Pattern-Based Approach to Table Extraction

Santosh K.C., Abdel Belaïd

Pattern-Based Approach to Table Extraction

K.C. Santosh and Abdel Belaïd

LORIA – Université de Lorraine
BP 239 - Loria Campus Scientifique, 54506 Nancy Cedex, FRANCE
{santosh.kc, abdel.belaid}@loria.fr

Abstract. In this paper, we address a client-driven approach to automatically extract information content within the table in document images. We start with a graph-based representation of a set of key-fields selected by clients and perform graph mining in a document in order to learn them to produce a model. Such models are aimed to use to extract information content in the absence of clients. To avoid NP-hard general problem, our graph matching is based on relation assignment to see whether pairs of nodes are semantically identical. We have validated the concept by using a real-world industrial problem.

Keywords: Input Pattern, Attributed Relational Graph, Graph Mining, Table Extraction.

1 Introduction

In document analysis and or processing, table extraction from document images has been received an important attention. In the context of table extraction [1,2,3,4], document image analysis and processing basically describes table 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 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 the content understanding [7]. Note that structural information i.e., considering relations between the contents, for instance can be very useful for indexing and retrieving the information contained in the document [2]. To analyse table-forms structure, rulings techniques are basically limited without a priori knowledge about table organisation [1]. Basically, it uses interest points such as intersections (crossings) and corners, that are not robust enough to be able to handle broken rulings. Such concepts are completely failed since not all tables possess graphical lines. Besides, plain ascii texts [8], text blocks [9] are used. Detecting columns, lines and headers and representing them in terms of graph [10], for instance is interesting. But considering real-world applications, one cannot find perfect alignment of table columns and thus methods like [11] are not worth-taking since local rules are inevitable.

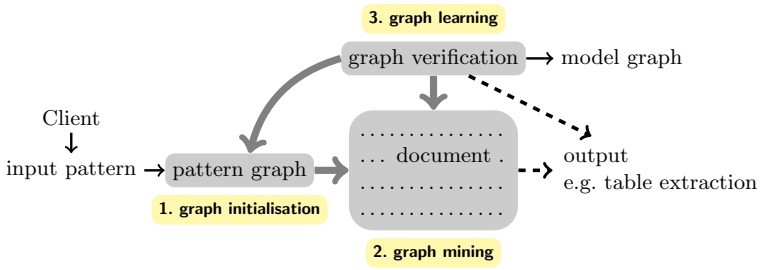


Fig. 1. Schematic block diagram — three main phases: graph initialisation, graph mining and graph learning.

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 lets say, input pattern. These key-fields are then used to develop a pragmatic graph model so that they can be applied for table extraction in the absence of clients.

The rest of the paper is organised as follows. We start with explaining the proposed method in Section 2. Full experiments are reported and analysed in Section 3. The paper is concluded in Section 4.

2 Outline of the proposed method

Following Fig. 1, clients first provide key-fields within the table. 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 used to perform 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 [12], we use semantic labels to confirm structural similarity 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 either a complete content within the table or a specific part of it in accordance with the client.

2.1 Graph initialisation

For any document d, clients provide input pattern(s) i.e., $\{\text{pattern}_n, n \in [1, \mathbb{N}]\}$, where the number of input patterns can be arbitrary. An example of input pattern is shown in Fig. 2 (a). An input pattern is just a collection of the selected

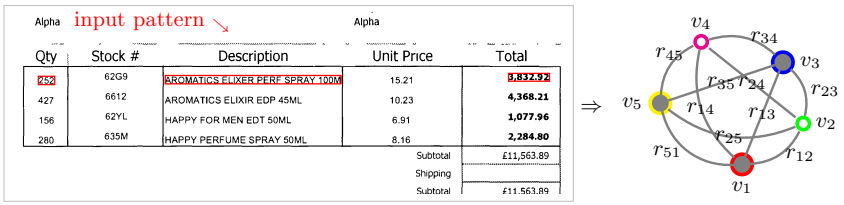


Fig. 2. An example of the input pattern including missing fields. On the right, their corresponding graphs are shown.

key-fields i.e., $\{\text{field}_i\}_{i=1}^{\mathbb{A}}$. To represent each field, we define a feature set $\mathcal{F}$. As an example, for any i-th field,

$$\begin{aligned}\text{field}_i^{\mathcal{F}} = \{\ & (\texttt{box}\text{: } [left, top, right, bottom]);\ (\texttt{wSep}\text{: words separation});\\ & (\texttt{value}\text{: content});\ (\texttt{noW}\text{: number of words used});\\ & (\texttt{type}\text{: content type});\ (\texttt{noL}\text{: number of lines});\\ & (\texttt{size}\text{: string length});\ (\texttt{label}\text{: } \textit{date} \text{ and } \textit{amount}\text{, for instance.})\}\end{aligned} \tag{1}$$

Thanks to the regular expressions, the labels are the derivative of features, representing semantic values. To exploit relative positioning between the key-fields, we basically use bounding box and its projection into 3×3 partitions which are defined in IR^2 i.e., spatial predicates like left, right and top, as presented in [13]. To integrate more precision about the level of neighbourhood k into the basic predefined set of spatial predicates defined in $\mathcal{R}$, we have

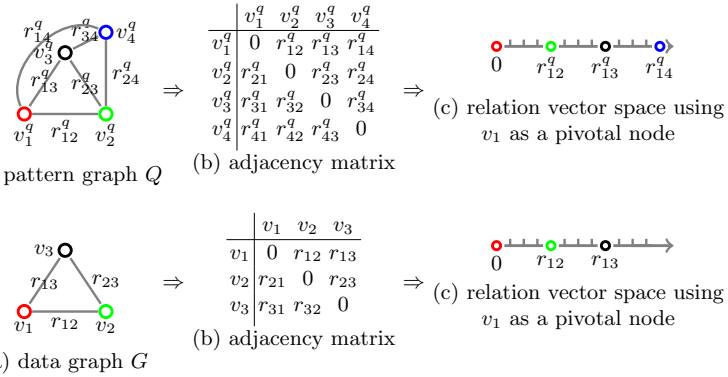
$$r_{ij} = \textit{spatial predicate}_{k_1,k_2}(\text{field}_i, \text{field}_j). \tag{2}$$

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

Altogether, we have an attributed relational graph (ARG) $G(V, E, F_V, F_E)$, 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 $v_i, v_j \in V$. $F_V : V \rightarrow L_V$, L_V represents a set of nodes as well as their labels defined in the particular domain. $F_E : E \rightarrow R_E$, R_E represents the edges via relations. Note that the selected list of key-fields however, may not provide sufficient information. Therefore, in our graph, we introduce missing as well as neighbouring fields. To determine the text blocks, we simply use inter and intra-field separation knowledge from the document images. Now, we separate those non-selected fields with '0' activation key and '1', otherwise. In Fig. 2, we have $\{v_i\}, i = [1, \ldots, 5]$ where node activation signature is [1 0 1 0 1], spanning horizontally (from left to right).

2.2 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.



(a) pattern graph Q (b) adjacency matrix (c) relation vector space using v_1 as a pivotal node

(a) data graph G (b) adjacency matrix (c) relation vector space using v_1 as a pivotal node

Fig. 3. A single relation vector space is shown to simplify relation assignment process. Taking a single pivotal node v_1 from a data graph G (with $\ell_1 = \ell_1^q \in \mathcal{L}$), the idea is to assign relations $\{r_{12}^q, r_{13}^q, r_{14}^q\}$ in data graph G. It provides $G \subseteq Q$.

1. For every node v_i^q in pattern graph Q, the corresponding label $\ell_i^q \in \mathcal{L}$ is defined i.e., $V^q = \{(v_i^q, \ell_i^q), i = 1 \dots \mathbb{V}^q\}$. Having these labelled nodes in Q, the target is to select nodes sharing identical labels $\{v_{\hat{i}}, \ell_{\hat{i}}\}$ in a document.
2. Each pivotal node is taken and started to validate relations with neighbouring nodes in a document, as in pattern graph. To simplify the explanation, as in Fig. 3, let us first create a relation vector space from a pattern graph and then realise the assignment process for each pivotal node in a document.
3. 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 validates with r_{ij}. More formally, we can compute feature score between two corresponding nodes v^q and v as

$$f.\text{score}(v^q, v) = \begin{cases} 1: \text{ label in } v^q = \text{ label in } v, \text{ and} \\ \frac{1}{\mathbb{F}} \sum_f \lambda_f \times s_{v^q,v}^{\text{feature}_f} : \text{ otherwise,} \end{cases} \tag{3}$$

where $\lambda_f \in [0, 1]$ provides weight to each features used to compute feature matching score $s_{(,)}$. We compute feature such as Levenshtein distance between the string *values*, difference in number of *word*s and *size*.

Formally, matching score S for data graph G with respect to Q is aggregation of both relations validation and feature scores

$$S(Q, G) = \alpha \frac{1}{\mathbb{R}} \sum_{i,j \in \mathcal{R}^q, i \neq j} r.\text{score}(r_{ij}^q, r_{ij}) + (1 - \alpha) \frac{1}{\mathbb{V}^q} \sum_{i \in V^q} f.\text{score}(v_i^q, v_i), \tag{4}$$

where $\alpha \in [0, 1]$. Using all possible $\{\ell_i^q\}$, we extract a set $\mathcal{G}$ of similar graphs plus their corresponding matching scores via $S(,)$ i.e., $\mathcal{G} = \{(G_g, S_g)\}_{g=1}^{\mathbb{G}}$. Since we employ word-level pivotal selection, labels like *description* are not used. These fields are extracted with the help of neighbouring labels, by taking structural i.e., relations and features like *size*, *noW* and *noL*. Due to this, faster graph isomorphism [14] does not fit into our application.

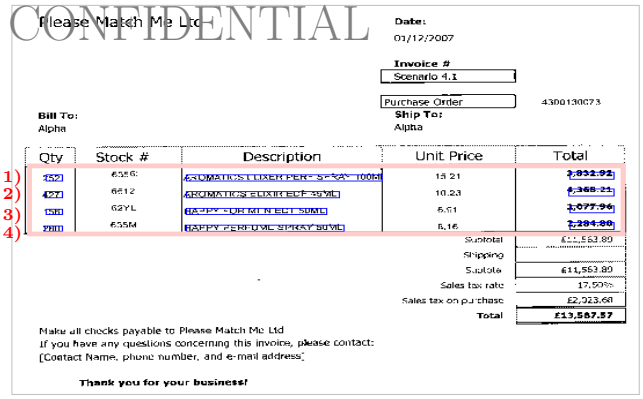


Fig. 4. An example showing pattern-wise output when activation node signature [1 0 1 0 1] is used, shown in Fig. 2. In the output, it provides four different graphs.

2.3 Graph models and table extraction

To learn a graph model, pruning is essential since not all extracted graphs can be used. To handle it, we employ two major criterion: graph consistency and matching score. The graph is said to be consistent if $\ell_{\hat{i}} \in \mathcal{L}$. It is not always certain that all nodes in G possess pre-defined labels. If so, we are then based on matching score that crosses the threshold which is empirically designed. After pruning, corresponding 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. 4 let us take field$_3$:

$$\begin{aligned}\text{field}_3^{\mathcal{F}} = \{\ & (\texttt{box}\text{: } [754, 1700, 1429, 1726]); && (\texttt{wSep}\text{: } `[4, 7]'); \\ & (\texttt{value}\text{: } `\{\text{AROMATICS' 'ELIXER' 'PERF' 'SPRAY' '100M'}\}'), && (\texttt{noW}\text{: } `[4, 5]'); \\ & (\texttt{size}\text{: } `[18, 28]'); && (\texttt{noL}\text{: } [1]); \\ & && (\texttt{label}\text{: } `\{\textit{description}\}')\}.\end{aligned} \tag{5}$$

Keeping relations as in pattern graph, the updated pattern graph will be a model graph $M(V, E, \hat{F}_V, F_E)$ in addition to matching score.

For any document d belonging to class $\mathbb{k}$, we have $\{\mathrm{M}_{\mathbb{k}}^n\}$ models. Since there exists several different input patterns, such a variation brings model variants. Considering all classes, a set $\mathcal{M}$ of models $\{\mathrm{M}_{\mathbb{k}}^n\}_{\mathbb{k}=1}^{\mathbb{K}}$. These models are used to exploit tables in test documents. From each model, as soon as we have extracted similar graphs, we compute confidence score (CS) i.e., an aggregation of all matching scores $\{S_g\}$, which is then normalised i.e., $\mathrm{CS}_{\mathbb{k}}^n = \frac{1}{\mathbb{G}} \sum_{g=1}^{\mathbb{G}} S_g$.

3 Experiments

Dataset and ground-truth formation. We work on a real-world industrial problem in direct collaboration with the **ITESOFT**[1], France. Currently, the

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

volume of our dataset is more than 1,000 scanned document images representing 30 classes and number of samples per each class is ranging from 30 to 100.

For each document, clients provide ground-truths i.e., all similar patterns within the table, according to the pattern selected.

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

To evaluate, we extend the area-ratio-based measure proposed by Shafait and Smith [11]. It uses bounding boxes to describe detected tables and the ground-truths. In our framework, the overlapping ratio between the two boxes is defined as $OR_1(B_b^\circ, B_b) = \frac{2\times|B_b^\circ \cap B_b|}{|B_b^\circ|+|B_b|}$, where $|B_b^\circ \cap B_b|$ is the intersected or common area of two bounding boxes from ground-truthed and detected table respectively and $|B_b^\circ|, |B_b|$ are the individual areas. Note that $OR_1(,) \in [0, 1]$.

We sum up all $OR_1(,)$ and normalise to compute overall overlapping ratio between ground-truth pattern G° and detected pattern G by

$$OR_2(G^\circ, G) = \frac{1}{\max(\mathbb{B}^\circ, \mathbb{B})} \sum OR_1(B_b^\circ, B_b), \quad \{b^\circ : b^\circ \in \mathbb{B}^\circ \wedge b \in \mathbb{B}^\circ\}.$$

Then for a whole table, we can express evaluation metric as

$$Eval(O^\circ, O) = \frac{1}{\max(\mathbb{G}^\circ, \mathbb{G})} \sum OR_2(G_g^\circ, G_g), \quad \{g^\circ : g^\circ \in O^\circ \wedge g \in O^\circ\}. \tag{6}$$

Results and analysis. To evaluate the proposed method, it makes sense to confront model learning quality. Therefore, our experimental tests will be carried out in two major modules: 1) learning and 2) testing. Learning dataset $\mathcal{DS}_{learn}$ will cover up to 60% of the complete dataset $\mathcal{DS}_{complete}$, with the step of 20%. In this framework, we have used the patterns created in the laboratory (mimicking the clients) as well as the real-world patterns from the clients. Both include linear (that spans horizontally along a single line) and sometimes non-linear (i.e., zig-zag) patterns. We have also highlighted in the experiments to know whether non-selected fields are necessary to complete a graph-based pattern representation. Based on these, results are summarised in Table 1. For visual understanding, Fig. 5 shows an example of table extraction.

In the reported results in Table 1, we observe the following. Without a surprise, higher the learning datasets, better the performance but not really surprising. Results in evaluation 1 (*cf.* Table 1), are better when input patterns are clean in comparison to the results presented in evaluation 2 where patterns are taken directly from real-world clients. It is in fact due to, for instance, a single field selection via clients, may sometimes contain word(s) from another closer fields (left and right), and many lines containing unnecessary words. In addition, overlapping of two different fields is possible especially when they are quite close to each other. As a consequence, feature properties representing the graph nodes can possibly varied. Considering the issue of missing and neighbouring fields in the pattern graph, the results can be compared with and without

Table 1. Results showing the performance (in %) over several different subsets of the learning and testing datasets. Test goes in accordance with (W) and without (WO) non-selected fields in a pattern i.e., W ‖ WO.

Perform. factor $(pf.)\rightarrow$		$pf. = 1$	$pf. = 2$	$pf. = 3$	Average
Evaluation 1.	Learning dataset	92 ‖ 81	94 ‖ 78	95 ‖ 82	94 ‖ 81
	Testing dataset	90 ‖ 74	92 ‖ 75	93 ‖ 76	92 ‖ 75
Evaluation 2.	Learning dataset	86 ‖ 74	86 ‖ 75	85 ‖ 76	86 ‖ 75
	Testing dataset	84 ‖ 72	85 ‖ 71	86 ‖ 74	85 ‖ 72

Learning dataset $\mathcal{DS}_{learn} = \mathcal{DS}_{learn0} \times pf.$,
where $\mathcal{DS}_{learn0} = 0.2 \times \mathcal{DS}_{complete}$.
Testing dataset $\mathcal{DS}_{test} = \mathcal{DS}_{complete} - \mathcal{DS}_{learn} \times pf.$
Processing time $\simeq$ 2 sec. per document image.

it. However, in case of input patterns with complex structural formats (lets say zig-zag), missing and neighbouring fields integration makes pattern graph more complex. Furthermore, as said before, our system performance has been affected due to OCR errors.

Regarding time complexity issue, graph mining is fast i.e., less than 2 sec. in average per document image). On the whole, considering the complexity of the problem, our concept provides recognition performance is encouraging.

4 Conclusions and future perspectives

In this paper, we have presented client-driven pattern-based table extraction via graph mining scheme. We have very much focussed and validated that the table extraction does not always mean only to detect the presence and absence 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.

Our current framework thus, opens the concept of exploiting structure-free documents, which is considered to be one of the further issues of the work. To achieve this, our prototype concept will further integrate dynamic labels and relations, rather than relying on fixed knowledge. Besides, in-depth evaluation in terms of client's relevance is another issue.

References

1. Zanibbi, R., Blostein, D., Cordy, J.R.: A survey of table recognition. IJDAR **7**(1) (2004) 1–16
2. Coüasnon, B.: Dmos, a generic document recognition method: application to table structure analysis in a general and in a specific way. IJDAR **8**(2-3) (2006) 111–122
3. Hurst, M.: Towards a theory of tables. IJDAR **8**(2-3) (2006) 123–131
4. Embley, D.W., Hurst, M., Lopresti, D.P., Nagy, G.: Table-processing paradigms: a research survey. IJDAR **8**(2-3) (2006) 66–86
5. Mandal, S., Chowdhury, S.P., Das, A.K., Chanda, B.: A simple and effective table detection system from document images. IJDAR **8**(2-3) (2006) 172–182

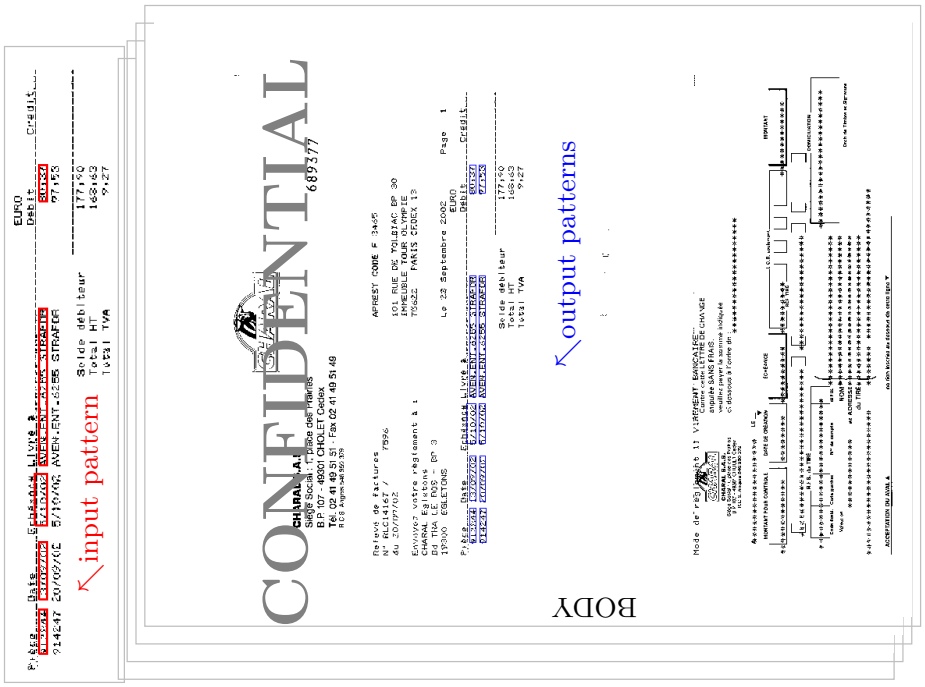


Fig. 5. An example showing an input pattern (in red) and three output patterns (in blue) that compose a table.

6. Liang, Y., Wang, Y., Saund, E.: A method of evaluating table segmentation results based on a table image ground truther. In: ICDAR. (2011) 247–251
7. Deckert, F., Seidler, B., Ebbecke, M., Gillmann, M.: Table content understanding in smartfix. In: ICDAR. (2011) 488–492
8. Hurst, M.: A constraint-based approach to table structure derivation. In: ICDAR. (2003) 911–915
9. Kieninger, T., Dengel, A.: Applying the t-recs table recognition system to the business letter domain. In: ICDAR. (2001) 518–522
10. Ramel, J.Y., Crucianu, M., Vincent, N., Faure, C.: Detection, extraction and representation of tables. In: ICDAR. (2003) 374–378
11. Shafait, F., Smith, R.: Table detection in heterogeneous documents. In: DAS. (2010) 65–72
12. Washio, T., Motoda, H.: State of the art of graph-based data mining. SIGKDD Explorations **5**(1) (2003) 59–68
13. Papadias, D., Theodoridis, Y.: Spatial relations, minimum bounding rectangles, and spatial data structures. IJGIS **11**(2) (1997) 111–138
14. Weber, M., Liwicki, M., Dengel, A.: Faster subgraph isomorphism detection by well-founded total order indexing. PR Letters **33**(15) (2012) 2011–2019