A SPATIAL QUERY LANGUAGE FOR PRESENTATION-ORIENTED DOCUMENTS

Ermelinda Oro
DEIS, University of Calabria, Via P. Bucci 41/C, 87036 Rende (CS), Italy

Francesco Riccetti, Massimo Ruffolo
ICAR-CNR, University of Calabria, Via P. Bucci 41/C, 87036 Rende (CS), Italy

Keywords: Information extraction, Web wrapping, PDF wrapping, Spatial reasoning, Grammars, Chart parsing.

Abstract: In last years the huge relevance of accessing and acquiring information made available by Web (HTML) pages and business (PDF) documents has grown much further. In this paper we present a textual query language, named ViQueL, whose main feature is to identify and extract relevant information from HTML and PDF documents on the base of their visual appearance by using easy-to-write queries. The proposed language is founded on spatial grammars, i.e. context free grammars extended by spatial constructs. Despite a considerable expressive power, combined complexity of ViQueL is in P-Time. Moreover, experiments show that ViQueL is reasonably efficient for real-life extraction tasks.

1 INTRODUCTION

In the literature is available a large body of work on formalisms and approaches aimed at manipulating contents of HTML and PDF documents. Existing approaches broadly fall in the following main areas: (i) visual languages aimed at manipulating visual information for extraction and transformation scopes (Kong et al., 2006); (ii) web information extraction approaches that allow for extracting relevant information by exploiting mainly the visual appearance of Web pages (Baumgartner et al., 2001), or their internal representation (Cafarella et al., 2008); (iii) PDF wrapping approaches that enable to acquire information from PDF documents (Hassan and Baumgartner, 2005); (iv) approaches aimed at manipulating contents of presentation-oriented documents (Adali et al., 2000; Lee et al., 1999). Nevertheless existing approaches suffer from the following main drawbacks: (i) they do not allow for querying the visual structure of both HTML and PDF documents in a user friendly way; (ii) they are frequently task oriented so each of them deal with a specific problem like record extraction, table extraction and so on; (iii) frequently their computational costs are unknown.

In this paper we present a query language, named ViQueL and based on our previous work (Oro et al., 2009), that allows for querying both Web and PDF documents in order to recognize a wide variety of *content structures* (e.g. repetitive records, tables, news, infoboxes, profiles, etc) by exploiting their spatial arrangement. The proposed language is founded on spatial grammars (SGs), i.e. context free grammars extended by spatial constructs coming from the rectangular cardinal relation formalism (Navarrete and Sciavicco, 2006). ViQueL exploits a *spatial document model* (SDM) in which each document is viewed as a two-dimensional Cartesian plan on which are placed rectangles called *content items* (CIs). A *query* on a document is constituted by a set of *spatial production rules* (SPR) of the SG. Basically SPRs describe how to spatially compose CIs on the plane in order to identify meaningful content structures. Document querying is obtained by parsing the SDM using SPRs in the query. The parsing strategy is an ad hoc extension of the CYK parsing algorithm. It is noteworthy that ViQueL queries can be used for recognizing a given content structure in different documents having also different internal encodings. In fact, the SDM constitutes an abstraction of underlying internal representation. Experiments on real cases show how the ViQueL language is pretty much simple and how it allows users for querying presentation oriented documents on the base of what they see. Experiments

Oro E., Riccetti F. and Ruffolo M..
A SPATIAL QUERY LANGUAGE FOR PRESENTATION-ORIENTED DOCUMENTS.
DOI: 10.5220/0003177603060312
In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence* (ICAART-2011), pages 306-312
ISBN: 978-989-8425-40-9

show also that the adopted parsing strategy make the extraction process reasonably efficient.

The paper is organized as follows. Section 2 presents the spatial document model. In Section 3 syntaxt and semantics of ViQueL is presented by using some running examples. Section 4 formally presents syntax and semantics of the ViQueL language, describes the spatial CYK algorithm and shows languages complexity issues. Section 5 is about results of experiments. Finally Section 6 concludes the paper and sketches future work.

2 THE SPATIAL DOCUMENT MODEL

In this paper we subsume documents encoded by HTML or PDF formats as *presentation-oriented documents* (PODs). In the rest of the paper we will assume PODs as represented by the *spatial document model* (SDM). Broadly speaking the main idea which this model is based on is that the area of the screen aimed at visualizing a POD can be viewed as a 2-dimensional Cartesian plane on which is arranged a set of *content items* (CIs). A CI is an atomic piece of content (i.e. an images, an alphanumeric string written with unique font features, a graphical or typographical element like: a line, a circle etc.) visualized in a rectangular area of the plane. So the spatial document model of a POD consists in a set of CIs which are formally defined as follows.

Definition 1. *Let T be a set of type names arranged in a taxonomy, a content item is a 6-tuple of the form:*

$$CI = \langle \tau, \sigma, r_x^-, r_y^-, r_x^+, r_y^+ \rangle$$

where:

- $\tau \in T$ *is the type of the content item (e.g. image, text, number, percentage, currency, etc).*
- σ *is the value of the content item (e.g. a string, the URL of an image, etc).*
- *The pairs (r_x^-, r_y^-) and (r_x^+, r_y^+) represent two points on the Cartesian plane that identify the bottom-left, and top-right vertices respectively, of the rectangle that surround the contents σ (see Figure 1).*

Figures 1 and 3 show some CIs computed as described in Section 5.

3 QUERYING PODS BY VIQUEL

The scope of this section is to give a by example and intuitive explanation of ViQueL capabilities and features. The language is more rigorously presented and discussed in following Sections.

Example 1. *The example in Figure 1 shows a characteristic Deep Web page in which are contained a set of repetitive records. In Figure 2 is highlighted one of the record in Figure 1. The spatial arrangement of CIs, and the type of their contents, help human readers to immediately understand the meaning of the record. For identifying and extracting all records in that page, the user can pose the following ViQueL query.*

1. `S` $\xrightarrow{W}$ `img A`
2. `A` $\xrightarrow{E}$ `B C`
3. `B` $\xrightarrow{W}$ `text B`
4. `B` $\xrightarrow{W}$ `text text`
5. `C` $\xrightarrow{N}$ `text text`

The query is constituted by spatial production rules of the spatial grammar (see Section 4.2). The SPR 5 express that a new composed CI (identified by the non-terminal symbol `C`*) can be derived when two CIs of type* `text` *are in the spatial relation north ($\xrightarrow{N}$). In Figure 2 such a production compose the name of the product (first terminal* `text`*) and its description (second terminal* `text`*). The name of the product, in fact, is on top (north) of the description. SPRs 3 and 4 allow for recursively identifying sequence of CIs. So the non terminal* `B` *identifies a sequence of CIs in which each CI is at west ($\xrightarrow{W}$) of the others (the behavior of the rules is shown in Figure 2 by using the non-terminal symbol* `B` *two times). The SPR 2 allows for obtaining a new composed CI (represented by the nonterminal* `A`*) by combining non terminals* `B` *and* `C` *along the east ($\xrightarrow{E}$) direction. Finally, the SPR 1 completes the records (axiom S) including the picture (terminal* `img`*) which is located at west ($\xrightarrow{W}$) of A.*

Figure 1: The Ebay Web Page with some Highlighted CIs.

Example 2. *Figure 3 depicts information in tabular form contained in a PDF document. The query for this example is shown in the following.*

1. `S` $\xrightarrow{N}$ `HEADER BODY`

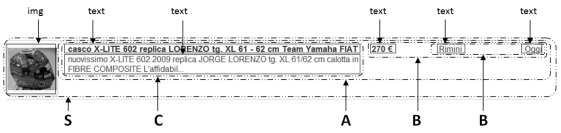

Figure 2: A Record with Associated CIs.

2. HEADER $\xrightarrow{E}$ HEADER text
3. HEADER $\xrightarrow{E}$ text text
4. BODY $\xrightarrow{S}$ ROW BODY
5. BODY $\xrightarrow{S}$ ROW ROW
6. BODY $\rightarrow$ ROW
7. ROW $\xrightarrow{W}$ text DATA
8. DATA $\xrightarrow{E}$ percent NUMBERS
9. NUMBERS $\xrightarrow{W}$ NUMBERS number
10. NUMBERS $\xrightarrow{W}$ number number

The following set of CI types are used: `text` *that represents plain text,* `number` *that represents integer and floating point numbers and is* `is_a` *of* `text`*,* `percent` *that represents percentage and is a sub type of* `number`*.*

Direct Energy Content [TJ]	1994	1996	1998	2000	2004	2005	2006	2007	Change '94 - '07
Total Gross Electricity Production	**144 708**	**192 879**	**147 998**	**129 776**	**145 583**	**130 468**	**164 199**	**140 964**	-2.6%
Oil	9 547	20 808	17 906	15 964	5 881	4 933	5 811	4 616	-51.7%
- Orimulsion	-	14 495	12 890	13 467	7	-	-	-	-
Natural Gas	8 206	20 442	29 260	31 589	35 807	31 606	33 903	24 886	203%
Coal	119 844	142 795	85 151	60 022	67 232	55 665	88 439	71 631	-40.2%
Surplus Heat	-	123	136	139	40	-	-	-	-
Waste, non-renewable	463	610	702	994	1 163	1 459	1 472	1 416	206%
Renewable Energy	**6 647**	**8 101**	**14 844**	**21 068**	**35 459**	**36 805**	**34 574**	**38 415**	**478%**
Solar Energy	0	1	1	4	7	8	8	9	3 017%
Wind Power	4 093	4 417	10 152	15 268	23 699	23 810	21 989	25 823	531%
Hydro Power	117	69	98	109	95	81	84	101	-14.1%
Biomass	2 116	3 207	3 911	4 936	10 646	11 889	11 517	11 504	444%
- Straw	293	748	960	654	3 057	3 088	3 359	3 185	988%
- Wood	429	340	512	828	3 546	3 730	3 041	3 398	691%
- Waste, renewable	1 393	2 120	2 439	3 454	4 043	5 071	5 117	4 921	253%
Biogas	321	407	682	751	1 013	1 017	976	978	205%

Figure 3: Table in a PDF Document with Highlighted CIs.

SPRs 9 and 10 (that use nonterminal symbol `NUMBERS`*) constitute a recursion that allow for capturing composed CIs constituted by sequence of numbers (terminal symbol* `number`*) along the west direction ($\xrightarrow{W}$). SPR 8 allows for creating composed CIs (identified by nonterminal* `DATA`*) constituted by the composition, along the east direction ($\xrightarrow{E}$), of a percentage (terminal symbol* `percent`*) and a composed CI identified as a* `NUMBERS`*. A row of the table (nonterminal* `ROW`*) is obtained by composing, along the west direction, a* `text` *CI and a composed CI* `DATA`*. SPRs 4, 5 and 6 use non-terminal* `BODY` *to identify the body of the table as a sequence of rows. A* `BOBY` *is a composed CI that can correspond to a single* `ROW` *(SPR 6 that acts as an* `isa` *relation) or to a recursive sequence of* `ROW`*s in the south direction ($\xrightarrow{S}$). SPRs 2 and 3, based on nonterminal* `HEADER`*, identify the header of the table as a sequence of* `text`*-type CIs. Finally, rule 1 combines header and body in order to identify the whole table in the axiom S. It is noteworthy that tables can be identified and extracted from Web documents in the same way. For lack of space we don't show here a related example.*

4 VIQUEL

In this section we give the formal definition of ViQueL. We first introduce the preliminary concept of Rect used for defining the language, then present language syntax, semantics and complexity issues.

4.1 Rectangular Cardinal Relations

In the Rectangular Cardinal Relations model (RCR) proposed in (Navarrete and Sciavicco, 2006), let r_1 and r_2 be two rectangles, spatial relations are expressed by analyzing the 9 regions (*cardinal tiles*) formed by the projections of the sides of a *reference rectangle* along the axes of the Cartesian plane. By considering cardinal tiles, *atomic* RCRs are traditionally expressed by means of the 9 symbols contained in the following set $R^A_{card} = \{B, S, SW, W, NW, N, NE, E, SE\}$. The symbol B denotes the *central tile* and the relation *belongs to*. Other symbols in R^A_{card} correspond to the *peripheral tiles* and denote the eight RCRs *South*, *SouthWest*, *West*, *NorthWest*, *North*, *NorthEast*, *East*, and *SouthEast*. Given $\rho \in R^A_{card}$, the expression $r_1\ \rho\ r_2$ means that r_2 lies entirely on the tile ρ of r_1. When a rectangle r_2 overlaps with several tiles relative to a reference rectangle r_1, we call the relation that links r_2 to r_1 *multi-tile relation* and represent it by $\rho_1 : \cdots : \rho_k$ where $\rho_1, ..., \rho_k$ are atomic RCRs. Atomic and multi-tile relations are called *Basic* RCR. For instance, by considering the RCR r_1 `E:NE` r_2 between a reference rectangle r_1 and another rectangle r_2, the RCR relation `E:NE` means that the rectangle r_2 lies on the tiles that are on east and north-east of the rectangle r_1. In the RCR model, the set R_{card} of all basic RCRs between any couple of MBRs contains 36 elements (Navarrete and Sciavicco, 2006).

4.2 Language Syntax and Semantics

In the following we introduce the concept of spatial grammar, give language semantics in terms of the parsing technique adopted for computing ViQueL queries, and present complexity results.

Definition 2. *A spatial grammar is a 5-tuple of the form:*

$$SG = (\Sigma, N, S, R_{rec}, \Pi)$$

where: (i) Σ is a set of terminal symbols that corresponds to CI types and identifies all the CIs in a SDM. (ii) N is a set of nonterminal symbols that identifies composed content items which structure is in Definition 3 below. (iii) $S \in N$ is the grammar axiom. (iv) R_{rec} is the set of basic cardinal direction relation introduced in Section 4.1. (iv) Π is a finite set of spatial production rules (SPRs) of the form $A \xrightarrow{dir} \alpha\beta$ and $A \rightarrow \alpha$, where: $A \in N$, $\alpha, \beta \in (\Sigma \cup N)$, and $dir \in R_{rec}$. Sets Σ, N and R_{rec}, are disjoint and finite.

In order to explain how spatial grammars works we have to define the concept of composed content item (CCI) as follows.

Definition 3. *A composed content item (CCI) is a 6-tuple of the form:*

$$CCI = \langle \Gamma, A, r_x^-, r_y^-, r_x^+, r_y^+ \rangle$$

where:

- Γ *is a non empty set of spatially contiguous CIs.*
- A *is the identifier of the CCI corresponding to a nonterminal symbol of the SG.*
- $r_x^- = \min(r_{x_\gamma}^- | \gamma \in \Gamma \wedge \gamma = (\tau, \sigma, r_{x_\gamma}^-, r_{y_\gamma}^-, r_{x_\gamma}^+, r_{y_\gamma}^+))$
- $r_y^- = \min(r_{y_\gamma}^- | \gamma \in \Gamma \wedge \gamma = (\tau, \sigma, r_{x_\gamma}^-, r_{y_\gamma}^-, r_{x_\gamma}^+, r_{y_\gamma}^+))$
- $r_x^+ = \max(r_{x_\gamma}^+ | \gamma \in \Gamma \wedge \gamma = (\tau, \sigma, r_{x_\gamma}^-, r_{y_\gamma}^-, r_{x_\gamma}^+, r_{y_\gamma}^+))$
- $r_y^+ = \max(r_{y_\gamma}^+ | \gamma \in \Gamma \wedge \gamma = (\tau, \sigma, r_{x_\gamma}^-, r_{y_\gamma}^-, r_{x_\gamma}^+, r_{y_\gamma}^+))$

CIs in Γ are spatially contiguous when the area defined by $r_{x_\gamma}^-, r_{y_\gamma}^-, r_{x_\gamma}^+, r_{y_\gamma}^+$ do not overlaps with other CIs not in Γ.

A SPR of the form $A \rightarrow \alpha$, where $\alpha \in \Sigma$, and $A \in N$ allows for creating a CCI corresponding to a CI identified by the terminal symbol α. This operation constitutes a generalization of the terminal α into the nonterminal A, and at the same time a transformation of the CI related to α into the CCI related to A. Rules of the form $A \xrightarrow{dir} \alpha\ \beta$, where $\alpha, \beta \in \Sigma$, and $A \in N$, compose the two contiguous CIs related to the terminal symbols α and β, along the direction specified by the relation $\xrightarrow{dir}$, in order to obtain a new CCI A having the structure described in Definition 3. SPRs having the form $A \xrightarrow{dir} BC$, where $A, B, C \in N$, compose the sets Γ_B and Γ_C of CIs in the CCIs related to the nonterminals B and C respectively, in order to obtain a new CCI having coordinates computed as specified in Definition 3. Similar considerations can be done for rules that combine terminal and nonterminal symbols.

4.2.1 The Spatial CYK Algorithm

In the following we present the pseudo-code of the SCYK algorithm. Computational complexity aspects are discussed in Section 4.2.2. Before presenting the algorithm we define a CNF-like normal form that allows a shortest and more intuitive pseudo-code. It is obviously possible and easy to extend the algorithm to parse any type of rules.

Definition 4. *A SG is in SG-normal form if all its production rules are either in the form $A \xrightarrow{dir} BC$, or $A \rightarrow \alpha$ where A,B and C are non-terminals, while α is a terminal symbol. Production of type $A \rightarrow \alpha$ are called unary spatial production rules.*

The algorithm takes as input a SG Q and a set of CIs D. In instruction 1 the algorithm creates two ordered sets L_x and L_y that contain coordinates r_x^- and r_y^- of all CIs $\in D$ respectively.

In the worst case $n = |L_x|$ and $m = |L_y|$ can be at most equal to the size of the document $|D|$, but in real cases both have a size smaller than $|D|$. In instruction 4 SPRs in Q are acquired in the set RS. Instruction 6 assigns to RS_U all unary SPRs in RS. In instruction 7 non unary SPRs are split in two subsets RS_H and RS_V that contain rules of the type $V \xrightarrow{dir} XY$, where dir is a RCR that expresses relations obtained from the subsets of basic RCR $\{E, SE, NE, W, SW, NW, B\}$ for RS_H and $\{N, NW, NE, S, SE, SW, B\}$ for RS_V.

Instructions 8 and 9 generate tables T_1 and T_2 respectively. Indices in the first four positions of T_1 and T_2, namely i_2, i_1, j_2, and j_1 identify the CCI having as bottom-left vertex $(L_x[i_2], L_y[j_2])$ and as top-rigth vertex $(L_x[i_2 + i_1], L_y[j_2 + j_1])$. The last position in table T_1 contains a nonterminal symbol.

The general idea which guides the algorithm is that elements in T_2 represent CCIs by the corresponding coordinates, while elements in T_1 are boolean values that state if a given nonterminal symbol V in the grammar is associated to the corresponding CCI in T_2 (it is noteworthy that a CCI can have different associated nonterminal symbols). Instruction 10 creates a two-dimensional table I where elements $I[i_1, j_1]$ contain a set of couples $< i_2, j_2 >$ which indicate that the CCI in $T_2[i_2, i_1, j_2, j_1]$ is not null. Instruction 11 creates the table *res* that represents the result of the algorithm (i.e. the set of CCIs that satisfies the axiom S in the grammar Q). Instruction 12 initializes tables T_1, T_2, and I by using the set D of input CIs and the set RS_U of unary SPRs. The initialization procedure works as follows: if an area having as vertices $(L_x[i_2], L_y[j_2])$ and $(L_x[i_2 + i_1], L_y[j_2 + j_1])$ contains only one CI, the couple $< i_2, j_2 >$ is added to $I[i_1, j_1]$ and $T_2[i_2, i_1, j_2, j_1]$ is filled by entering coordinates of that CI. Let α be a CI type (i.e. a terminal symbol), then for each unary rule $V \rightarrow \beta$, where β *isa* α (*isa* is computed by using the taxonomy of CI types), element $T_1[i_2, i_1, j_2, j_1, V]$ is set to *true* (such an operation corresponds to the generation of a CCI for each initial CI). Remaining values in tables T_1, T_2 and I are computed in the main loop (instructions 13-

Algorithm 1: Spatial CYK.

```
   Input  : A SG Q and a set D of CIs (i.e. the document)
   Output: A set of CCIs that satisfy the grammar Q
1  < L_x, L_y > = createOrderedCoordinateSets(Q);
2  n = |L_x|;
3  m = |L_y|;
4  RS = getRuleSet(Q);
5  R = getNonTerminalNumber(RS);
6  RS_U = getRuleSet(RS);
7  < RS_H, RS_V > = splitRuleSetByDirection(RS − RS_U);
8  createTable T_1[n,n,m,m,R];
9  createTable T_2[n,n,m,m];
10 createTable I[n,m];
11 createSet res;
12 < T_1, T_2, I > =initialize(D, RS_U);
13 for i_1 ← 1 to n do
14     for j_1 ← 1 to m do
15         for j_3 ← 1 to j_1 − 1 do
16             indexSet = I[i_1, j_3];
17             foreach < i_2, j_2 > ∈ indexSet do
18                 if j_2 + j_1 ≤ m then
19                     for i_3 ← i_1 downto 1 do
20                         ver(i_2, i_1, j_2, j_1, i_1, j_3,
                           i_2, i_3, j_2 + j_3, j_1 − j_3,
                           T_1, T_2, RS_V, res);
21                     end
22                 end
23             end
24         end
25         for i_3 ← 1 to j_1 − 1 do
26             indexSet = I[i_3, j_1];
27             foreach < i_2, j_2 > ∈ indexSet do
28                 if i_2 + i_1 ≤ n then
29                     for j_3 ← j_1 downto 1 do
30                         ver(i_2, i_1, j_2, j_1, i_3, j_1,
                           i_2 + i_3, i_1 − i_3, j_2, j_3,
                           T_1, T_2, RS_H, res);
31                     end
32                 end
33             end
34         end
35     end
36 end
37 return < res, T_1 >;
```

36). CCIs of increasing sizes are discovered by iterating with the two most external cycle by using indexes i_1 (to increase length) and j_1 (to increase height) in instructions 13 and 14 respectively. Two different internal loops are then used in order to find larger CCIs in horizontal and vertical directions respectively, by merging contiguous CCIs. In instruction 15 each iteration uses pairs $< i_2, j_2 >$ in the entry $I[i_1, j_3]$ of table I (instruction 16) in order to consider each existing CCI cci_1 having width equals to $|L_x[i_2 + i_1] - L_x[i_2]|$ and height equals to $|L_y[j_2 + j_3] - L_y[j_3]|$. The inner

Procedure ver(i_2,i_1, j_2, j_1,$i_1{}'$, $j_1{}'$,$i_2{}''$, $i_1{}''$, $j_2{}''$, $j_1{}''$,T_1, T_2,RS, res).

```
1  p_1 = Q[i_2, i_1', j_2, j_1'];
2  p_2 = Q[i_2'', i_1'', j_2'', j_1''];
3  if p_2 ≠ null then
4      foreach V -dir-> XY ∈ RS_V do
5          c_1 = T_1[i_2, i_1', j_2, j_1', X] ∧ T_1[i_2'', i_1'', j_2'', j_1'', Y] ∧
           (p_1 dir p_2);
6          c_2 = T_1[i_2, i_1', j_2, j_1', Y] ∧ T_1[i_2'', i_1'', j_2'', j_1'', X] ∧
           (p_2 dir p_1);
7          if c_1 ∨ c_2 then
8              T_1[i_2, i_1, j_2, j_1, V] ← true;
9              update(T_2[i_2, i_1, j_2, j_1]);
10             if V is axiom then add(res, T_2[i_2, i_1, j_2, j_1]);
11         end
12     end
13     break;
14 end
```

cycle (Instruction 19 and procedure *verify* in instruction 20) takes (if exists) the widest CCI cci_2 contiguous to cci_1. It is necessary to iterate from i_1 to 1 with index i_3 because we have to consider also inclusion relations (B) between two CCIs. If we don't consider inclusion relation we can take as contiguous CCI cci_2 the one having as corners $(L_x[i_2], L_y[j_2 + j3])$ and $(L_x[i_2 + i_3], L_y[(j_2 + j3), (j_1 - j3)])$. If such CCI exists the algorithm examines each rule $V \xrightarrow{dir} XY \in RS_V$. If values of the two elements in T_1 referring to (cci_1, X) and (cci_2, Y) are *true* and cci_1 is at dir to cci_2 then $T_1[i_2, i_1, j_2, j_1, V]$ is set to *true* and value of $T_2[i_2, i_1, j_2, j_1]$ is set to the proper dimension of the CCI. If V is the start symbol in Q, the CCI just found is added to the result set *res*. Specular operations are done in the second sub-loop in order to evaluate SPRs in RS_H. At the end the algorithm returns the set *res* containing founded CCIs and the table T_1 that can be used for generating the query (grammar) parse tree by a trace-back procedure.

4.2.2 Complexity Issues

Theorem 1. *Let D be the SDM of a presentation-oriented document, the evaluation of a ViQueL query Q, performed by Algorithm 1, requires space* $O(|D|^4 \cdot |Q|)$ *and time* $O(|D|^6 \cdot |Q|)$*, where* $|D|$ *and* $|Q|$ *are the size of the document in terms of CIs and the size of the query in terms of SPRs respectively.*

Proof. Space complexity. Memory usage of the algorithm corresponds to the size of table T_1. In table T_1, R represents the number of nonterminals in the query Q, so the space complexity is $O(m^2 \cdot n^2 \cdot R)$. Since the number of non terminals can be at most $|Q|$,

and in the worst case we have that $m = n = |D|$. The space complexity bound $O(|D|^4 \cdot |Q|)$ follows. ◊

Proof. Time complexity. Filling ordered sets L_x and L_y can be done in $O(|D| \cdot lg(|D|))$. Comparisons in the algorithm can be all done in constant time using appropriate data structures. Remember also that m and n are both bounded by $|D|$. Operations concerning splitting of SPRs in Q are done in linear time, i.e. $O(|Q|)$. Initialization procedure can be performed in $O(|D|^2 \cdot |Q|)$. Main loop is clearly the part of the algorithm that takes more time. Procedure $verify$ is manifestly in $O(|Q|)$. The maximum number of couple $< i_2, j_2 >$ that can be contained in a element $I[i_1, j_1]$ of table I is $O(m \cdot n)$, in the worst case scenario we have $O(|D|^2))$ because $m = n = |D|$. So time complexity in the worst case is computed as $O(n \cdot m \cdot (m^2 \cdot n \cdot (n \cdot |Q|) + n^2 \cdot m \cdot (m \cdot |Q|))) = O(n^3 \cdot m^3 \cdot |Q|)$, i.e. $O(|D|^6 \cdot |Q|)$. ◊

5 EXPERIMENTS

In Figure 4 are shown results of experiments carried out for empirically verify Theorem 1. We ran experiments on a Windows 7 machine with 2,53 GHz Intel core-duo processor and 4 GB of RAM. We considered the table and the query in Example 2 and linearly increased their sizes by adding table rows and dummy rules respectively. Figures 4a and 4b show required space and time for document sizes that grown from $|D| = 25$ to $|D| = 1020$. Figures 4c and 4d show required space and time considering query sizes form $|Q| = 10$ to $|Q| = 108$. Curves in the figures refer to normalized values and has been drown in loglog scale. Experiments show that required space in the average case is linear w.r.t. both the size of Q and D, while required time, in the average case, is linear w.r.t. the size of Q and polynomial (with a degree smaller that in the worst case) w.r.t. the size of D. In Example 2 the system takes about 200 milliseconds for recognizing the table, so the implemented system results effectively usable in real cases. We have, also, performed usability experiments by asking 10 user to learn the language and apply it for extracting tables and data records from a dataset composed of 5 PDF documents and 5 web pages. Experiments have shown that the language is easy to learn and that can by intuitively applied for real extraction tasks. We don't give further details about usability experiments for lack of space.

6 CONCLUSIONS AND FUTURE WORK

In this paper we presented ViQueL, a spatial query language that allow for querying presentation-oriented documents on the base of their visual appearance. The language is founded on spatial grammars that are obtained by combining classical context free grammars and qualitative spatial reasoning constructs. The main feature of ViQueL is that it allows for easily defining visual queries that enable to recognize complex content structures in both HTML and PDF documents. ViQueL queries are computed by the SCYK algorithm, a spatial extension of the well known CYK algorithm. Despite the increased expressiveness of spatial grammars, the complexity of the SCYK algorithm remains in P-Time. Furthermore, experiments prove that the algorithm is efficient and usable in real-life cases with satisfactory results. The proposed approach can be improved by adopting a stochastic extension to SGs in order to better manage ambiguous queries. As future work we intend to apply inductive approaches for learning ViQueL from portions of documents visually selected by users. This way no manual code writing will be required to the users.

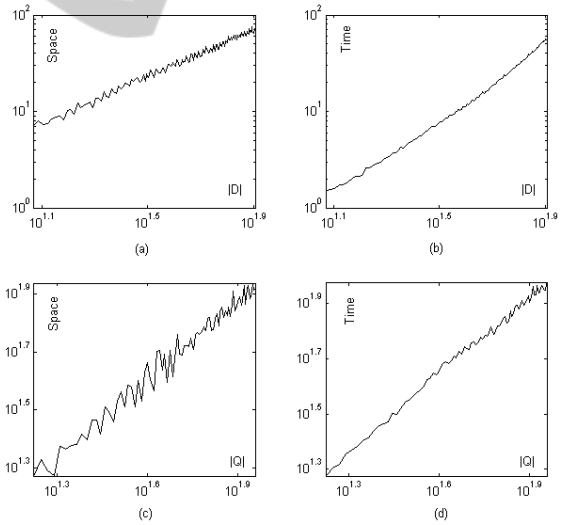

Figure 4: Results of Experiments.

REFERENCES

Adali, S., Sapino, M. L., and Subrahmanian, V. S. (2000). An algebra for creating and querying multimedia presentations. *Multimedia Syst.*, 8(3):212–230.

Baumgartner, R., Flesca, S., and Gottlob, G. (2001). Visual web information extraction with lixto. In *VLDB*

'01: Proceedings of the 27th International Conference on Very Large Data Bases, pages 119–128, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Cafarella, M. J., Halevy, A., Wang, D. Z., Wu, E., and Zhang, Y. (2008). Webtables: exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549.

Hassan, T. and Baumgartner, R. (2005). Intelligent text extraction from pdf documents. In *CIMCA/IAWTIC*, pages 2–6.

Kong, J., Zhang, K., and Zeng, X. (2006). Spatial graph grammars for graphical user interfaces. *ACM Trans. Comput.-Hum. Interact.*, 13(2):268–307.

Lee, T., Sheng, L., Bozkaya, T., Balkir, N. H., Özsoyoglu, Z. M., and Özsoyoglu, G. (1999). Querying multimedia presentations based on content. *IEEE Trans. on Knowl. and Data Eng.*, 11(3):361–385.

Navarrete, I. and Sciavicco, G. (2006). Spatial reasoning with rectangular cardinal direction relations. In *ECAI*, pages 1–9.