Document Information Extraction and its Evaluation based on Client's Relevance

Santosh K.C., Abdel Belaïd

Document Information Extraction and its Evaluation based on Client's Relevance

K.C. Santosh and Abdel Belaïd
LORIA – Université de Lorraine
BP 239 - Loria Campus Scientifique, 54506 Nancy Cedex, FRANCE
Email: {santosh.kc, abdel.belaid}@loria.fr

***Abstract*—In this paper, we present a model-based document information content extraction approach and perform in-depth evaluation based on clients' relevance. Real-world users i.e., clients first provide a set of key fields from the document image which they think are important. These are used to represent a graph where nodes (i.e., fields) are labelled with dynamic semantics including other features and edges are attributed with spatial relations. Such an attributed relational graph (ARG) is then used to mine similar graphs from a document image that are used to reinforce or update the initial graph iteratively each time we extract them, in order to produce a model. Models therefore, can be employed in the absence of clients. We have validated the concept and evaluated its scientific impact on real-world industrial problem, where table extraction is found to be the best suited application.**

I. Introduction

In document analysis, information content exploitation has been received an important attention. Extracting similar information in accordance with the clients, will be one of the key commercial applications in the domain. Within this framework, lets take a table extraction problem since it is assumed to have several similar items that basically span horizontally from left to right, regardless their key fields alignment as well as number of words or lines they are composed of.

In the context of table extraction [1]–[4], existing algorithms basically describe it either in terms of lines and (un)analysed text blocks, a set of cells resembling the two-dimensional grid or a set of strings that are integrated with each other via relations, for instance. Basically, table detection and its structure recognition are two major tasks. Table detection can be taken as a primary issue, which however does not provide a complete solution [5] since one needs to be able to extract key fields within it. Existing methods such as table segmentation [6] do not extract key fields, nor do they explicitly perform the content understanding [7] without considering relations between the contents, for instance. Note that structural information can be very useful in indexing and retrieving the information contained in the document [2]. To analyse table-forms structure, rulings techniques are essentially limited since since not all tables possess graphical lines [1]. Besides, plain ascii texts, text blocks are basically used where detecting columns, lines and headers, and representing them in terms of graph, for instance is interesting. In order to fully exploit table in the scanned documents rather than just outlining the overall boundary, it is interesting to extract those fields that are important or meaningful for the clients. To handle this, in this paper, key fields are provided by the clients. These key fields are then used to build a graph so that it can be applied for table extraction in the absence of clients. On the whole, key fields exploitation, which are not necessarily found in structured documents like table, but in semi-structured documents like form and in structure-free documents, is the interest of the paper.

The rest of the paper is organised as follows. We start with explaining the proposed method in Section II. It mainly includes graph initialisation, graph mining and graph learning. Full experiment evaluations are reported and analysed in Section III. The paper is concluded in Section IV.

II. Proposed method

Following Fig. 1, clients first provide key fields within the table which they think are important. An input pattern graph is now initialised from such a set of key fields where each key field is labelled and the possible relations are attributed. The pattern graph is then transformed into a model graph via graph mining. It simply starts with a pivotal node selection in a document (with respect to each labelled node in pattern graph). From each pivotal node, relation assignment will guide feature score computation between the pairs of nodes. Our relations constraint feature score computation is fast since the search space is limited to the degree of the node associated with the pattern graph. To avoid polynomial time computational complexity [8], we use semantic labels to confirm structural similarity via relations, between the graphs. The extracted similar graphs are now verified and used to reinforce or update the pattern graph as a model graph. Such a model is able to extract information content.

This paper is the thorough extension of the work presented in [9] where it basically provides a proof of a concept, validating on limited input patterns that are mostly linear and are more focussed on table extraction problem. Based on this, in this paper, we integrate dynamic labels at nodes instead of just not relying one a pre-defined list as well as introduce a possible swinging (i.e., back and forth, and up and down) capability of spatial relations in graph mining scheme. In addition, in-depth evaluations that are not just limited to extract information content associated with tables, are made.

A. Graph initialisation

Any document d is composed of a set $\mathcal{Z}$ of zones i.e., $\mathcal{Z} = \{\text{header, body, footer}\}$. In generic form, we have $\text{doc.}_d = \{\text{zone}_z, z \in [1, \mathbb{Z}]\}$. For each zone type z, the clients provide input pattern(s)

$$\text{zone}_z = \{\text{pattern}_p, p \in [1, \mathbb{P}]\}, \quad (1)$$

Fig. 1. Our concept in three consecutive phases: 1) graph initialisation, 2) graph mining and 3) graph learning.

where the number of input patterns can be arbitrary. An example of an input pattern is shown in Fig. 2. In this example, to make an explanation simple, we take body zone type where table-like structure is basically present. An input pattern is just a collection of selected key fields i.e.,

$$\text{pattern}_p = \{\text{field}_i\}_{i=1}^{\mathbb{A}}. \tag{2}$$

To represent a field, we define a feature set. $\mathcal{F} = \{\text{feature}_f\}$. In our case, for any i-th field, we can formally represent a feature as $\text{field}_i^{\mathcal{F}} = \{$

(`box`: [left, top, right, bottom]);	(`wSep`: words separation);
(`value`: content);	(`noW`: number of words);
(`type`: content type);	(`noL`: number of lines);
(`size`: string length);	(`label`: e.g., *date*, *price* ...}

(3)

Features of course, are application dependent. The labels in the feature set, are the derivatives of the table headers, representing semantic values. To update or enrich semantic labels in a set $\mathcal{L}$, a dynamic field analyser is proposed so that its corresponding regular expression format can be achieved. Thanks to the regular expressions, we are able to avoid possible OCR errors in case characters are broken, documents with noise and characters are connected with graphics. For example, consider a field having OCR output as (*cf.* Fig. 2)

```
<field id="1"
name="CODE-ARTICLE" value="0652-000159"
left="261" top="1311" right="528" bottom="1338"/>
```

can be transformed into a label named `code-article` with regular expression of `'\d{4}[-]\d{6}'`. Such labels are then used to update the set $\mathcal{L}$.

To exploit relative positioning between the key fields, we basically use bounding box and its projection into 3×3 partitions [10]: $\mathcal{R} = \{$left top, top, right top, ..., right bottom$\}$. To integrate more precision about the level of neighbourhood k into the basic set of spatial predicates defined in $\mathcal{R}$, we have

$$r_{ij} = \textit{spatial predicate}_{k_1,k_2}(\text{field}_i, \text{field}_j). \tag{4}$$

Formally, $k = 0$ for an adjacent (an immediate field), and k varies from 1 to $\mathbb{A}-1$ for non-adjacent ones. Note that k_1 and k_2 represent horizontal and vertical orientations, respectively.

The whole input pattern can be represented by a complete 4-tuple ARG

$$G = (V, E, F_V, F_E), \tag{5}$$

where V is a finite set of nodes (fields) and $E \subseteq V \times V$ i.e., a finite set of edges. Each $r_{ij} \in E$ is a pair of (v_i, v_j) where $v_i, v_j \in V$. $F_V : V \to L_V$, where L_V represents a set of nodes as well as their labels using the set $\mathcal{L}$ and $F_E : E \to R_E$, where R_E represents the edges via relations.

Code Article	Désignation Article	Quantité	Prix Unitaire	Prix Total H.T.	Code TVA
0652-000159	PORTEMANTEAU 10 CROCHETS ALU	1	28.05	28.05	02
0014-000486	ARMOIRE SUR SOCLE GRIS/BLEU 2 TABLETTES ET 2 TIROIRS	1	301.13	301.13	02

Fig. 2. An example of an input pattern (cropped image), provided by a client.

Fig. 3. Corresponding graph for the pattern provided in Fig. 2. In the graph representation, bigger and filled nodes are the selected fields while the remaining are missing ones.

To complete graph, we introduce missing as well as neighbouring fields such as `Quantity`, `Prix Total H.T.` and `Code TVA` in Fig. 2. To determine those fields, we simply use intra-field separation from the selected key fields:

$$\text{intra-field}_i = \max\{\text{intra-wSep}_i\}, \tag{6}$$

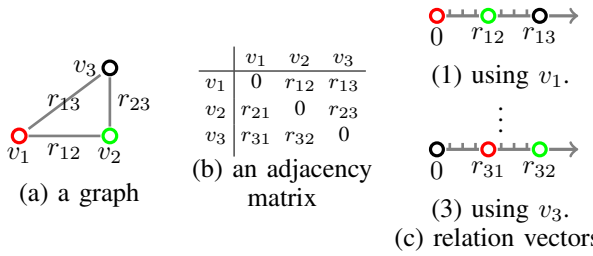
where intra-wSep$_i$ is the minimum distance between two consecutive words within the same field i.e., intra-wSep$_i$ = $\min\{\delta(\text{word}_w, \text{word}_{w+1}) : \text{word}_w, \text{word}_{w+1} \in \text{field}_i\}$. Multiple words are taken as a single field as long as their separation is defined in Eq. (6). To separate missing and neighbouring fields from the selected ones, we simply use activation key at nodes: 1 for selected fields and 0, otherwise. For example, in Fig. 2, we have $\{v_i\}_{i=1}^{6}$ where node activation signature is [1 1 0 1 0 0], spanning horizontally, from left to right.

B. Graph mining

Given the pattern graph Q, to extract similar graphs from a document, it starts with pivotal nodes selection in a document and perform relation assignment to compute feature score between the pairs of nodes. Relations assignment repeats until a similar graph G is achieved, with respect to Q.

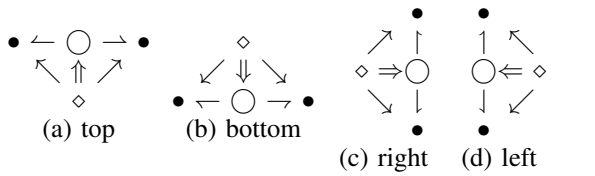
Step 1. For every node v_i^q in pattern graph Q, the corresponding label $\ell_i^q \in \mathcal{L}$ is defined i.e., $V^q = \{(v_i^q, \ell_i^q), i = 1 \ldots \mathbb{V}^q\}$. Having these labelled nodes in Q, the target is to select nodes sharing identical labels $\{v_i, \ell_i\}$ in a document.

Step 2. Each pivotal node is taken and started to validate relations with neighbouring nodes in a document, as in Q. As an example, Fig. 4 simplifies the relation assignment process by showing relation vector spaces from a graph. To compute feature score between the pair of nodes (v_i, v_j) in a document with respect to $(v_i^q, v_j^q) \in Q$, their respective relations must be identical i.e., $r_{ij}^q = r_{ij}$. If exact relations do not validate, our



(a) a graph (b) an adjacency matrix (1) using v_1. (3) using v_3. (c) relation vectors.

Fig. 4. An example showing relation vector spaces to simplify relation assignment. A single relation vector space is sufficient to extract structurally similar graph.



(a) top (b) bottom (c) right (d) left

◇: reference, ◯: exact relation, and •: immediate relation

Fig. 5. A few examples showing possible immediate relations. For an exact top relation, left top and right top can be two possible immediate relations.

algorithm introduces possible swinging to immediate relations that goes back and forth, and up and down. The idea is elaborated in Fig. 5, where relation can swing both left and right for top_{ij} and bottom_{ij} relations, and top and bottom for left_{ij} and right_{ij} relations. More formally, we can compute feature score between two corresponding nodes v^q and v as

$$f.\text{score}(v^q, v) = \begin{cases} 1 : \text{if } \ell^q = \ell, \text{ and} \\ \frac{1}{\mathbb{F}} \sum_f \lambda_f \times s^{\text{feature}_f}_{v^q, v} : \text{ otherwise,} \end{cases} \quad (7)$$

where $\lambda_f \in [0, 1]$. For each feature, weight λ_f can be varied according to its robustness and so is application dependent. Given two strings: x reference and y primary, we compute feature matching scores from string *type*, *noW* and *size*:

$$\begin{aligned} s^{type}_{x,y} &= 1 - (\text{Levenshtein dist.}(x, y)/\max(x, y)), \\ s^{noW}_{x,y} &= 1 - \left(\text{dist.}^{noW}(x, y)/\max(x, y)\right), \text{ and} \\ s^{size}_{x,y} &= 1 - \left(\text{dist.}^{size}(x, y)/\max(x, y)\right), \end{aligned} \quad (8)$$

where in case of s^{type}, we treat numerals $\{0-9\}$, all alphabets $\{A-Z, a-z\}$ and symbols equally. Thus, matching score S for data graph G with respect to Q is an aggregation of both relations validation and feature scores

$$\begin{aligned} S(Q, G) \quad = \quad & \alpha \tfrac{1}{\mathbb{R}} \textstyle\sum_{i,j \in \mathcal{R}^q, i \neq j} r.\text{score}(r^q_{ij}, r_{ij}) \\ & + (1 - \alpha) \tfrac{1}{\mathbb{V}^q} \textstyle\sum_{i \in V^q} f.\text{score}(v^q_i, v_i), \end{aligned} \quad (9)$$

where $\alpha \in [0, 1]$. Based on this, we extract a set $\mathcal{G}$ of similar graphs plus their matching scores i.e., $\mathcal{G} = \{G_g, S_g\}^{\mathbb{G}}_{g=1}$. To avoid possible redundant graphs in case every node is employed from the set $\{(v^q_i, \ell^q_i)\}$, we take $\arg\max_{1 < i \leq \mathbb{V}^q} S(Q, G_i)$. Since we employ word-level pivotal selection, labels like *description* are not taken. These are extracted with the help of neighbouring labels, their relations and features like *size*, *noW* and *noL*. Therefore, faster graph isomorphism like [11] may not directly fit into our application.

Code Article	Désignation Article	Quantité	Prix Unitaire	Prix Total H.T.	Code TVA
0652-000159	PORTEMANTEAU 10 CROCHETS ALU	1	28.05	28.05	02
0014-000486	ARMOIRE SUR SOCLE GRIS/BLEU 2 TABLETTES ET 2 TIROIRS	1	301.13	301.13	02

Fig. 6. An output, representing two similar items (i.e., two graphs) from an input pattern shown in Fig. 2.

C. Graph models

The mined graphs $\{(G_g, S_g)\}$ are now pruned by using two major criterion: graph consistency and matching score. The graph is said to be consistent if $\ell_i \in \mathcal{L}$. In case all nodes are not labelled, the graph is taken if $S(,) \geq$ threshold (empirically designed). Node features are updated by taking their *label*, and properties like *size*, *noW* and *noL*, including their variations. As an example, features at query nodes are updated from data graph. As an example, in Fig. 6, $\text{field}^{\mathcal{F}}_2 = \{$

(box: $[541, 1314, 1221, 1348]$); (value: '{'PORTEMANTEAU' '10' 'CROCHETS' 'ALU'}'), (size: '$[25, 44]$'); (wSep: '$[4, 6]$'); (noW: '$[4, 9]$'); (noL: $[1, 2]$); (label: '{*description*}')}.

Keeping relations as in pattern graph, the updated pattern graph will be a model graph $M(V, E, \hat{F}_V, F_E)$. Considering a set $\mathcal{Z}$ of zones, we have a set $\mathcal{M}$ of models in class $\mathbb{k}$ as,

$$\mathcal{M}_{\mathbb{k}} = \left\{ M^{\text{header}_p}_{\mathbb{k}}, M^{\text{body}_p}_{\mathbb{k}}, M^{\text{footer}_p}_{\mathbb{k}} \right\}, \ 1 \leq p \leq \mathbb{P}. \quad (10)$$

These models are used to exploit similar information from test documents (in the absence of clients). From each model, an output confidence score (CS) is computed from the aggregation of all matching scores of the extracted similar graphs i.e.,

$$\text{CS}^{z_p}_{\mathbb{k}} = \tfrac{1}{\mathbb{G}} \textstyle\sum_g S_g. \quad (11)$$

As an example, in Fig. 6, we have $\text{CS}^{\text{body}_p}_{\mathbb{k}} = \frac{1}{2} \sum^2_{g=1} S_g, = \frac{1}{2}(1 + 1) = 1$. In case of multiple models, the outputs are now ranked based on the order of similarity.

III. Experiments

A. Dataset and ground-truth formation

We work on a real-world industrial problem in direct collaboration with the **ITESOFT**[1], France. Currently, in our test, the dataset is composed of 25 suppliers, having a few hundreds of images per class. In our dataset, both printed and handwritten texts including graphics are present. Broken strokes in characters, noise due to touched handwritten texts – signatures, for instance and overlapping of graphics such as stamps are considerable issues. Tabular alignments are not always regular even when headers are found to be relatively well positioned. For each document, clients provide ground-truths i.e., all similar patterns, according to the selected pattern.

B. Evaluation metric

Consider the list of the extracted graphs, representing detected table or output $O = \{G_g\}^{\mathbb{G}}_{g=1}$ in a test document. For this, there are $\mathbb{G}^\circ$ list of ground-truthed patterns corresponding to the ground-truthed table $O^\circ = \{G^\circ_{g^\circ}\}^{\mathbb{G}^\circ}_{g^\circ=1}$. Each graph G has a number of fields that are simply represented by iconic

[1] http://www.itesoft.com.

boxes $\{B_b\}_{b=1}^{\mathbb{B}}$ and strings (i.e., content) $\{\text{str}_{st}\}_{st=1}^{\mathbb{ST}}$ itself. In this framework, two different evaluation metrics are proposed.

B.1 Area-ratio-based metric (ARM). – Unlike the evaluation made for a complete table detection [12], in our framework, we use the overlapping ratio between the two key field boxes,

$$\text{OR}_0(B_{b^\circ}^\circ, B_b) = \frac{2 \times |B_{b^\circ}^\circ \cap B_b|}{|B_{b^\circ}^\circ| + |B_b|} \text{ and } \text{OR}_0(,) \in [0,1], \quad (12)$$

where $|B_b^\circ \cap B_b|$ is the intersected or common area of them and $|B_b^\circ|, |B_b|$ are the individual areas. Based on this, we use two different cases:

$$\begin{aligned} &\text{case 1. } \text{OR}_1(,) = \begin{cases} 1 : \text{if } \text{OR}_0(,) > 0.8, \text{ and} \\ 0 : \text{ otherwise; and} \end{cases} \\ &\text{case 2. } \text{OR}_1(,) = \text{OR}_0(,). \end{aligned} \quad (13)$$

We aggregate all $\text{OR}_1(,)$ to compute overlapping ratio between the graphs, $\text{OR}_2(G^\circ, G) = \frac{1}{\max(\mathbb{B}^\circ, \mathbb{B})} \sum \text{OR}_1(B_{b^\circ}^\circ, B_b)$. Then for a whole table, we can express evaluation metric as

$$\begin{aligned} \text{Eval}_{\text{ARM}}(O^\circ, O) = \frac{1}{\max(\mathbb{G}^\circ, \mathbb{G})} \sum \text{OR}_2(G_{g^\circ}^\circ, G_g), \\ \{g^\circ : g^\circ \in O^\circ \wedge g \in O^\circ\}. \end{aligned} \quad (14)$$

Both cases are separately used and analysed.

B.2 String-matching-based metric (SMM). – In this, we use strings and compute normalised Levenshtein distance (NLD),

$$\text{NLD}_0(\text{str}_{st^\circ}^\circ, \text{str}_{st}) = 1 - \left(\frac{\text{Lev. dist.}(\text{str}_{st^\circ}^\circ, \text{str}_{st})}{\max(\text{str}_{st^\circ}^\circ, \text{str}_{st})} \right). \quad (15)$$

Based on this, as before, we use two different cases:

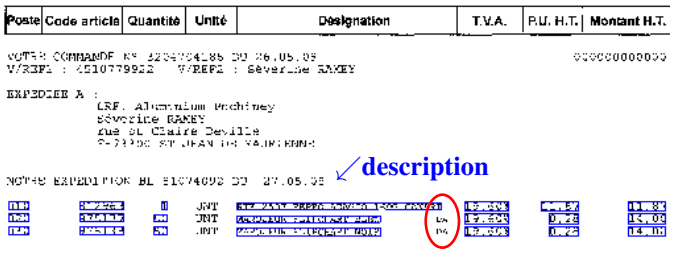
$$\begin{aligned} &\text{case 1. } \text{NLD}_1(,) = \begin{cases} 1 : \text{if } \text{NLD}_0(,) > 0.8, \text{ and} \\ 0 : \text{ otherwise; and} \end{cases} \\ &\text{case 2. } \text{NLD}_1(,) = \text{NLD}_0(,). \end{aligned} \quad (16)$$

Both cases replace Eq. (13) and as a consequence we can update Eq. (14) respectively with $\text{Eval}_{\text{SMM}}(,)$.

C. Results and analysis.

For each input pattern, we have measured the performance of the system by taking the associated ground-truths. Before reporting an overall performance of the system, we start with an example as shown in Fig. 7, aiming to provide an intuitive feeling on how evaluation has been made from two different metrics. In this example, we observe that $\text{Eval}_{\text{SMM}}(,)$ provides better results in comparison to $\text{Eval}_{\text{ARM}}(,)$. It is primarily due to missing portions in the *description* fields, that are encircled (in red) and found in two different lines. This missing portion contains just a couple of letters. When computing SMM, a couple of letters does not make a big difference but affects ARM since empty space is also taken into account in between the strings. In what follows, we aim to analyse whether similar phenomenon will be observed in overall performance.

Table I shows an overall performance of the system, evaluated over 25 different suppliers. It is mainly composed of two different ways of validation: one is associated with the input pattern created in the laboratory and another one is directly related with client or real-world patterns. In Table I, we observe that cleaner the input pattern, better the performance. This happens to be in test n^o 1 since input patterns are created



(a) An example of the information exploitation output where missing portions are encircled in red, in both 2^{nd} and 3^{rd} items, when considering *description* in the extracted table.

	item 1	item 2	item 3	Avg.
Eval_{ARM}	100 $_{100}$	86 $_{94}$	86 $_{94}$	90 $_{96}$
Eval_{SMM}	100 $_{100}$	100 $_{98}$	100 $_{98}$	100 $_{98}$

(b) Evaluation in %, from two different metrics, using the output in (a). Scores are given in the form of case $1_{\text{case 2}}$.

Fig. 7. An example to see how $\text{Eval}_{\text{ARM}}(,)$ and $\text{Eval}_{\text{SMM}}(,)$ are affected due to missing small portion.

TABLE I. AVERAGE RECOGNITION PERFORMANCE (IN %).

		Header	Body	Footer	Avg.
Eval_{ARM}	Test n^o 1	94 $_{97}$	95 $_{99}$	95 $_{98}$	95 $_{99}$
	Test n^o 2	92 $_{96}$	94 $_{98}$	93 $_{95}$	93 $_{96}$
Eval_{SMM}	Test n^o 1	97 $_{98}$	99 $_{99}$	99 $_{98}$	98 $_{98}$
	Test n^o 2	97 $_{95}$	98 $_{96}$	95 $_{95}$	97 $_{95}$

Performance score: $X_{\ Y}$ = case $1_{\text{case 2}}$
Test n^o 1: input patterns created in lab.
n^o 2: input patterns from clients.
Execution time $\simeq$ 2 sec./doc. image.

in accordance with what OCR results. In contrast, in case of the client input patterns (in test n^o 2), a single field selection may sometimes take word(s) from another closer fields (can be left or right), and multiple lines. Besides, we have observed better results in body zone, compared to footer and header because it contains more structured contents i.e., table-like formats. Considering body zone, further label-wise analysis is provided in Table II where *description* are not well spotted i.e., some words may be missed (*cf.* Fig. 7), for instance.

Fig. 8 shows a few demonstrations where both linear as well as zig-zag type patterns are taken into account including missing fields as well as missing lines. In this illustration, body is found in all documents, but not always header and footer. In Fig. 8 (a), the last body pattern does not detect the first field due to OCR error. On the other hand, the detected pattern maintains its structure similar to others regardless the alignment of the fields within it – which in fact, demonstrates

TABLE II. LABEL-WISE AVERAGE PERFORMANCE (IN %).

	Eval_{ARM}	Eval_{SMM}
1. enumerate	100 $_{100}$	100 $_{100}$
2. reference/code article	95 $_{97}$	100 $_{98}$
3. description	88 $_{92}$	96 $_{94}$
4. quantity	93 $_{95}$	100 $_{98}$
5. price/cost/amount/TVA	100 $_{98}$	100 $_{99}$

Performance score: $X_{\ Y}$ = case $1_{\text{case 2}}$.

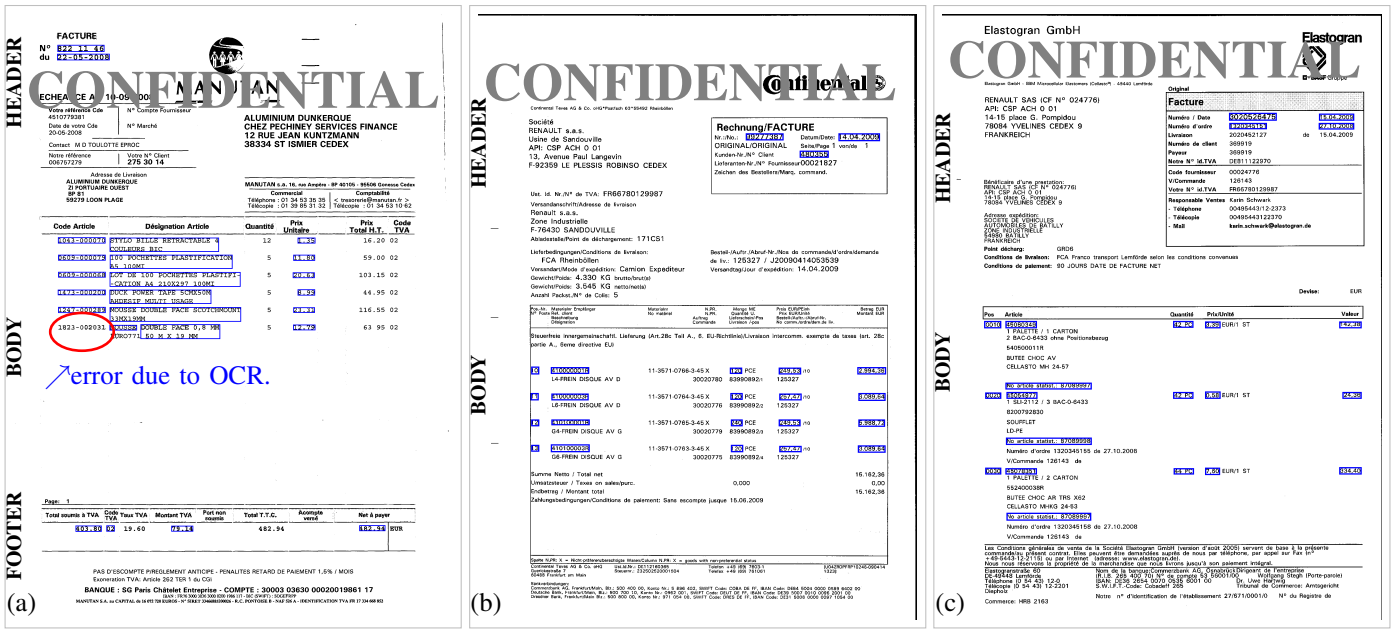


Fig. 8. Examples showing content extraction in accordance with the learnt patterns (from client). Linear as well as zig-zag type patterns with missing fields as well as fields with multiple lines are implemented. Except header and footer, body provides table which is composed of separately (a) six, (b) four and (c) three similar patterns from three different suppliers. For remaining zones, a unique similar pattern is spotted.

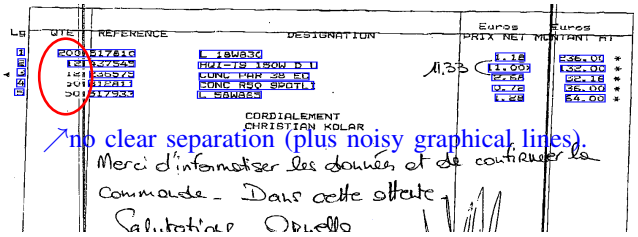
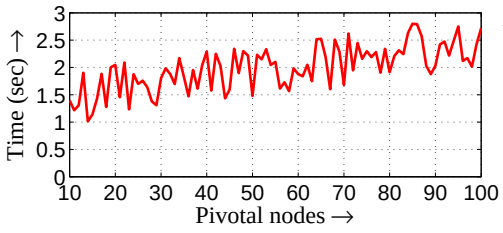


Fig. 9. An example showing an encircled part where OCR is failed to read.



Running system: MATLAB in Linux platform.

Fig. 10. Graph mining time complexity (in sec) in accordance with the number of pivotal nodes selection in document.

the interesting part of concept. Furthermore, OCR errors do not provide the expected semantics label at nodes in the graph. Fig. 9 shows another example of it.

Regarding time complexity issue, graph mining is faster (approx. 2 sec. in average per document image) since for each pivotal node v_i in a document, our search space is basically limited to $deg(v_i^q) \in \mathcal{R}$ in a pattern graph. It depends on how many pivotal nodes are selected as shown in Fig. 10.

IV. Conclusions and Future Perspectives

In this paper, we have presented client-centred pattern-based document information content exploitation via graph mining scheme. We have very much focussed and validated that the table extraction does not always mean only to detect the presence as well as to spot the area where table(s) is(are) located but also to select important key fields within it while rejecting others. Besides, we have shown the concept can be employed to exploit structure-free documents, which is still considered as its further extension.

References

[1] R. Zanibbi, D. Blostein, and J. R. Cordy, "A survey of table recognition," *IJDAR*, 7(1): 1–16, 2004.

[2] B. Coüasnon, "Dmos, a generic document recognition method: application to table structure analysis in a general and in a specific way," *IJDAR*, 8(2-3): 111–122, 2006.

[3] M. Hurst, "Towards a theory of tables," *IJDAR*, 8(2-3): 123–131, 2006.

[4] D. W. Embley, M. Hurst, D. P. Lopresti, and G. Nagy, "Table-processing paradigms: a research survey," *IJDAR*, 8(2-3): 66–86, 2006.

[5] S. Mandal, S. P. Chowdhury, A. K. Das, and B. Chanda, "A simple and effective table detection system from document images," *IJDAR*, 8(2-3): 172–182, 2006.

[6] Y. Liang, Y. Wang, and E. Saund, "A method of evaluating table segmentation results based on a table image ground truther," in *ICDAR*, pp. 247–251, 2011.

[7] F. Deckert, B. Seidler, M. Ebbecke, and M. Gillmann, "Table content understanding in smartfix," in *ICDAR*, pp. 488–492, 2011.

[8] T. Washio and H. Motoda, "State of the art of graph-based data mining," *SIGKDD Explorations*, 5(1): 59–68, 2003.

[9] K.C. Santosh and A. Belaïd, "Pattern-based approach to table extraction," in *IbPRIA*, 2013, to appear.

[10] D. Papadias and Y. Theodoridis, "Spatial relations, minimum bounding rectangles, and spatial data structures," *IJGIS*, 11(2): 111–138, 1997.

[11] M. Weber, M. Liwicki, and A. Dengel, "Faster subgraph isomorphism detection by well-founded total order indexing," *PR Letters*, 33(15): 2011–2019, 2012.

[12] F. Shafait and R. Smith, "Table detection in heterogeneous documents," in *DAS*, pp. 65–72, 2010.