Journal of Computer and Communications, 2016, 4, 36-48
Published Online May 2016 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2016.46004

Scientific Research Publishing

Supporting Information Extraction from Visual Documents

Giuseppe Della Penna, Sergio Orefice

Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila Via Vetoio, L'Aquila, Italy
Email: giuseppe.dellapenna@univaq.it, sergio.orefice@univaq.it

Received 10 March 2016; accepted 27 May 2016; published 30 May 2016

Abstract

Visual Information Extraction (VIE) is a technique that enables users to perform information extraction from visual documents driven by the visual appearance and the spatial relations occurring among the elements in the document. In particular, the extractions are expressed through a query language similar to the well known SQL. To further reduce the human effort in the extraction task, in this paper we present a fully formalized assistance mechanism that helps users in the interactive formulation of the queries.

Keywords

Information Extraction, Spatial Relations, Visual Appearance

1. Introduction

The process of knowledge acquisition from generic information domains has its central phase in the Information Extraction (IE), which aims to extract from the located documents relevant information that appear in certain semantic or syntactic relationships [1]. In particular, IE tries to process the relevant information found on the documents in order to make it available to structured queries. Most often, information extraction systems are customised for specific application domains, and require manual or semi-automatic training sessions.

In order to apply IE to visual documents, we have developed a technique called *Visual Information Extraction* (VIE) that is an IE approach based on the visual appearance of the information, conceived as its user-perceived rendering. This allows one to shift the IE problem from the low level of code (e.g., raster graphics, vector drawing, word processor formatted text, web page, etc.) to the higher level of visual features, providing a paradigm of the kind "what you see drives your search" that supports a natural query formulation. In particular, the extractions are based on the spatial arrangement occurring among the objects of a document which is modelled by

How to cite this paper: Penna Della, G. and Orefice, S. (2016) Supporting Information Extraction from Visual Documents. *Journal of Computer and Communications*, **4**, 36-48. http://dx.doi.org/10.4236/jcc.2016.46004

common qualitative spatial relations like topological (*i.e.*, overlapping, adjacency, containment) and direction (*i.e.*, left, up, etc.) relations. Then, the focus of our approach is on the spatial feature, which best characterises it and also represents the main novelty with respect to the current IE literature.

The origin of the VIE technique can be found in [2] [3]. In those works, the approach is inspired by the *box* spatial relation theory [4], such that graphical objects are syntactically described and manipulated through their bounding boxes whatever the shape of the object is. However, the box formalism has some intrinsic limitations that make no possible to use it in contexts where graphical objects are represented by complex figures. The major shortcoming of such an early proposal is that the Visual Information Extraction technique is strongly customised to specific application domains. Indeed, in [2] [3] the VIE framework is ad-hoc adapted to perform extractions on web pages, PDF documents, respectively, and the technique presented in each of these works is ad-hoc customised for the corresponding domain. Finally, in [5] the VIE theory is refined, allowing to work directly on the real shapes, and generalised in order to be applied to a wide range of interesting visual domains, including geospatial data.

The Visual Information Extraction technique has been accomplished thanks to an extraction language, namely the Spatial Relation Query Language (SRQL), which allows users to write SQL-like queries based on the visual arrangement of the information. Moreover, both the spatial relation theory and the SRQL language are implemented in a graphical software system, the Spatial Relation Query (SRQ) tool, which also allows the use of further semantic information to refine the queries, thus providing an environment where the information extraction can be performed integrating spatial relations, visual attributes, textual content, etc.

The main purpose of the research presented in this paper is to further reduce the human effort needed to compose the extraction queries through an assistance mechanism that may be used to interactively formulate the queries. To this aim, we have developed a spatial relation-based assistance technique whose theoretical issues are described in this paper. In particular, we reused the spatial relation formalism of [5] (which has been extended with connection-based relations in order to model also interconnections among objects) and we extensively developed the information extraction task theory based upon the new notion of pictorial view, providing a more refined specification of the queries that has been integrated in the algorithm for the assisted formulation of the queries.

The paper is organised as follows. Related work is summarised in Section 2. In Section 3 we illustrate the spatial relation formalism underlying the VIE technique, whereas in Section 4 we introduce the notion of Pictorial View and give an in-depth overview of the formalisation of the query process. Section 5 contains the algorithm for the interactive formulation of the queries together with some experimentation examples. Finally, some concluding remarks are outlined in Section 6.

2. Related Work

Information extraction is a very active research area that has received a growing attention from different communities, such as the Artificial Intelligence, Information Retrieval and Processing and Web communities.

In particular, the enormous amount of information present in the web makes it the most appealing domain for developing new information extraction techniques. In the last decade, many approaches have been proposed in this field, and they are usually classified in the following categories (for an extended survey of these approaches we refer the reader to [6] [7]):

- *HTML-aware tools* (e.g., LIXTO [8] and SCRAP [9]), that make use of the HTML parse trees to create extraction rules.
- *Wrapper induction tools* (e.g., DEPTA [10]), where the extraction rules are derived from a given set of training examples or using pattern discovery techniques.
- *NLP-based tools* (e.g., WHISK [11] or the technique in [12]), where the rules are derived by considering phrases and sentences and applying either syntactic analysis, filtering, part-of-speech tagging or lexical semantic tagging.
- *Modelling-based tools* (e.g., DEByE [13] or the Bayesian learning framework in [14]), which rely on locating in the web page portions of data conforming to a predefined structure.
- *ontology-based tools* (e.g., WeDaX [15] or [16]), that are based on the data and not on the structure of the source documents, and ontologies are used to locate constants in the page and construct objects with them.

Furthermore, a number of languages for wrappers development have been designed (e.g., Minerva [17] or

TSIMMIS [18]) as well as specific-targeted approaches tools like WiNTs [19], VENTex [20], GRAPE [21] or SRES [22].

What differentiates our approach from the cited works is that we directly exploit the visual appearance of the information, rather than relying on the structural and textual information. Furthermore, we do not require any predefined model (as it is the case for the approaches based on training sets, ontologies or wrappers).

Information Extraction literature does include works dealing with the visual appearance of information (see, e.g., [19] [20] [23] [24]), but they are based on visual web page analysis and are targeted to specific tasks (like table extraction or similarities detection). Finally, in [25] the authors propose a visual information extraction algorithm which is specifically targeted to extraction from PDF documents.

Some more recent works are presented in [26]-[28]. The former presents an algebra for expressing spatial and textual rules that can be defined directly at a layout level. It is an interesting approach that, however, is restricted to the web-page domain. On the other hand, the work in [27] deals with the domain of charts, and proposes an approach for improving their accessibility. Finally, the work in [28] presents an hybrid approach based on decision tree learning and extracting rules.

3. Spatial Relation Theory

In this section we provide the basic notions of the spatial relation theory underlying our VIE technique.

3.1. Graphical Objects

A graphical object is formally defined as pair $O=(C,A)$, where C denotes the set of all the points $p\in\mathbb{R}^2$ forming the external contour of O (which is disjoint from its internal area), and A is the set of the object *attributes*. Each attribute is a pair (a,v), where a is a property name and v its value. More in detail, A models the object semantic properties (e.g., *name*, *colour*, *font* or *latitude*), and never includes attributes depending on C (such as the object width, area, etc.), whereas C is used essentially for its syntactic manipulation.

We consider only graphical objects which do not contain *holes* and have a contour that can be modelled as a *closed curve* without self loops (*simple curve*). We do this since this restriction still defines a wide and significant domain, excluding only graphical objects with very irregular shapes, which would be hard to handle and, often, of little interest.

In the following, we shall write $p\in C$ to indicate a generic point of C and p_x, p_y to indicate the x and y-coordinate, respectively, of p. Moreover, we shall use the expression $\mathrm{Inside}(O)$ to indicate the set of points *enclosed* by the contour C of the graphical object O, that we shall call the *internal points* of O. Finally, we shall write $a\in A$ to indicate a generic attribute in A and shall use $attr(a)$ and $val(a)$ to refer to its property name and value, respectively, and, for any attribute a with $attr(a)=x$, we shall use the common object-oriented notation $O.x$ to denote $val(a)$.

Figure 1 shows a fragment of a geographic map containing five nations (*a*, *c*, *e*, *f*, *g*), a lake (*b*) and a river (*d*). All these map symbols can be naturally modelled as graphical objects. As an example, let us give a possible definition of the graphical object O_e corresponding to the nation *e* on the map. In this case, the component C of O_e would be the set of points forming the nation contour highlighted in the figure, whereas the graphical object attributes A may be, e.g., the set

$$\{(\text{name},e),(\text{latitude},150),(\text{longitude},20),(\text{population},20000),(\text{map\_colour},\text{green})\}$$

3.2. Spatial Relations

In this section, we briefly illustrate the spatial relation formalism underlying the VIE technique.

So far [5], we have considered only position-based spatial relations, describing where a graphical object is placed relative to one other. Below we give a brief summary of such relations.

3.2.1. Position-Based Relations

The position-based relations can be subdivided in three classes:

- *disjoint* relations, holding between objects with disjoint contours and areas (e.g., “left of”, “above”, “close to”);

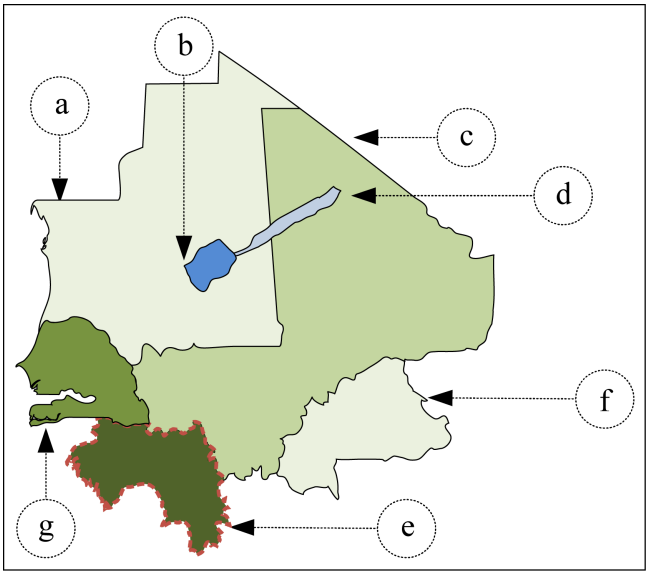


Figure 1. A sample geographic map.

- *overlap* relations, holding between objects with intersecting areas and possibly intersecting contours (*i.e.*, "full inclusion" or "partial intersection");
- *attach* relations, holding between objects with intersecting contours but not areas (*i.e.*, adjacency in one or more points).

The four distinct *disjoint* spatial relations are *UP*, *DOWN*, *LEFT* and *RIGHT*, and are sufficient to model any disjoint spatial arrangement between two graphical objects. Actually, since of course *UP* and *DOWN* are inverse of each other, as well as *LEFT* and *RIGHT*, we could restrict the set of disjoint spatial relations to just one of the possible pairs, e.g., *LEFT* and *DOWN*. However, for sake of completeness, in the following we will present the complete set of disjoint spatial relations.

In order to define them, we need to introduce the following terminology for a generic graphical object $O=(C,A)$. Note that, in our formalism, we refer to the canonical orientation of the cartesian axes (*i.e.*, the *x* coordinate increases rightwards, and the *y* one increases upwards).

1) $y_{UM(C)}$ represents the highest *y* coordinate of the points in the contour *C*, that is the *y* coordinate of each point in $UM(C)=\{p\in C \mid \forall p'\in C, p'_y \le p_y\}$, where $UM(C)$ denotes the set of upmost points of the contour *C*.

2) $y_{DM(C)}$ represents the lowest *y* coordinate of the points in the contour *C*, that is the *y* coordinate of each point in $DM(C)=\{p\in C \mid \forall p'\in C, p'_y \ge p_y\}$, where $DM(C)$ denotes the set of downmost points of the contour C.

3) $y_{LM(C)}$ represents the lowest *x* coordinate of the points in the contour *C*, that is the *x* coordinate of each point in $LM(C)=\{p\in C \mid \forall p'\in C, p'_x \ge p_x\}$, where $LM(C)$ denotes the set of leftmost points of the contour *C*.

4) $y_{RM(C)}$ represents the highest *x* coordinate of the points in the contour *C*, that is the *x* coordinate of each point in $RM(C)=\{p\in C \mid \forall p'\in C, p'_x \le p_x\}$, where $RM(C)$ denotes the set of rightmost points of the contour *C*.

Then, given two graphical objects $a=(C,A)$ and $b=(C',A')$, the four disjoint spatial relations are defined as in **Table 1**. Intuitively, the *UP* relation models a spatial arrangement between *a* and *b* whenever the graphical object *b* is completely (both internal points and external contour) below the graphical object *a*. For example, **Figure 2(a)** shows a spatial arrangement where the *UP* relation holds between an hexagon *a* and a circle *b*. Note that, in the figure, the round callouts are not part of the graphical objects, but are only used to indicate their names.

The two *overlap* spatial relations *INCLUDE* and *INTERSECT* are defined as shown in **Table 2**.

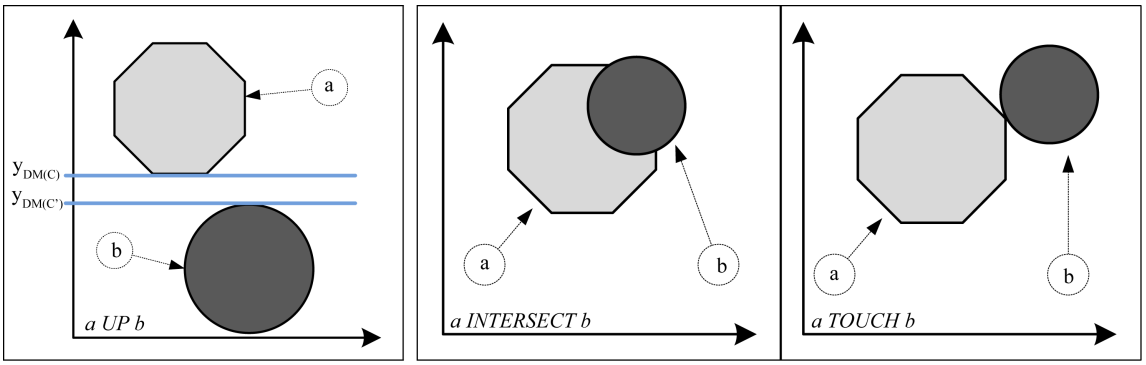
In particular, the *INCLUDE* relation models a spatial arrangement between *a* and *b* whenever the graphical object *b* is properly contained in *a*, and then their external contours do not intersect.

Table 1. Definition of disjoint spatial relations.

UP	$a \; UP \; b \Leftrightarrow y_{UM(C')} \leq y_{DM(C)}$
DOWN	$a \; DOWN \; b \Leftrightarrow y_{UM(C)} \leq y_{DM(C')}$
LEFT	$a \; LEFT \; b \Leftrightarrow x_{RM(C)} \leq x_{LM(C')}$
RIGHT	$a \; RIGHT \; b \Leftrightarrow x_{RM(C')} \leq x_{LM(C)}$

Table 2. Definition of overlap spatial relations.

INCLUDE	$a \; INCLUDE \; b \Leftrightarrow \left(\text{Inside}(b) \subset \text{Inside}(a)\right) \wedge \left(\nexists p \in C' : p \in C\right)$
INTERSECT	$a \; INTERSECT \; b \Leftrightarrow \left(\text{Inside}(a) \cap \text{Inside}(b) \neq \varnothing\right) \wedge \left(\exists p \in C' : p \in C\right)$



(a) UP (b) INTERSECT (c) TOUCH

Figure 2. Spatial relations *UP*, *INTERSECT* and *TOUCH*.

On the other hand, the *INTERSECT* relation models a spatial arrangement between *a* and *b* whenever there is a partial or full overlapping of their internal points, and there is an intersection between their external contours.

For example, **Figure 2(b)** shows a spatial arrangement where the *INTERSECT* relations holds between the objects *a* and *b*, respectively.

Finally, the *attach* spatial relation *TOUCH* is defined as shown in **Table 3**.

As a matter of fact, the *TOUCH* relation models a spatial arrangement between *a* and *b* whenever there is no overlapping of their internal points, but there is an intersectionn between their external contours.

For example, **Figure 2(c)** shows a spatial arrangement where the *TOUCH* relation holds between the objects *a* and *b*.

3.2.2. Connection-Based Relations

In this section, let us introduce the new connection-based spatial relation *LINK*(*h*,*k*), which models the inter connections among graphical objects, and is defined as shown in **Table 4**. Note that *LINK*(*h*,*k*) is an explicit relation, therefore it is visualised through a polyline.

In other words, *LINK*(*h*,*k*) is a relation that models a spatial arrangement between graphical objects *a* and *b* whenever they are disjoint and there is a connection between a point $p \in h \subseteq C$ and a point $p' \in k \subseteq C'$. For example, **Figure 3** shows a spatial arrangement where the *LINK*(*h*,*k*) relation holds between the objects *a* and *b*. It is worth noting that, in the definition of *LINK*(*h*,*k*) given in **Table 4**, *h* and *k* are predefined (both in number and position) subsets of *C* and C', respectively.

In particular, in general graphs (e.g., state-transition diagrams, see **Figure 4(a)**), where the specific connection point is not relevant for the relation semantics, these subsets coincide with the overall contours, whereas in "puzzle" like languages (e.g., tiles, see **Figure 4(b)**) they coincide with predefined contiguous segments of the contours. In the latter case, the specific connection point is still not relevant, as long as it belongs to one of the predefined segments. Finally, in plex ([29]) languages (e.g., flow charts, see **Figure 4(c)**) *h* and *k* contain specific predefined points of the contours, which determine the relation semantics. For example, in **Figure 4(c)**), the flowchart diamond *D* has exactly three connection points, with a precise position on the contour of the object,

Table 3. Definition of the *TOUCH* spatial relation.

TOUCH	$a\ TOUCH\ b \Leftrightarrow \left(\text{Inside}(a) \cap \text{Inside}(b) = \varnothing\right) \wedge \left(\exists p \in C' : p \in C\right)$

Table 4. Definition of the *LINK(h,k)* spatial relation.

LINK(*h*,*k*)	$a\ LINK(h,k)\ b \Leftrightarrow \left(\text{Inside}(a) \cap \text{Inside}(b) = \varnothing\right) \wedge \left(\nexists p \in C' : p \in C\right)$ $\wedge \left(\exists p \in h \subseteq C,\ p' \in k \subseteq C' : p \text{ is connected to } p'\right)$

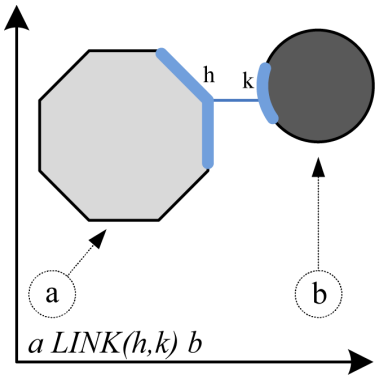
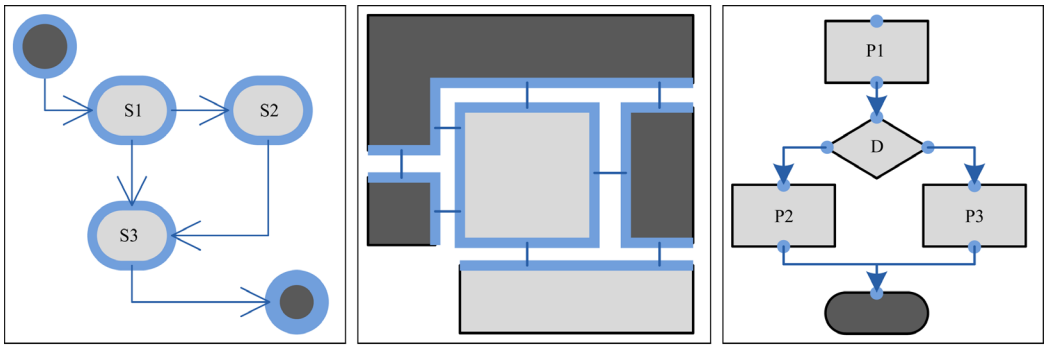


Figure 3. Spatial relation *LINK*(*h*,*k*).



(a) Graph (b) Puzzle (c) Plex

Figure 4. Examples of interconnections.

corresponding to the input flow, the "true" output flow and the "false" output flow, respectively.

4. VIE Query Theory

In this section we provide the theoretical foundations of the VIE query task showing how to formalise queries to extract information from visual documents, *i.e.*, generic documents composed of graphical objects with attributes arranged in a two-dimensional space.

To begin, in order to formally represent a visual document let us introduce the notion of *pictorial view*, which is defined as a set of triples

$$D = \left\{\left(O_1, R_1, O_1'\right), \cdots, \left(O_n, R_n, O_n'\right)\right\}$$

where O_i, O_i' are *graphical objects* and R_i is a *spatial relation*. The triple $\left(O_i, R_i, O_i'\right)$ means that the spatial relation R_i holds between the objects O_i and O_i' in the given visual representation.

In other words, we represent visual documents through the notion of *pictorial view*, conceived as a collection of *graphical objects* and a set of *spatial relations* arranging them in a two dimensional space. Pictorial views allow one to characterise the essential content of visual representations, including only the knowledge strictly necessary to manipulate its graphical objects.

Indeed, a visual document can have many different pictorial views, each one containing only the part of the overall information which is strictly necessary for that specific view. Therefore, a pictorial view is not required to refer to all the graphical objects present in a visual document or list all the relations holding among them.

Then, we can define in detail the structure of a VIE query, modelling the task of information extraction from a pictorial view. In the following, we use P to denote a generic pictorial view, $\mathcal{G} = \{O_1, \cdots, O_{gn}\}$ and $\mathcal{R} = \{R_1, \cdots, R_{rn}\}$ to denote the set of graphical objects and spatial relations included in P, respectively, and $\mathcal{A} = \{a_1, \cdots, a_{an}\}$ a set of attribute names relative to the graphical objects in $\mathcal{G}$.

Definition 1 (Query) *A VIE query Q can be*:

1) a *simple query*, *i.e.*, a sequence $Q = \langle s_1, s_2, \cdots, s_k \rangle$ of *query steps*;

2) a *combined query*, *i.e.* the combination of two queries through the set operations $\cup$ (union), $\cap$ (intersection), \(difference), e.g. $Q = Q_1 \cup Q_2$.

Definition 2 (Query step) *A query step s can be one of the following*:

1) a *direct relation step*, written as R or $\vec{R}$, with $R \in \mathcal{R}$;

2) an *inverse relation step*, written as R or $\overleftarrow{R}$, with $R \in \mathcal{R}$;

3) a *global* (*direct or inverse*) *relation step*, written as $\forall R$ (direct) or $\forall \vec{R}$ (inverse), with $R \in \mathcal{R}$;

4) an *attribute filter step*, written as a sequence $(a_1 op_1 v_1, \cdots, a_k op_k v_k)$, where $a_i \in \mathcal{A}$, v_i is a constant value and $op_i \in \{<, >, =, <>, <=, >=\}$;

5) a *order step*, written as $\text{Order}(a, d)$, where $a \in \mathcal{A} \cup \{\$UM, \$DM, \$LM, \$RM\}$, $d \in \{ASC, DESC\}$ and $\{\$UM, \$DM, \$LM, \$RM\}$ are "internal attributes" denoting the uppermost and downmost y-coordinate and the leftmost and rightmost x-coordinate of a point in a object contour, respectively;

6) a *extract step*, written as $First()$ or $Last()$;

7) a *subquery step*, written as a valid query Q'.

Definition 3 (Query Context) *A query context is an ordered list of graphical objects from P, written as* $C = \langle O_1^c, \cdots, O_{cn}^c \rangle$ *with* $O_i^c \in \mathcal{G}$.

Definition 4 (Query Results) *A query Q, executed on a pictorial view P in a context* $C = \langle O_1^c, \cdots, O_{cn}^c \rangle$, *returns a list of graphical objects denoted by* $Apply(Q, P, C)$, *which is recursively defined as follows.*

For simple queries:

1) if $Q = \langle R \rangle$, then $\text{Apply}(Q, P, C) = \langle O \in \mathcal{G} \mid \exists O' \in C, O' \; R \; O \in P \rangle$

2) if $Q = \langle \vec{R} \rangle$, then $\text{Apply}(Q, P, C) = \langle O \in \mathcal{G} \mid \exists O' \in C, O \; R \; O' \in P \rangle$

3) if $Q = \langle \forall R \rangle$, then $\text{Apply}(Q, P, C) = \langle O \in \mathcal{G} \mid \forall O' \in C, O' \; R \; O \in P \rangle$

4) if $Q = \langle \forall \vec{R} \rangle$, then $\text{Apply}(Q, P, C) = \langle O \in \mathcal{G} \mid \forall O' \in C, O \; R \; O' \in P \rangle$

(note that the actual order of objects in the result of these steps is left unspecified).

5) if $Q = \langle (a_1 op_1 v_1, \cdots, a_k op_k v_k) \rangle$, then $\text{Apply}(Q, P, C) = \langle O_{l_1}^c, \cdots, O_{l_n}^c \rangle$ with $l_i < l_{i+1}$ and $\forall j = 1 \cdots k, O_{l_i}^c . a_j \; op_j \; v_j$ holds.

6) if $Q = \langle \text{Order}(a, d) \rangle$, then $\text{Apply}(Q, P, C) = C$ ordered w.r.t. the ascending/descending (d) value of attribute a.

7) if $Q = \langle \text{First}() \rangle$, then $\text{Apply}(Q, P, C) = \langle O_1^c \rangle$.

8) if $Q = \langle \text{Last}() \rangle$, then $\text{Apply}(Q, P, C) = \langle O_{cn}^c \rangle$.

9) if $Q = \langle Q' \rangle$, then $\text{Apply}(Q, P, C) = \text{Apply}(Q', P, C)$.

10) if $Q = \langle s_1, \cdots, s_k \rangle$, then $\text{Apply}(Q, P, C) = \text{Apply}(\langle s_k \rangle, P, \text{Apply}(\langle s_1, \cdots, s_{k-1} \rangle, P, C))$.

For combined queries, let Q_1, Q_2 be two (sub) queries, with $\text{Apply}(Q_1, P, C) = \langle O_1^{r1}, \cdots, O_{r1n}^{r1} \rangle$, $\text{Apply}(Q_2, P, C) = \langle O_1^{r2}, \cdots, O_{r2n}^{r2} \rangle$:

11) if $Q = Q_1 \cup Q_2$, $\text{Apply}(Q, P, C) = \langle O_1^{r1}, \cdots, O_{r1n}^{r1}, O_1^{r2}, \cdots, O_{r2n}^{r2} \rangle$.

12) if $Q = Q_1 \cap Q_2$, $\text{Apply}(Q, P, C) = \langle O_{l_1}^{r1}, \cdots, O_{l_n}^{r1} \rangle$ with $l_i < l_{i+1}$ and $\exists j \mid O_{l_i}^{r1} = O_{l_j}^{r2}$.

13) if $Q = Q_1 \setminus Q_2$, $\text{Apply}(Q, P, C) = \langle O_{l_1}^{r1}, \cdots, O_{l_n}^{r1} \rangle$ with $l_i < l_{i+1}$ and $\nexists j \mid O_{l_i}^{r1} = O_{l_j}^{r2}$.

To better understand how a VIE query extracts graphical objects from a pictorial view following the rules introduced in the above query results definition, let us consider the visual document depicted in **Figure 5**. For sake of simplicity, we avoid to introduce specific pictorial views of this document, and in the queries reported below we consider only spatial relations (e.g., *Left* or *Right*) and attributes (e.g., the *shape*) which can be clearly desumed from that document. In other words, the following queries will be based on an overall pictorial view containing all the ten graphical objects together with all the disjoint spatial relations holding among them.

- $Q_1 = \langle (\text{shape} = \text{square}), \text{Left} \rangle$

  (this query extracts all the objects having a square on the left, *i.e.* T1, T3, T4, C2, C3, S3)
- $Q_2 = \langle (\text{shape} = \text{circle}), \text{Right} \wedge \text{Up} \rangle$

  (this query extracts all the objects having a circle on the upright, *i.e.*, T2, S2, T3, T4)
- $Q_3 = Q_1 \cap Q_2$

  (this query combines the previous Q_1 and Q_2 and allows to extract all the objects having a square on the left and a circle on the upright, *i.e.*, T3, T4)
- $Q_4 = \langle Q_3, \text{Order}(\$UM, ASC), \text{First}(\,) \rangle$

  (this query uses Q_3 as a subquery in order to extract the first object in y-order having a square on the left and a circle on the upright, *i.e.*, T3)

5. Assisted Query Formulation

In the Introduction, we outlined the origin and meaning of the VIE research, whereas in the previous section we presented the new VIE query theory. In this section, we show how the VIE framework has been upgraded by developing an incremental algorithm to assist the user in the interactive formulation of extraction queries. The algorithm follows the query formalisation illustrated in Section 4, which as been implemented in a prototypal software application written in Java. The overall query definition algorithm is shown in **Figure 6**.

Starting form a pictorial view (which is shown in the application interface) and an empty query (which conventionally selects all the graphical objects of the pictorial view), in each iteration the application highlights the objects returned by the current query and allows the user to choose the next step to add to the query. Then, the objects resulting from the application of this step to the current query are previewed on the interface. Once the user confirms the step, the query is updated to the current results. This process is repeated until the user confirms the query, which is then stored in the application query library.

More in detail, **Figure 7** show the specific algorithm fragments underlying the **Get_Step** function. As an example, as shown in **Figure 7(a)** after selecting the "Relation Step" mode, the user should choose a feasible graphical object from the highlighted ones. Then, the interface opens a menu displaying the feasible spatial relations holding between the selected object and the other ones, and the user can select one of these relations. This process is repeated so that the user can select more objects and relative relations, which are combined through the AND operator. Finally, the corresponding direct relation step (which is the only relation step currently supported by the application) is built and returned to the overall algorithm.

Figures 8-11 show an example where the above algorithm is executed during a sample running session of the

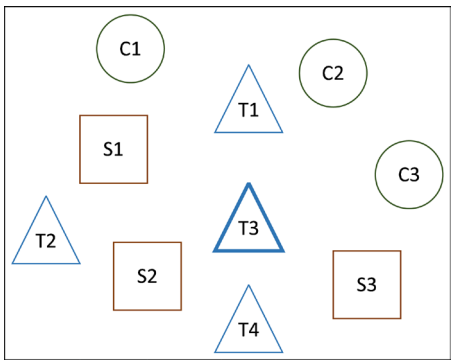


Figure 5. A sample visual document.

```
INPUT: Pictorial View P

List_Of_Objects Ctx;
Query Q;
Step S;

Ctx = Objects(P)
Query = ∅

Show(P);
repeat {
 Highlight(Ctx);
 repeat {
  S = Get_Step();
  Fade(Objects(P)-Apply(S,P,Ctx));
 } until (user confirms step)
 Q = Q · S;
 Ctx = Apply(S,P,Ctx);
} until (user confirms query)
Store_to_Query_Library(Q)
```

Figure 6. Overall query definition algorithm.

```
INPUT: Pictorial View P
OUTPUT: Step S;
PRECONDITION: User selects Relation Step Mode

Graphical_Object O;
Spatial_Relation R,Rc;
Rc = true;

repeat {
 O = Get_User_Selection();
 Display_Relations_Holding_From(O)
 R = Get_User_Selection();
 Rc = Rc ∧ R;
} until (user confirms selection)
S = new Step ⟨Rc⟩;
return S;
```

(a) Relation step algorithm

```
INPUT: Pictorial View P
OUTPUT: Step S;
PRECONDITION: User selects Attribute Filter
 Mode

Graphical_Object O;
Attribute A ;
Set_of_Attribute_Comparisons Ac;
Ac = ∅;

repeat {
 O = Get_User_Selection();
 Display_Attributes_Of(O);
 A = Get_User_Selection();
 (op,val) = Execute_Constraint_Dialog(A);
 Ac = Ac ∪ (A op val);
} until (user confirms selection)
S = new Step ⟨Ac⟩;
return S;
```

(b) Attribute filter step algorithm

```
INPUT: Pictorial View P
OUTPUT: Step S;
PRECONDITION: User selects Order Mode

Graphical_Object O;
Attribute A ;
Direction D;

//returns ASC or DESC
D = Execute_Direction_Dialog();
O = Get_User_Selection();
Display_Attributes_Of(O);
A = Get_User_Selection();
S = new Step ⟨Order(A, D)⟩;
return S;
```

(c) Order step algorithm

```
INPUT: Pictorial View P
OUTPUT: Step S;
PRECONDITION: User selects Extract Mode

Extract_Function E;
//returns First() or Last()
E = Execute_Extract_Dialog();
S = new Step ⟨E⟩;
return S;
```

(d) Extract step algorithm

```
INPUT: Pictorial View P
OUTPUT: Step S;
PRECONDITION: User selects Subquery Mode

Query Q;

Show_Query_Library();
Q = Get_User_Selection(); //returns selected
  query
S = new Step ⟨Q⟩;
return S;
```

(e) Subquery step algorithm

Figure 7. Specific query step definition algorithms.

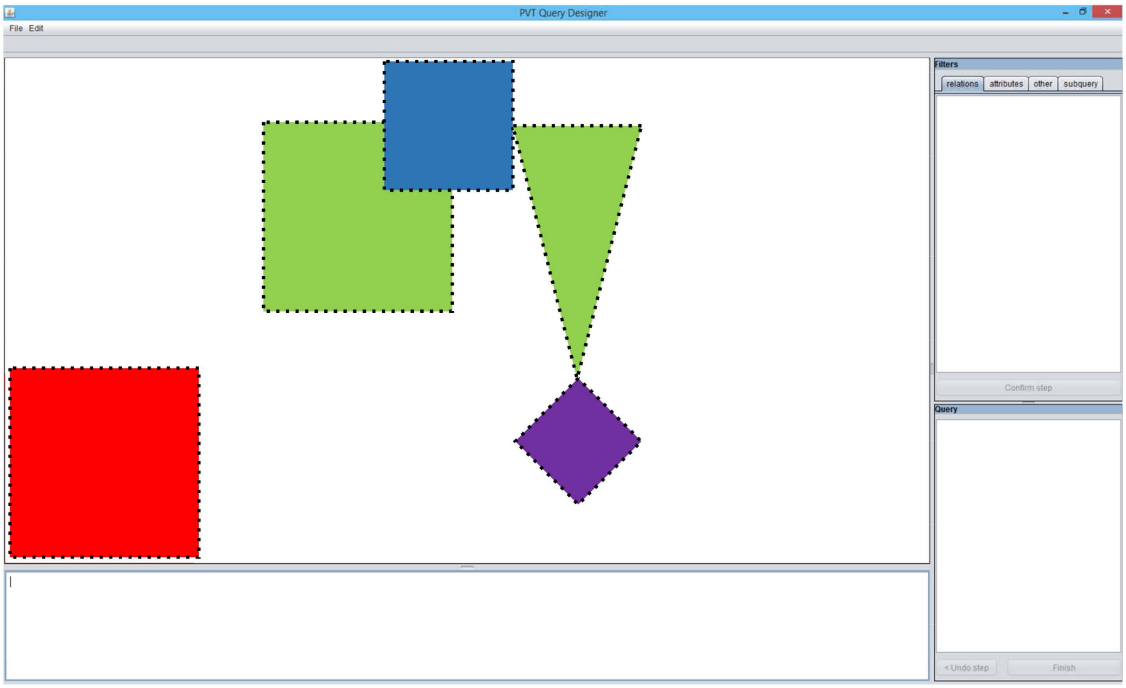
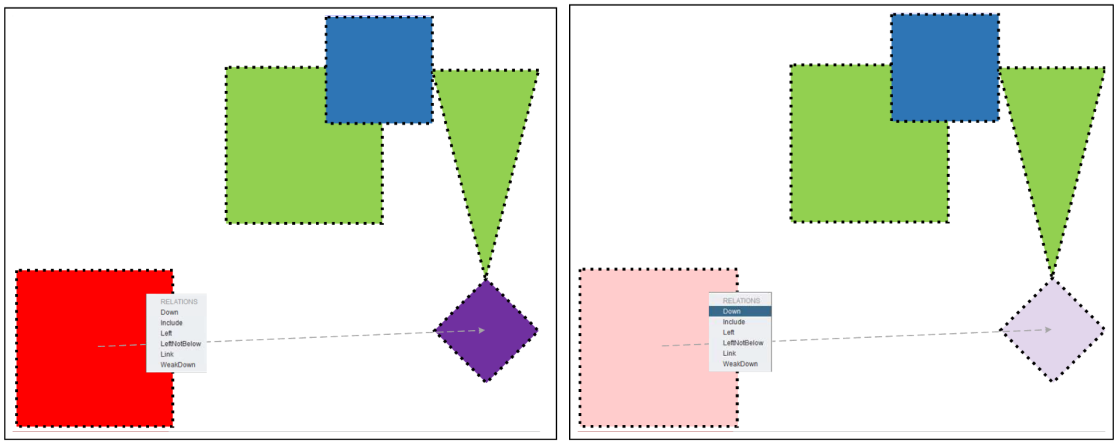
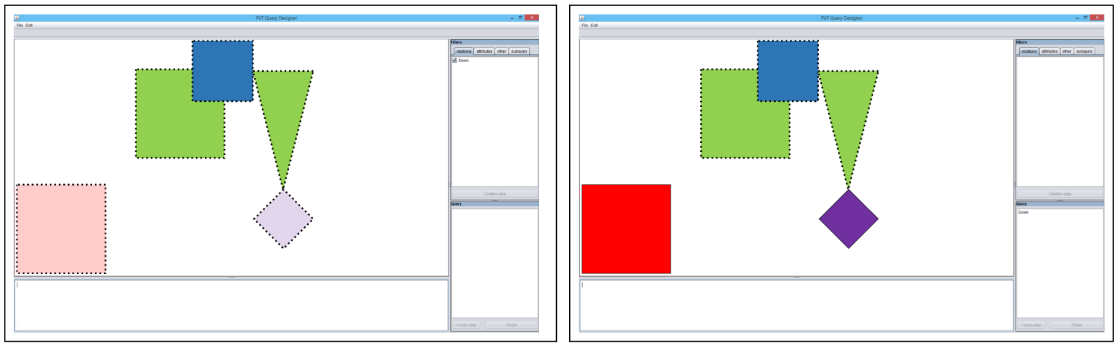


Figure 8. The application main interface.



(a) The application interface showing the relations from a selected object

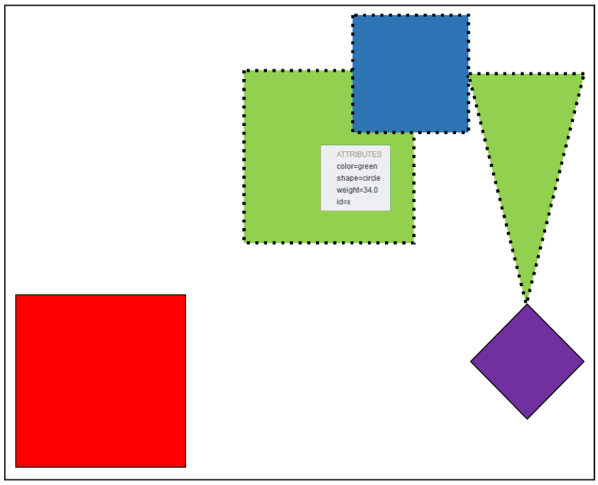
(b) Preview of the selected relation



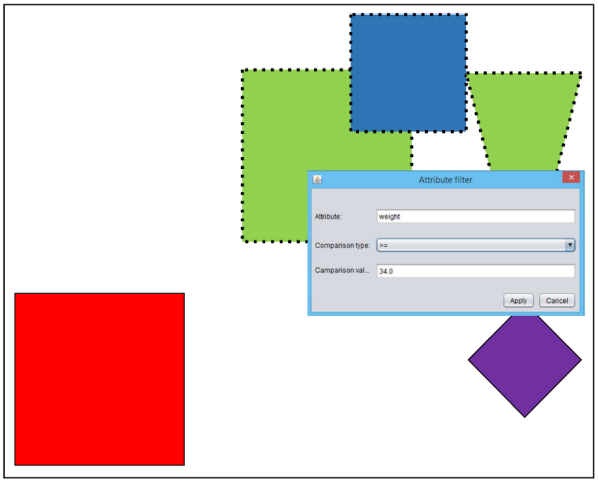
(c) The application interface showing the confirmed relation

(d) The application interface showing the objects returned by the relation step

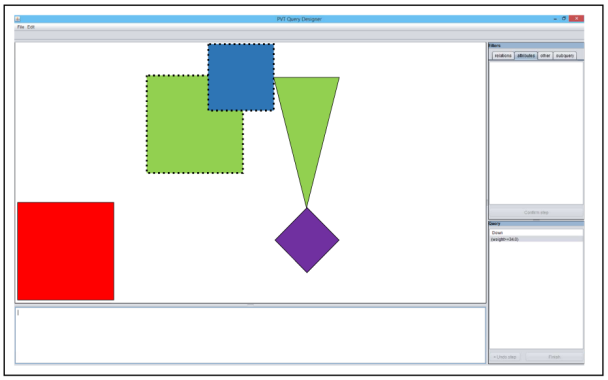
Figure 9. Building a relation step.



(a) The application interface showing the attributes of a selected object

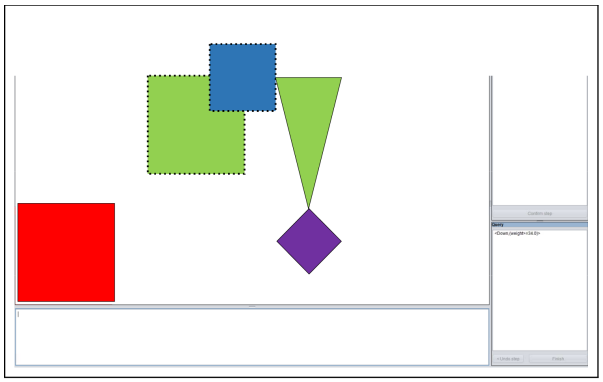


(b) The attribute step definition dialog

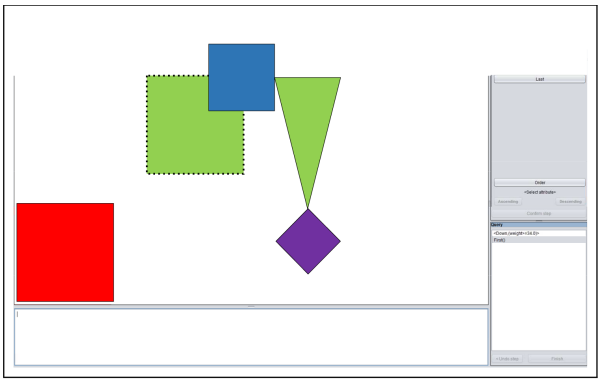


(c) The application interface showing the objects returned by both the relation and attribute steps

Figure 10. Building an attribute step.



(a) The application interface showing the final query and its results



(b) A refinement of the query in Figure 11(a) , with the addition of an extraction step

Figure 11. Completing the query definition.

application graphical interface. In particular, the user builds a query which selects the graphical objects having some other object below them (**Figure 9**) and a weight attribute greater than or equal to 34 (**Figure 10**). Then, this query is stored in the library (**Figure 11(a)**) and reused to build a second query which adds an extraction step (**Figure 11(b)**) in order to select only the first of such objects, in extraction order.

```
<Query name="down-and-weight">
 <Step>
  <Relation>Down</Relation>
 </Step>
 <Step>
  <Filter>
   <Attribute name="weight" type="number" op="GE">34</Attribute>
  </Filter>
 </Step>
</Query>
<Query name="first-down-weight">
 <Step>
  <Query>
   <Step>
    <Relation>Down</Relation>
   </Step>
   <Step>
    <Filter>
     <Attribute name="weight" type="number" op="GE">34</Attribute>
    </Filter>
   </Step>
  </Query>
 </Step>
 <Step>
  <Extract>First()</Extract>
 </Step>
</Query>
```

Figure 12. Query library in XML format.

6. Conclusions

In this paper we have developed a general query theory which has been integrated in the VIE technique, allowing to formalise how graphical objects can be extracted from visual documents.

This theory has been used to define an incremental algorithm implemented in a software application that allows to formulate queries through a user-friendly graphical interface, directly interacting with the visual document.

Our main further research aims to integrate this application within the SRQ tool, providing it with an assisted visual editor to perform VIE without the need to write complex SRQL statements. To this aim, the library of queries created by the application can be exported (see the XML fragment in **Figure 12** as an example) in a format compatible with the SRQ internal query representation.

Once accomplished, this integration would allow us to experiment our assisted query mechanism also on the different, meaningful domains which are currently supported by the SRQ tool, like web pages or geospatial data.

References

[1] Moens, M.-F. (2006) Information Extraction: Algorithms and Prospects in a Retrieval Context. *The Information Retrieval Series*, **21**, 1-45.

[2] Della Penna, G., Magazzeni, D. and Orefice, S. (2010) Visual Extraction of Information from Web Pages. *Journal of Visual Languages and Computing*, **21**, 23-32. http://dx.doi.org/10.1016/j.jvlc.2009.06.001

[3] Della Penna, G., Magazzeni, D. and Orefice, S. (2012) A Spatial Relation-Based Framework to Perform Visual Information Extraction. *Knowledge and Information Systems*, **30**, 667-692.

[4] Costagliola, G., De Lucia, A., Orefice, S. and Polese, G. (2002) A Classification Framework to Support the Design of Visual Languages. *Journal of Visual Languages and Computing*, **13**, 573-600. http://dx.doi.org/10.1006/jvlc.2002.0234

[5] Della Penna, G., Magazzeni, D. and Orefice, S. (2013) A General Theory of Spatial Relations to Support a Graphical Tool for Visual Information Extraction. *Journal of Visual Languages and Computing*, **24**, 71-87. http://dx.doi.org/10.1016/j.jvlc.2012.11.002

[6] Laender, A.H.F., Ribeiro-Neto, B.A., da Silva, A.S. and Teixeira, J.S. (2002) A Brief Survey of Web Data Extraction Tools. *ACM SIGMOD Record*, **31**, 84-93. http://dx.doi.org/10.1145/565117.565137

[7] Lam, M. and Gong, Z. (2005) Web Information Extraction. *Proceedings of the IEEE International Conference on Information Acquisition*, Hong Kong and Macau, 27 June-3 July 2005, 596-601..

[8] Gottlob, G., Koch, C., Baumgartner, R., Herzog, M. and Flesca, S. (2004) The LIXTO Data Extraction Project—Back and Forth between Theory and Practice. *Proceedings of the Symposium on Principles of Database Systems* (*PODS*-04), Paris, 14 June 2004, 1-12.

[9] Fazzinga, B., Flesca, S. and Tagarelli, A. (2011) Schema-Based Web Wrapping. *Knowledge and Information Systems*, **26**, 127-173. http://dx.doi.org/10.1007/s10115-009-0275-2

[10] Zhai, Y. and Liu, B. (2005) Web Data Extraction Based on Partial Tree Alignment. *WWW'*05: *Proceedings of the* 14*th International Conference on World Wide Web*, Chiba, 10-14 May 2005, 76-85. http://dx.doi.org/10.1145/1060745.1060761

[11] Soderland, S. (1999) Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, **34**, 233-272. http://dx.doi.org/10.1023/A:1007562322031

[12] Mittendorfer, M. and Winiwarter, W. (2002) Exploiting Syntactic Analysis of Queries for Information Retrieval. *Data & Knowledge Engineering*, **42**, 315-325. http://dx.doi.org/10.1016/S0169-023X(02)00049-6

[13] Laender, A.H.F., Ribeiro-Neto, B. and da Silva, A.S. (2002) DEByE—Data Extraction by Example. *Data & Knowledge Engineering*, **40**, 121-154. http://dx.doi.org/10.1016/S0169-023X(01)00047-7

[14] Wong, T.-L. and Lam, W. (2010) Learning to Adapt Web Information Extraction Knowledge and Discovering New Attributes via a Bayesian Approach. *IEEE Transactions on Knowledge and Data Engineering*, **22**, 523-536. http://dx.doi.org/10.1109/TKDE.2009.111

[15] Snoussi, H., Magnin, L. and Nie, J. (2002) Towards an Ontology-Based Web Data Extraction. *BASeWEB Proceedings of the Fifteenth Canadian Conference on Artificial Intelligence AI* 2002, Alberta, 27-29 May 2002.

[16] Jimeno-Yepes, A., Llavori, R.B. and Rebholz-Schuhmann, D. (2010) Ontology Refinement for Improved Information Retrieval. *Information Processing & Management*, **46**, 426-435. http://dx.doi.org/10.1016/j.ipm.2009.05.008

[17] Crescenzi, V. and Mecca, G. (1998) Grammars Have Exceptions. *Information Systems*, **23**, 539-565. http://dx.doi.org/10.1016/S0306-4379(98)00028-3

[18] Hammer, J., McHugh, J. and Garcia-Molina, H. (1997) Semistructured Data: The TSIMMIS Experience. In: Manthey, R. and Wolfengagen, V., Eds., *Advances in Databases and Information Systems*, Springer, St. Petersburg, 1-13.

[19] Zhao, H., Meng, W., Wu, Z., Raghavan, V. and Yu, C. (2005) Fully Automatic Wrapper Generation for Search Engines. *Proceedings of the* 14*th International Conference on World Wide Web*, Chiba, 10-14 May 2005, 66-75. http://dx.doi.org/10.1145/1060745.1060760

[20] Gatterbauer, W., Bohunsky, P., Herzog, M., Krüpl, B. and Pollak, B. (2007) Towards Domain-Independent Information Extraction from Web Tables. *Proceedings of the* 16*th International Conference on World Wide Web*, Banff, 8-12 May 2007, 71-80. http://dx.doi.org/10.1145/1242572.1242583

[21] Jiang, L., Wang, J., An, N., Wang, S., Zhan, J. and Li, L. (2009) GRAPE: A Graph-Based Framework for Disambiguating People Appearances in Web Search. *Proceedings of IEEE International Conference on Data Mining*, Miami, 6-9 December 2009, 199-208. http://dx.doi.org/10.1109/icdm.2009.25

[22] Rosenfeld, B. and Feldman, R. (2008) Self-Supervised Relation Extraction from the Web. *Knowledge and Information Systems*, **17**, 17-33. http://dx.doi.org/10.1007/s10115-007-0110-6

[23] Gu, X., Chen, J., Ma, W. and Chen, G. (2002) Visual Based Content Understanding towards Web Adaptation. In: De Bra, P., Brusilovsky, P. and Conejo, R., Eds., *Adaptive Hypermedia and Adaptive Web-Based Systems*, Springer-Verlag, Berlin, 164-173. http://dx.doi.org/10.1007/3-540-47952-x_18

[24] Yang, Y. and Zhang, H. (2001) HTML Page Analysis Based on Visual Cues. *Proceedings of the* 6*th International Conference on Document Analysis and Recognition*, Seattle, 10-13 September 2001, 859-864. http://dx.doi.org/10.1109/ICDAR.2001.953909

[25] Aumann, Y., Feldman, R., Liberzon, Y., Rosenfeld, B. and Schler, J. (2006) Visual Information Extraction. *Knowledge and Information Systems*, **10**, 1-15. http://dx.doi.org/10.1007/s10115-006-0014-x

[26] Chenthamarakshan, V., Varadarajan, R., Deshpande, P.M., Krishnapuram, R. and Stolze, K. (2012) WYSIWYE: An Algebra for Expressing Spatial and Textual Rules for Information Extraction. In: Gao, H., Lim, L., Wang, W., Li, C. and Chen, L., Eds., *Web-Age Information Management*—13*th International Conference*, Springer, Berlin, 419-433. http://dx.doi.org/10.1007/978-3-642-32281-5_41

[27] Gao, J., Zhou, Y. and Barner, K.E. (2012) View: Visual Information Extraction Widget for Improving Chart Images Accessibility. 19*th IEEE International Conference on Image Processing*, Orlando, 30 September-3 October 2012, 2865-2868. http://dx.doi.org/10.1109/icip.2012.6467497

[28] Uzun, E., Agun, H.V. and Yerlikaya, T. (2013) A Hybrid Approach for Extracting Informative Content from Web Pages. *Information Processing & Management*, **49**, 928-944. http://dx.doi.org/10.1016/j.ipm.2013.02.005

[29] Feder, J. (1971) Plex Languages. *Information Sciences*, **3**, 225-241. http://dx.doi.org/10.1016/S0020-0255(71)80008-7