal relation [11]. Such a similarity measure takes into account visual cues, available after document rendering, that help human readers to make sense of page contents independently from the internal structure of Web pages;
- The definition of a very efficient and effective instance learning algorithm, based on a hierarchical clustering technique and heuristic aggregation methods, that allows for recognizing data records and data items in Deep Web pages independently from their visual arrangement.

Main contribution of this paper are:

- The definition of a data model well suited for representing the spatial structure and the visual features of layouted Deep Web pages.
- The definition of an instance learning algorithm ables to identify data records and items spread on multiple (data) regions of a single page. It is noteworthy that the algorithm allows for recognizing data records and items having any spatial arrangement (e.g. data records arranged either as lists or matrices where data items are indifferently organized in vertical or horizontal way).

The paper is organized as follows. Section 2 describes the related work. Section 3 presents the positional document object model used for representing spatial layout and presentation features of Deep Web Pages. Section 4 introduces a novel visual similarity measure based on rectangular cardinal direction spatial model that takes into account both spatial and visual features of Deep Web Pages. Section 5 presents and discusses the instance learning algorithm. Section 6 describes experimental results. Finally, Section 6 concludes the paper.

2 Related Work

Several approaches have been proposed in the literature for extracting data records from Deep Web pages. For the scopes of this paper, existing approaches can be classified in two main groups: (i) approaches that mainly use the internal representation of Deep Web pages, and (ii) approaches that exploit the visual appearance of Deep Web pages. HTML-based approaches can be further classified in manual and automatic.

In manual approaches, like W4F [14], the programmer finds patterns, expressed for example by XPath, from the page and then writes a program/wrapper that allows for identifying and extracting all the data records along with their data items/fields. Manual approaches are not scalable and not usable in the current Web because of the very large number of different arrangement of data records in available Deep Web pages.

Automatic approaches exploit three main types of algorithms, wrapper induction, instance learning and automatic extraction. In wrapper induction approaches, like SoftMealy [6], Stalker [10], etc. extraction rules are learnt, by using supervised machine learning algorithms, from a set of manually labeled pages. Learned rules are used for extracting data records from similar pages. Such kind of approaches still require a significative manual effort for selecting and labeling Web pages in the training set. The method proposed in this paper is completely automatic and no manual effort is required to the user. Instance learning approaches exploit regularities available in Deep Web pages in terms of DOM structures for detecting data records and their data items. In this family of approaches fall methods like MDR [7], DEPTA [16,17], STAVIES [13]. These approaches exploit unsupervised machine learning algorithms based on tree alignment techniques, hierarchical clustering, etc. Approaches falling in this category are strongly dependent from the HTML structure of Deep Web Pages. Our approach is completely independent form the HTML structure of Web pages because it uses a spatial representation of Web pages obtained from page presentations produced by layout engines of Web browsers. In automatic extraction approaches, like Roadrunner [4], patterns or grammars are learnt from set of pages containing similar data records. In this kind of approaches pages to use for learning wrappers have to be found manually or by another system then a set of heuristic rules based on highest-count tags, repeating-tags or ontology matching, etc. is used for identifying record boundaries. Furthermore many approaches falling in this category need two or more Web pages for learning the wrapper, while our method works on each single Deep Web page.

By analyzing a huge number of Deep Web pages we have observed that: (i) HTML is continuously evolving. When new versions of HTML or new tags are introduced, approaches based on previous versions have to be updated. (ii) Web designers use presentation features and spatial arrangement of data items for helping human user to identify data records. They do not take into account the complexity of underlying HTML encoding. So, (iii) the complexity of the source code of Web pages is ever-increasing. In fact, the final appearance of a Deep Web page depends from a complex combination of HTML, XML (XHTML), scripts

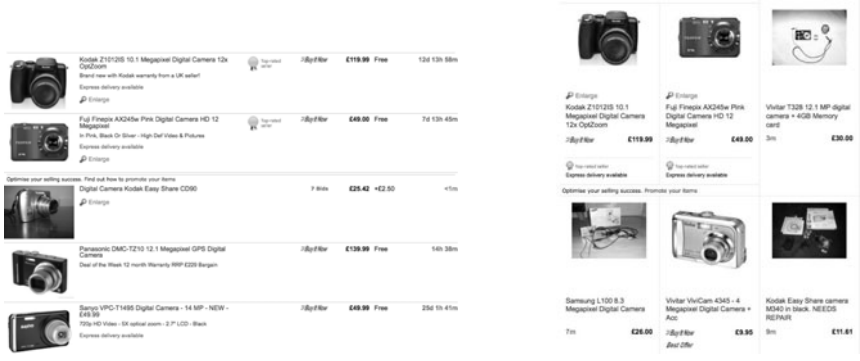

Fig. 1. Examples of layout models of data records on deep web pages

(javaScript), XSLT, and CSS. (iv) Data records are laid out either as lists or matrices where data items are indifferently organized in vertical or horizontal way (for instance, some layout models are shown in Figure 1). (v) Data records can be contained in not contiguous portions of a Web page (multiple data regions). These aspects make very difficult for existing approaches to learn instances and wrappers by using the internal encoding of Web pages so they constitute a source of strong limitations for approaches already proposed in literature [12].

Visual-based approaches, like LixTo [1,5,2], ViNTS [18], ViPERS [15], and ViDE [8], exploit some visual features of the Deep Web pages for defining and learning wrappers. In LixTo the programmer is helped in manually designing the wrapper by using the visual appearance of the Deep Web pages. In this case the programmer doesn't have to write code, s/he can design the wrapper by using only mouse click on the target Deep Web page. The user visually selects data items and records, then the system computes HTML patterns associated to visual area selected by the user and writes a wrapper that allow for applying such patterns in similar pages. LixTo is essentially a manual approach based on the HTML encoding of Web Pages. ViNTS uses visual features in order to learn wrappers that extract answers to queries on search engines. The approach detects visual regularities, i.e. content lines, in search engine answers, and then uses HTML tag structure to combine content lines into records. ViPER incorporates visual information on a web page for identifying and extracting data records by using a global multiple sequence alignment technique. Both last two approaches are strongly dependent from the HTML structure of Web page, whereas visual information play a small role, so they suffer from previously described limitations. Furthermore ViPER is able to identify and extract only the main data region missing records contained in multiple data regions. ViDE is the most recent visual-based approach. It make use of the page segmentation algorithm ViPS. Such algorithm takes in input a web page and returns a Visual Block tree, i.e. a hierarchical visual segmentation of a web page in which children blocks are spatially contained in ancestor blocks. The algorithm exploits some heuristics in order to identify similar groups of blocks that constitutes data records in which

constituent blocks represent data items. The ViDE approach suffer from some limitations. First it strongly depends from the page segmentation algorithm ViPS that in turn depends from the HTML encodings of Web Pages and from the set of assumptions made for segmenting Web pages. The ViPS algorithm, in fact, tries to compute a spatial representation in terms of Visual Block of a Web page by considering the DOM structure and visual information of a Web page produced by the layout engine of a Web browser. In particular, page segmentation algorithm strongly exploits the concept of separator. Separators are identified, in ViPS, by heuristic rules that make use of weights experimentally set. Moreover, the ViPS algorithm and then the ViDE approach suffer when data records are spread in multiple data regions each contained in different page segments, and also when data records are arranged as a matrix. The approach proposed in this paper only construct a spatial and visual model (PDOM) of Deep Web pages by considering presentation information returned by layout engined of Web browsers. To construct the PDOM our approach explores the DOM and acquires positions assigned by the layout engine to each node on the visualized Web page, and presentation features assigned to nodes. Data region, records and items recognition is then performed on the PDOM by using an heuristic algorithm that allows for discovering data records spread on multiple data regions, and data records having all possible spatial arrangement.

3 Positional Document Object Model

In this section we introduce the notion of *Positional Document Object Model* (PDOM) of Web pages. Then we describe how PDOMs are created starting from the traditional DOM by considering Web pages rendered by Web browsers. Usually, a Web page designer would organize the content of a Web page to make it easy for reading. However, the logical structure is encoded in a very intricate hierarchical HTML structure, in fact tag-nesting is used for representing presentation features, other than layout of Web Pages. As described in Section 2, some existing approaches first use heuristic document segmentation algorithms, e.g. work presented in [3,9] that use visual and/or content information (such as: separators, lines, blank areas, images, font sizes, colors, etc.), in order to point out the Web content structure, and then they try to recognize data records. So, the success of such approaches depends from the segmentation algorithms. Whereas, we adopt the PDOM that is based on the intrinsic segmentation hidden in the HTML structure and produced by Web browser.

3.1 Preliminary Definitions

A Web page can be seen as a 2-dimensional Cartesian plane on which are placed 2-dimensional objects (e.g. data records and items) surrounded by *Minimum Bounding Rectangles* (MBRs). MBRs are the most common approximations in spatial applications of 2-dimensional objects because they need only two points for their representation in the Cartesian space. The concept of MBr is defined as follows.

Definition 1 (Minimum Bounding Rectangle). *Let o be a 2-dimensional object, the minimum bounding rectangle (MBR) of o is the minimum rectangle r that surrounds o and has sides parallel to the axes (x and y) of the Cartesian plane. We call r_x and r_y the segments that are obtained as the projection of r on the x-axis and the y-axis respectively. Then, each side of the rectangle is represented by the segments (r_x^-, r_x^+) and (r_y^-, r_y^+), where r_x^- (resp. r_y^-) denote the* infimum *on the x-axis (y-axis) and r_x^+ (resp. r_y^+) denote the* supremum *on the x-axis (y-axis) of the segments r_x and r_y.*

Considering MBRs, directional and containment relations among 2-dimensional objects can be simply modeled. For representing directional relations we adopt the *Rectangular Cardinal Relation* (RCR) spatial reasoning model [11]. RCRs are computed by analyzing the 9 regions (cardinal tiles) formed, as shown in Fig. 2, by the projections of the sides of the reference MBR (i.e. r). The *atomic* RCRs are: *belongs to* (B), *South* (S), *SouthWest* (SW), *West* (W), *NorthWest* (NW), *North* (N), *NorthEast* (NE), *East* (E), and *SouthEast* (SE). Using the symbol ":" it is possible to express *conjunction* of atomic RCRs. For instance, by considering Fig. 2, r `E:NE` r_1 means that the rectangle r_1 lies on east and (symbol ":") north-east of the rectangle r. Moreover, the RCR model allows for expressing uncertain (disjunction of) directional relations: for example r `E|E:NE` r_1 means that r_1 lies on `E` or (symbol "|") on `E:NE` of r.

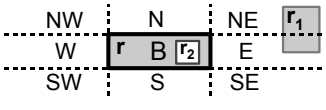

Fig. 2. Cardinal tiles

Definition 2 (Containment Relation). *Let $r = \langle (r_x^-, r_y^-), (r_x^+, r_y^+) \rangle$ and $\langle (r_x'^-, r_y'^-), (r_x'^+, r_y'^+) \rangle$ be two MBRs, we can say that r contains r' iff $r_x^- \leqslant r_x'^- \leqslant r_x'^+ \leqslant r_x^+$, and $r_y^- \leqslant r_y'^- \leqslant r_y'^+ \leqslant r_y^+$, and at least one inequality is strict. It is noteworthy that if no inequality are strict, then the MBRs are equivalent (i.e. $r = r'$).*

Given a set of no-intersecting 2-dimensional objects, it can be spatially ordered from left to right and from top to bottom considering their MBRs. In order to compute a spatial order among MBRs, we define the relations `above` and `before` and the concept of horizontal aligned MBRs.

Definition 3 (Above and Before relations). *Let a, b be two MBRs, we have a above b (a before b) iff $a_y^- < b_y^-$ ($a_x^- < b_x^-$ respectively).*

Definition 4 (Horizontal aligned MBRs). *Let a, b be two MBRs, they are on the same line iff $a_y^- \leqslant b_y^- \leqslant a_y^+$ or $b_y^- \leqslant a_y^- \leqslant b_y^+$.*

The Algorithm 1 allows for sorting no-intersecting MBRs on the base of the order in which they appear on the Cartesian plane (i.e. from left to right and from top to bottom).

Algorithm 1. Sort
Input: A set of MBRs $R = (r_1, \ldots, r_m)$;
Output: The ordered set of MBRs R'.
1.1: **for all** $(r_i, r_j \in R \wedge 1 \leqslant i < j \leqslant |R|)$ **do**
1.2: **if** $(r_i, r_j \notin$ *same line*$)$ **then**
1.3: **if**$(r_j$ *above* $r_i)$ **then** `swap`(r_i, r_j);
1.4: **else**
1.5: **if** $(r_j$ *before* $r_i)$ **then** `swap`(r_i, r_j);
1.6: **end if**
1.7: **end for**
1.8: **return** R;

Finally, we define the function `closest` that takes as input an MBR r and a set of MBRs R and returns the closest MBR $r_1 \in R$ to r. Closeness is computed by considering the distance between the center of the MBRs.

3.2 PDOM Definition

In this section we define the *Positional Data Object Model* (PDOM) of Web pages which the proposed instance learning method is based on. A PDOM is a tree structure where each node, named *positional node* (PNode), represents one or more DOM nodes laid out by the layout engine of a Web browser. PNodes and the PDOM are defined as follows.

Definition 5 (PNode). *Let Λ be a set of tag names, a PNode is a 3-tuple of the form:*

$$PNode = \langle value, mbr, Style \rangle$$

where:
- *value is the value of the node (such as strings, images, etc).*
- *$mbr = \langle (r_x^-, r_y^-), (r_x^+, r_y^+) \rangle$ is a minimum bounding rectangle as defined in Definition 1.*
- *Style represents the set of presentation features of the node.*

Definition 6 (PDOM). *A PDOM is a 3-tuple of the following form:*

$$PDOM = \langle V, root, C \rangle$$

where:
- *V is a set of PNodes (as defined in Definition 5).*
- *root is an unary relation, which contains the root PNode of the tree.*
- *$C \subseteq V \times V$ is the containment relation among PNodes. Let u and w be PNodes in V, u C w holds iff $mbr(u)$ contains $mbr(w)$ and there is not a third node v such that $mbr(u)$ contains $mbr(v)$ and $mbr(v)$ contains $mbr(w)$.*

On PDOMs are defined the following functions:

Definition 7 (Children function). *Let $v \in V$ be a PNode, the function children $: V \to 2^V$ is defined as children$(v) := \{w \in V | v \; C \; w\}$.*

Definition 8 (Leaf function). *Let $u \in V$ be a PNode, the function leaf $: V \to 2^V$ is defined as leaf$(v) := \{w \in V | (mbr(v)$ contains $mbr(w) \vee mbr(v) = mbr(w)) \wedge (\nexists u \in V : w \; C \; u)\}$.*

3.3 PDOM Building

In this section, we describe how PDOMs are created starting from the traditional DOM and considering the rendered Web pages by Web browsers. Layout engines of Web browsers consider the area of the screen aimed at visualizing a Web page, as a 2-dimensional Cartesian plane. They adopt rendering rules that take into account the page DOM structure and the associated *cascade style sheets* (CSS). In the rendered page each DOM node is visualized in a rectangle having sides parallel to the axes of the Cartesian plane. For computing the PDOM, the implemented system embeds the Mozilla browser by exploiting the Mozilla XULRunner[1] application framework that allows for implementing the function *mbr* (see Def. 1). The layout engine assign to each visible DOM node an MBR.

A PNode P can be equivalent to one or more DOM nodes $N = \{n_1 \dots n_k\}$, where $k \geqslant 1$. Let D be a DOM, a PDOM P is built on the base of the containment relations among the MBRs of nodes in D, starting from the root D. For each pair of nodes u and v in D, we have:

- iff $mbr(u) = mbr(v)$, then u and v correspond to a same PNode p.
- iff $mbr(u)$ *contains* $mbr(v)$, then u corresponds to a PNode p_1 and v correspond to a PNode p_2 and $mbr(p_1)$ *contains* $mbr(p_2)$.
- iff $mbr(v)$ *contains* $mbr(u)$, then v corresponds to a PNode p_1 and u correspond to a PNode p_2 and $mbr(p_1)$ *contains* $mbr(p_2)$.
- else, there exists two PNodes p_1 and p_2 such that $mbr(p_1)$ do not intersect $mbr(p_2)$

Let p be a PNode that corresponds to a set of DOM node $N = \{n_1 \dots n_k\}$, where $k \geqslant 1$, then $p.value$, $p.mbr$ and $p.Style$ are computed as follows:

- $p.mbr = mbr(N)$, where the function *mbr* returns the MBR that surrounds one or a set of 2-dimensional objects (N).
- $p.Style$ is the set of attributes and stylistic features contained in CSS that visually describe n_k
- $p.value$ is: (i) the string value of n_k, if $k = 1$ and n_k is a leaf node of string type; (ii) the url of of n_k, if $k = 1$ and n_k is a leaf node of IMG type; (iii) $\emptyset$ otherwise.

In Figure 3 are represented different DOMs (3.a and 3.b) that encode the same logical structure, which is caught by the PDOM derived by the DOMs (3.c).

4 Visual Similarity

In this section we present the novel visual similarity measure between two sets of PDOM nodes which the instance learning algorithm proposed in this paper is founded on. The visual similarity measure is founded on the idea that set

[1] https://developer.mozilla.org/en/XULRunner_1.9.2_Release_Notes

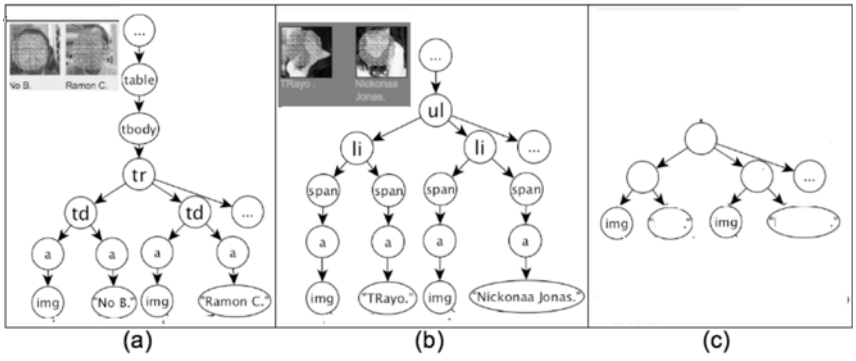

Fig. 3. Tree DOMs of fragments of Web Pages representing friend lists of social networks (a) Bebo and (b) Care sites. (c) The corresponding underlying layout structure that is caught by the PDOM derived by the DOMs.

of PNodes are visually similar if they have the same presentation features and contain the same leaf nodes arranged in the same way. In order to describe the algorithm that computes the visual similarity between two set of PNodes, we first formally define the concepts of *presentation similarity* between two PNodes, and then introduce the concepts of *visual content* of a PNode, and *spatial context* of a set of PNodes, as follows.

Definition 9 (Presentation Similarity). *Let n_1 and n_2 be two PNodes, their presentation similarity is computed by the following formula:*

$$\frac{|n1.Style \cap n2.Style|}{\texttt{max}(|n1.Style|, |n2.Style|)}$$

.

Definition 10 (Visual Content). *Let $n \in V$ be a PNode, the* visual content *of n is the set of leaf nodes spatially contained in n, computed by means of the function Leaf (see Definition 8).*

Definition 11 (Spatial Context). *Let $V_n \subseteq V$ be a set of PNodes, the* spatial context *of V_n is the set of 4-tuples of the form $\langle u, w, \rho, \rho^{-1} \rangle$ where $u, w \in V_n$, $u \neq w$ and ρ, ρ^{-1} are the RCRs that link u and w, where (u ρ w) and (u ρ^{-1} w) hold respectively.*

The spatial context represents for each pair of PNodes in the input set of PNodes reciprocal RCRs. It is computed by means the function $Context : 2^V \rightarrow 2^{V \times V \times RCR \times RCR}$, which for each pair of PNodes in the input set of PNodes compare coordinates and computes the RCR relations.

Now we are ready to present Algorithm 2 that computes the visual similarity.

Algorithm 2. visualSim
Input: Two set of PNodes $V_1 \in V$, and $V_2 \in V$ and threshold λ;
Output: Spatial similarity between V_1, and V_2, having value between $[0, 1]$.
2.1: $L_1 := \bigcup_{v_i \in V_1} leaf(v_i)$;
2.2: $L_2 := \bigcup_{v_i \in V_2} leaf(v_i)$;
2.3: **if**$(|L_1| = 1 \wedge |L_2| = 1)$ **then return** `presentationSim`$(L_1[1], L_2[1])$;
2.4: $s_1 :=$ `Context`(L_1);
2.5: $s_2 :=$ `Context`(L_2);
2.6: $M[][] = \emptyset$
2.7: **for all** $(\langle u, w, \rho, \rho^{-1} \rangle \in s_1)$ **do**
2.8: **for all** $(\langle u', w', \rho', \rho'^{-1} \rangle \in s_2)$ **do**
2.9: $sim_1 :=$ `presentationSim`(u, u');
2.10: $sim_2 :=$ `presentationSim`(w, w');
2.11: **if**$(\rho = \rho')$ **do** $\alpha := 1$ **else** $\alpha := 0$;
2.12: **if**$(\rho^{-1} = \rho'^{-1})$ **do** $\beta := 1$ **else** $\beta := 0$;
2.13: **if** $(sim_1 \geq \lambda \wedge sim_2 \geq \lambda)$ **then**
2.14: $M[\langle u, w, \rho, \rho^{-1} \rangle][\langle u', w', \rho', \rho'^{-1} \rangle] := (\frac{sim_1 + sim_2}{2}) * \frac{\alpha + \beta}{2}$;
2.15: **else**
2.16: $M[\langle u, w, \rho, \rho^{-1} \rangle][\langle u', w', \rho', \rho'^{-1} \rangle] := 0$;
2.17: **end if**
2.18: **end for**
2.19: **end for**
2.20: $simValue := 0$;
2.21: **while** $((vmax := \mathtt{max}_{m \in M}) > 0)$ **do**
2.22: `removeRow`$(M, vmax.rowIndex)$;
2.23: `removeCol`$(M, vmax.colIndex)$;
2.24: $simValue := simValue + vmax$;
2.25: **end while**
2.26: **return** $\sqrt{\frac{simValue}{\mathtt{max}(|s_1|, |s_2|)}}$;

The algorithm takes as input two sets of PNodes (V_1 and V_2) and the presentation similarity threshold λ. It considers visually similar two PNodes if they contain similar leafs (same presentation features) spatially arranged in similar way. If V_1 and V_2 have only one leaf, the visual similarity is given in term of the leaf presentation similarity (instruction 2.3). Else, we consider the spatial contexts of leaf PNodes in V_1 and V_2 (L_1 and L_2). More in details, we computes:

- The spatial contexts s_1 and s_2 (instructions 2.4 and 2.5).
- A partial similarity between all elements in s_1 and s_2 (instructions 2.6-2.19). The partial similarity between elements of the spatial contexts is computed if and only PNodes in the considered elements of the spatial contexts have presentation similarity above the threshold λ (instructions 2.13-2.14).
- The final visual similarity that considers best partial similarity values by using a greedy strategy (instructions 2.20-2.26).

5 Instance Learning Algorithm

In this section we present the instance learning algorithm that extracts data records and items from Deep Web pages by exploiting visual patterns created

by Web designers in order to help human readers in make sense of Deep Web pages contents. The Algorithm 3 takes as input a PDOM and returns a set of data records instances with aligned data items.

Algorithm 3. InstanceLearner
Input: A PDOM P;
Output: A set I of data records instances with aligned data items.
3.1: $Rs :=$ `findDataRegions`(P, λ);
3.2: $R :=$ `maxRegion`(Rs, μ);
3.3: $Rs' :=$ `similarRegions`$(R, (Rs - R))$;
3.4: **for all** $(R \in Rs')$ **do** $rs := rs \cup R.records$;
3.5: $I :=$ `getDataItems`(rs);
3.6: **return** I;

The Algorithm 3 is constituted by two steps described below:

1. *Data region and data record identification.* In this step, the PDOM of a Deep Web page is taken as input and a set of data regions that are portions of Deep Web page containing list or matrices of similar data records are returned (instructions 3.1-3.3). The procedure `findDataRegions` collects PNodes that represent data regions performing a depth-first search along the PDOM in input. The procedure `maxRegion` takes as input found data regions and a threshold μ that represents the minimum number of records that compose a data region, and chooses the region R that has the greatest area. In fact, the size of the most important data region is usually larger than the size of the area of the other data regions. The method `similarRegions` founds similar regions to R. Two regions are considered similar if they are composed by visually similar records, similarity is computed by using the Algorithm 2. This way the algorithm allows for finding data records spread in multiple regions.
2. *Data records and data item extraction.* In this step, the algorithm detects the desired data records and items. Data records are recognized and data items of the same semantics are aligned together (instructions 3.4-3.6) by means of the procedure `getDataItems`.

5.1 Data Region and Data Record Identification

In this section we present the procedure `findDataRegions` that consists in a depth-first search along the PDOM in input. During the depth-first search the procedure calls the `createDataRegion` that is described in the following.

The Procedure 1 allows for checking if a PNode represents a data region. If the analyzed PNode represents a data region, the procedure recognizes the list of its similar data records by exploiting the Procedures `cluster` and `potentialRecords`.

Procedure 1. createDataRegion
Input: A PNode u of a PDOM P, and a threshold λ;
Output: A data region R that consists of a list of records if u represents a data region, *null* otherwise.

1.1: $F := \{\{c\} | c \in \texttt{children}(u)\}$;
1.2: $C := \texttt{cluster}(F, \lambda)$;
1.3: **while** $(\forall c \in C, |c| = 1)$ **do**
1.4: $F := \{\{c\} | c \in \texttt{children}(F)\}$;
1.5: **if** $(F = \varnothing)$ **then return** $\varnothing$;
1.6: $C := \texttt{cluster}(F, \lambda)$;
1.7: **end while**
1.8: $D := \texttt{potentialRecords}(C)$;
1.9: $C'' := \texttt{cluster}(D.nodesets, \lambda)$;
1.10: **if** $(|C''|>1)$ **then return** $\varnothing$;
1.11: **end if**
1.12: $Dr := D$;
1.13: $F := \{\{c\} | c \in children(\bigcup_{i=1}^{i=|D|} D[i].nodes)\}$;
1.14: **while** $(F \neq \emptyset)$ **do**
1.15: $C' := \texttt{cluster}(F, \lambda)$;
1.16: $D' := \texttt{potentialRecords}(C')$;
1.17: $C'' := \texttt{cluster}(D'.nodesets, \lambda)$;
1.18: **if** $(|C''| \neq 1 \vee |D'| < |D|)$ **then return** Dr;
1.19: **if** $(|D'| > |D|)$ **then** $Dr := D'$;
1.20: $D := D'$;
1.21: $F := \{\{c\} | c \in children(\bigcup_{i=1}^{i=|D|} D[i].nodes)\}$;
1.22: **end while**
1.23: **return** Dr;

The Procedure 1 consists of three steps. In the first step (instructions 1.1-1.7) the procedure computes the level of the PDOM containing groups of similar nodes starting from the input node u. In this step the algorithm uses the procedure `cluster` that takes as input a list of sets of PNodes (possibly composed by a single node) and clusters them in order to obtain clusters of PNodes, by using the single linkage clustering strategy and the spatial similarity measure defined in Algorithm 2. In the second step, (instruction 1.8) potential data records are computed exploiting the Procedure 2. If obtained data records are similar, then they can be clustered in the same group, otherwise the input PNode u do not represents a data region (instruction 1.10). In the third step, the procedure decides if the current set of potential data records D is the best set of data records contained in the input PNode u (instructions 1.12-1.23). It considers the next level of the PDOM (instructions 1.13 and 1.21) and checks if the new set of potential data records D' represents a better choice of the current set of potential data records D (instruction 1.19). This inspection is repeated until leaf PNodes, in the portion of PDOM in u, are reached ($F = \emptyset$) or similar potential data records ($|C''| \neq 1$), or D' are worse than D ($|D'| < |D|$).

The Procedure 2, takes as input clusters of similar PNodes and regroups PNodes in order to point out nodes that belong to same candidate data records.

Procedure 2. potentialRecords
Input: A set of Clusters $C = \{C_1, \ldots, C_m\}$ of sets of PNodes;
Output: The set of groups of PNodes D representing potential Data Records.
2.1: **for all** $(C_i \in C)$ **do** $\texttt{sort}(C_i)$; **end for**
2.2: $C_{seed} := \{C_i || C_i| = \texttt{max}(\{|C_1|, \ldots, |C_m|\})\}$;

2.3: $D[] := \emptyset$
2.4: **for all** $(i = 1$ **to** $|C_{seed}|)$ **do** $D[i] := \langle C_{seed}[i], mbr(C_{seed}[i])\rangle$; **end for**
2.5: $C := C - C_{seeds}$;
2.6: **for all** $(C_i \in C)$ **do**
2.7: **for all** $(c \in C_i)$ **do**
2.8: $D[i] := \langle \{D[i].nodes \cup c\}, mbr(mbr(c), D.mbrs)\rangle$
2.9: where $(D[i] \in D \wedge D[i].mbr = \texttt{closest}(mbr(c), D.mbrs))$;
2.10: **end for**
2.11: **end for**
2.12: **return** D;

Procedure 2 consists of three steps: (i) PNodes contained in each cluster of C are sorted in according to their MBR positions in the Web page, from top to bottom and from left to right exploiting the Algorithm 1 (instruction 2.1 in Section 3.1). (ii) The C_{seed} that contains a representative item for each potential data record is chosen. It is the cluster with the maximal cardinality, when more clusters with maximal cardinality exist, the first is chosen (instruction 2.2). Then, the set of groups of PNodes D that represents the set of potential Data Records, is initialized. Each record in D is composed by a PNode in the seed cluster C_{seed} and its MBR (instructions 2.2-2.4). (iii) Each PNode belonging to non-seed clusters are put in the closest potential data record exploiting the `closest` function defined between MBRs (instructions 2.6-2.10).

5.2 Data Records and Data Item Extraction

Up to this point, a set of data regions containing similar data records are recognized. Now, the aim of the Procedure 3 is to recognize and align data items having the same semantics, which compose different data records.

Procedure 3. getDataItems
Input: A set of data record $D = \{R_1, \ldots, R_m\}$, where each data record $R_i \in D$ is represented by a list of leaf PNodes that represents data items;
Output: Aligned records in a $m * n$ matrix I of leaf PNodes, where m is the number of records retrieved in the web page, n is the number of items belonging to each record.

3.1: $R_{seed} := R \in D$, whose $|R|$ is $\texttt{max}\{|R_1| \ldots |R_m|\}$;
3.2: **for all** (i:= 1 **to** $|R_{seed}|$) **do** $I[1][i] := R_{seed}[i]$; **end for**
3.3: $D := D - R_{seed}$;
3.4: **for all** $(R_j \in D)$ **do**
3.5: $M[][] = \emptyset$;
3.6: **for all** $(n_k \in R_j)$ **do**
3.7: **for all** $(n_i \in R_{seed})$ **do**
3.8: $synSim := \texttt{syntacticSim}(n_k, n_i)$;
3.9: $valueSim := 1 - \texttt{editDist}(n_k.value, n_i.value)$;
3.10: $M[j][i] := \frac{synSim + valueSim}{2}$;
3.11: **end for**
3.12: **end for**
3.13: **while** $((vmax := \texttt{max}_{m \in M}) > 0)$ **do**
3.14: $I[j][vmax.colIndex] := vmax$;
3.15: $\texttt{removeRow}(M, vmax.rowIndex)$;

3.16: `removeCol`$(M, vmax.colIndex)$;
3.17: **end while**
3.18: **end for**
3.19: **return** I;

The Procedure 3 takes as input a set of data records $D = \{R_1, \ldots, R_m\}$, and aligns records in a $m*n$ matrix I of leaf PNodes, where m is the number of records retrieved in the web page, and n is the number of items belonging to each record. Because some optional data items can occur, the records having the maximal number of data items is chosen as the representative record (instructions 3.1-3.2). As shown experimentally, this simple method allows for aligning data items, even if it is not completely correct when there are not quite complete records. After that the representative data record is chosen, for each other data record R_j the best data items alignment is found by exploiting a similarity matrix M among the target type of data items (n_i) and the items of the current data record (n_k) (instructions 3.3-3.17).

6 Empirical Evaluation

The instance learning method presented in the paper has been experimentally evaluated on a dataset of 100 Deep Web Pages randomly selected from most known Deep Web Sites. Table 1 reports the precision, recall and F-measure calculated for the proposed method. The table compares results obtained by the approach with results obtained on the same dataset by MDR [7] and ViNTs [18] systems. It is noteworthy that versions of MDR and ViNTs available on the Web allow for performing only data record extraction.

Table 1. Precision, Recall and F-Measure of the Proposed Instance Learning Method

	Records			**Items**		
	P	R	F	P	R	F
Proposed Instance Learning Method	96.01%	94.33%	95.16%	93.62%	99.01%	96.24%
MDR	24.26%	42.85%	30.98%	–	–	–
ViNTs	51.52%	47.46%	49.41%	–	–	–

7 Conclusions and Future Work

In this paper has been presented a novel spatial instance learning method for Deep Web pages. The method is based on: (i) a novel positional document object model that represents both spatial and visual features of data records and data items/fields produced by layout engines of Web browser in rendered Deep Web pages; (ii) a novel visual similarity measure that exploit the rectangular cardinal relation spatial model for computing visual similarity between nodes of the PDOM. Experiments carried out on 100 Deep Web pages randomly selected from well known Deep Web sites, show very high precision and recall. Most importantly, experiments show that the wrapper induction algorithm enables to identify data records and items spread on multiple (data) regions of

a single page, and to recognize data records and items having many different spatial arrangement (i.e. data records arranged either as lists or matrices having data items indifferently organized in vertical and horizontal way). Future work will be aimed at extending the method towards spatial wrappers learning. This way information will be extracted from Deep Web pages by applying spatial wrappers on PDOM representations directly.

References

1. Baumgartner, R., Flesca, S., Gottlob, G.: Visual web information extraction with lixto. In: VLDB, pp. 119–128. Morgan Kaufmann Publishers Inc., San Francisco (2001)
2. Baumgartner, R., Gottlob, G., Herzog, M.: Scalable web data extraction for online market intelligence. VLDB 2(2), 1512–1523 (2009)
3. Cai, D., Yu, S., Wen, J., Ma, W.-Y.: Extracting content structure for web pages based on visual representation. In: Zhou, X., Zhang, Y., Orlowska, M.E. (eds.) APWeb 2003. LNCS, vol. 2642, pp. 406–417. Springer, Heidelberg (2003)
4. Crescenzi, V., Mecca, G., Merialdo, P.: Roadrunner: Towards automatic data extraction from large web sites. In: VLDB, pp. 109–118. Morgan Kaufmann Publishers Inc., San Francisco (2001)
5. Gottlob, G., Koch, C., Baumgartner, R., Herzog, M., Flesca, S.: The lixto data extraction project: back and forth between theory and practice. In: PODS, pp. 1–12 (2004)
6. Hsu, C.N., Dung, M.T.: Generating finite-state transducers for semi-structured data extraction from the web. Inf. Syst. 23, 521–538 (1998)
7. Liu, B., Grossman, R., Zhai, Y.: Mining data records in web pages. In: KDD 2003: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 601–606. ACM, New York (2003)
8. Liu, W., Meng, X., Meng, W.: Vide: A vision-based approach for deep web data extraction. IEEE Trans. on Knowl. and Data Eng. 22(3), 447–460 (2010)
9. Mehta, R.R., Mitra, P., Karnick, H.: Extracting semantic structure of web documents using content and visual information. In: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web, WWW 2005, pp. 928–929. ACM, New York (2005)
10. Muslea, I., Minton, S., Knoblock, C.A.: Hierarchical wrapper induction for semistructured information sources. Autonomous Agents and Multi-Agent Systems 4(1-2), 93–114 (2001)
11. Navarrete, I., Sciavicco, G.: Spatial reasoning with rectangular cardinal direction relations. In: ECAI, pp. 1–9 (2006)
12. Oro, E., Ruffolo, M., Staab, S.: Sxpath - extending xpath towards spatial querying on web documents. PVLDB 4(2), 129–140 (2010)
13. Papadakis, N.K., Skoutas, D., Raftopoulos, K., Varvarigou, T.A.: Stavies: A system for information extraction from unknown web data sources through automatic web wrapper generation using clustering techniques. TKDE 17(12), 1638–1652 (2005)
14. Sahuguet, A., Azavant, F.: Building intelligent web applications using lightweight wrappers. DKE 36(3), 283–316 (2001)

15. Simon, K., Lausen, G.: Viper: augmenting automatic information extraction with visual perceptions. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, CIKM 2005, pp. 381–388. ACM, New York (2005)
16. Zhai, Y., Liu, B.: Web data extraction based on partial tree alignment. In: WWW, pp. 76–85. ACM, New York (2005)
17. Zhai, Y., Liu, B.: Structured data extraction from the web based on partial tree alignment. TKDE 18(12), 1614–1628 (2006)
18. Zhao, H., Meng, W., Wu, Z., Raghavan, V., Yu, C.: Fully automatic wrapper generation for search engines. In: WWW, pp. 66–75. ACM, New York (2005)