A two-stage approach for tables extraction in invoices

Thomas Saout, Frédéric Lardeux, Frédéric Saubion

A Two-stage Approach for Tables Extraction in Invoices

Thomas Saout
Univ Angers, LERIA, and KaliConseil
F-49000 Angers, France
thomas.saout@etud.univ-angers.fr

Frédéric Lardeux
Univ Angers, LERIA,
F-49000 Angers, France
frederic.lardeux@univ-angers.fr

Frédéric Saubion
Univ Angers, LERIA,
F-49000 Angers, France
frederic.saubion@univ-angers.fr

***Abstract*—The automated analysis of administrative documents is an important field in document recognition that is studied for decades. Invoices are key documents among the documents available in companies and public services. Most of the time, invoices include data that are presented in tables. These tables must be clearly identified to extract suitable information. In this paper, we propose an approach that combines an image processing-based estimation of the shape of the tables with a graph-based representation of the document. We aim to precisely identify different types of tables, including possible complex layouts. We propose an experimental evaluation using a real case application and a classic dataset.**

***Index Terms*—Data extraction; invoice; graph-based representation**

I. Introduction

Automated analysis of administrative documents, particularly invoices, is a significant area of research in document recognition. Commercial systems developed by companies like ITESOFT and ABBYY address the complexity of automating invoice processing. Invoices involve intricate administrative procedures across departments such as accounting, logistics, and the supply chain, requiring specific workflows.

Comprehensive invoice processing involves two essential steps: digitalization using Optical Character Recognition (OCR) techniques and information extraction to identify crucial elements like identifiers, types, amounts, and dates [1], [2] Overcoming these challenges often requires advanced techniques such as machine learning. In summary, the primary challenges in automated invoice processing are: handling layout variability, training and rapid adaptation, minimizing user tasks, efficient table detection and extraction.

Aim and Contribution

The main goal is to automate invoice processing, particularly extracting crucial data from tables. This objective arises from a practical use case within a comprehensive document management system.

Our focus includes various types of information like location, tables, dates, and actors (organizations or individuals) mentioned in the invoice. These fields have been identified based on analysis of multiple invoice models and the current needs expressed by companies. In summary, important invoice information (not restricted to) can be outlined : actors, addressees, dates and tables. In this paper, we propose an integrated approach to extract tabular information from invoices. Our goal is to extract complete data, rather than specific labels like 'total price' or 'product description', enabling automated processing and storage in a dedicated database for invoice information retrieval. In our dataset, tables may not have precise boundaries using lines and may contain missing information.

When analyzing an invoice, we identify two levels of table detection. The first level involves visually detecting characteristic shapes like vertical or horizontal lines. The second level, referred to as the semantic level, involves identifying structural organization based on the arrangement of information tokens in rows and columns, even if there are no explicit graphical boundaries. Our contribution consists of two main aspects. Firstly, we present a comprehensive formalization of the document using ordering relations, which enables more precise processing using a graph-based model of the document's structure. Secondly, we integrate visual analysis of the document with this semantic structure to develop an efficient tool for extracting tables.

II. Related Work

Machine learning techniques, particularly classification, have been widely employed for managing information in scanned invoices. Ongoing research in this field focuses on addressing challenges related to invoice identification, information extraction, and table processing [1]. Initially, the task involved identifying invoices from document sets [3] leading to the development of models to facilitate their processing [4].

Once invoices are correctly scanned and identified, the next crucial step is extracting relevant information from them. Labeling techniques utilizing rules [5] and named entity recognition (NER) using neural networks [6] have been applied for this purpose. Since invoices have distinct structures and text sequences, specific information extraction methods have been proposed to consider these document characteristics. For example, [7] employs a star graph to analyze token neighborhoods within invoices.

Due to the unique structures of invoices, graph-based models that consider the geographical organization of the document have proven to be relevant [8]. Tables are a predominant structural element in most invoices, making table detection a critical processing task [8]. Table processing is indeed an old challenge [9]. This study primarily focuses on table

detection (identifying tabular structures within a document) and extraction (presenting table data in a readable format), which remain active research challenges [10].

Recent approaches [11] aim to detect the overall table framework and extract its content, often utilizing neural networks for recognizing table structures in documents with large training sets [12]. For more specific table types, characteristics such as headers [13] are leveraged to improve performance. Rule-based systems, which were early table extraction techniques, can still be relevant [14].

Graph-based approaches have also been explored, such as graph mining for table extraction using key fields [15]. In [16], a graph convolutional network is employed, combining position and text information, along with visual recognition to predict the number of columns and lines accurately.

III. Ordering relations for document representation

A graph-based representation of the document has been previously explored [8]. Nevertheless, to address more complex structures within the document, we establish multiple relations that correspond to different abstraction levels. This section aims to define these abstraction levels, enabling us to gradually handle the document's basic tokens and incorporate them into more intricate structures. We introduce a model based on ordering and partial ordering relations to precisely define the relationships between the various pieces of information in our documents. Moving forward, we consider a table to consist of lines and columns. Note we will use the term "line" or "row" indifferently in the following.

A. Basic alignment relations

Let us consider two tokens $t_1, t_2 \in T$, where T is the set of tokens obtained after applying the OCR and the tokenization processes. Each token t_i is included in a box whose coordinates are respectively X_{t_i} and Y_{t_i}. We define two basic binary relations on T.

$$r_{horizontal}(t_1, t_2) \Leftrightarrow (|Y_{t_1} - Y_{t_2}| < \tau_Y) \quad (1)$$

$$r_{vertical}(t_1, t_2) \Leftrightarrow (|X_{t_1} - X_{t_2}| < \tau_X) \quad (2)$$

Where τ_X and τ_Y are two thresholds that are set to allow some possible tolerance in the vertical and horizontal alignments of the tokens.

Nevertheless, this approach has some limitations in the presence of texts that are justified using central or right alignments. Hence, we propose to use the interval, inspired by [17]. We consider that there is an alignment relation as soon as the interval of positions containing the first and last characters of the tokens have an intersection. For a token $t_i \in T$ this interval is denoted $[X_{t_i}, endX_{t_i}]$, since X_{t_i} was the position on the X-axis of the first character. Hence, we get

$$\begin{aligned} r_{vertical}(t_1, t_2) \quad \Leftrightarrow \quad & X_{t_1} \in [X_{t_2}, endX_{t_2}] \vee \\ & X_{t_2} \in [X_{t_1}, endX_{t_1}] \end{aligned} \quad (3)$$

Let us note that, due to the use of intervals, $r_{horizontal}$ and $r_{vertical}$ are not necessarily equivalence relations. Hence, we may have $r_{horizontal}(t_1, t_2)$ and $r_{horizontal}(t_2, t_3)$ and, due for instance to a progressive offset to the bottom of the document, $\neg r_{horizontal}(t_1, t_3)$. Hence, alignment relations are computed using a dedicated algorithm that uses the coordinates of the tokens that have been extracted from the tasks previously described. This algorithm handles the previous above-mentioned cases to transform the relations into equivalence relations using thresholds. For instance, it will decide that t_1 and t_2 will be on the same line while t_3 belongs to the next line.

Hence, these relations allow us to get an abstract of the relative positions of the token, which is the relevant information for extracting tables. Since these relations are now equivalence relations, they induce equivalence classes between tokens. Column and line labels can directly be obtained from these equivalence classes, one label per class since we are only interested in determining if tokens are on the same line/column. Using the coordinates of the tokens, we assume that we get a totally ordered set of line labels $(L, <)$ and a totally ordered set of column labels $(C, <)$.

B. Ordering relations on tokens

We consider now that a document is a set of labeled tokens T such that each token $t \in T$ has a line label $l(t) \in L$ and a column label $c(t) \in C$. We use the total ordering relations previously defined on labels to get partial ordering relations on tokens.

Definition 1: Given the set of labeled tokens T we define two partially ordered sets $(T, \preceq_l)$ and $(T, \preceq_c)$ where

- $t \preceq_l t'$ iff $l(t) < l(t')$ and $c(t) = c(t')$
- $t \preceq_c t'$ iff $c(t) < c(t')$ and $l(t) = l(t')$

We may now precisely define the concept of lines and columns.

Definition 2 (Lines): A line is any subset $L \subseteq T$, such that $(L, \preceq_l)$ is a totally ordered set (linearly ordered). $(L, \subseteq)$ is the partially ordered set of all possible lines.

A full line is a subset $L \subseteq L$ such that L is a maximal element of $(L, \subseteq)$. Note that if L is a full line then $sup_l(L) \in L$ and $inf_l(L) \in L$.

Definition 3 (Column): A column is any subset $C \subseteq T$, such that $(C, \preceq_c)$ is a totally ordered set. $(C, \subseteq)$ is the partially ordered set of all possible columns.

A full column is a subset $C \subseteq C$ such that C is a maximal element of $(C, \subseteq)$. Note that if C is a full column then $sup_c(C) \in L$ and $inf_c(C) \in L$.

C. Tables through different levels of abstraction

Hence, a table is a set of tokens composed of lines and columns that share elements. Let us define a table ordering on T as $\preceq_t = (\preceq_l \cup \preceq_c)^*$, i.e., the transitive closure of the union of the two previous ordering relations on lines and columns. Let us note that for two tokens $t, t' \in T$, $t \preceq_t t'$ iff $l(t) \leq l(t')$ and $c(t) \leq c(t')$.

Definition 4 (Table): A table is any subset $T \subseteq \boldsymbol{T}$, such that $(T, \preceq_t)$ is a complete lattice. As a consequence table $T \subseteq \boldsymbol{T}$ satisfies $sup_t(T) \in T$ (its greatest element $\top_T$) and $inf_t(T) \in T$ (its least element $\bot_T$). A table can be then defined by these two elements $T = (\bot_T, \top_T)$.

As a consequence, in a table T, given any two tokens $t_1, t_2 \in T$, $sup_t(\{t_1, t_2\})$ and $inf_t(\{t_1, t_2\})$ exist and constitute the smallest sub-table $T' = (sup_t(\{t_1, t_2\}, inf_t(\{t_1, t_2\})$ that contains t_1 and t_2.

Let us consider $(\boldsymbol{T}, \subseteq)$ a partially ordered set of all possible tables. A full table is a subset $T \subseteq \boldsymbol{T}$ such that T is a maximal element of $(\boldsymbol{T}, \subseteq)$.

Definition 5 (Well-formed full table): A full table T is well-formed iff:

- T can be partitioned into a set $\{L_1, \cdots, L_n\}$ of full lines, such that $sup_t(T) = sup_l(L_n)$ and $inf_t(T) = inf_l(L_1)$
- T can be partitioned into a set $\{C_1, \cdots, C_m\}$ of full columns, such that $sup_t(T) = sup_c(C_n)$ and $inf_t(T) = inf_c(C_1)$
- $\forall 1 \leq i \leq n, 1 \leq j \leq m, |L_i \cap C_j| = 1$

Since we want to detect general tables, with possibly missing cells, we may turn any table into a well-formed full table by adding empty cells, i.e. new tokens.

- Lines:
  1) T cannot be partitioned into a set $\{L_1, \cdots, L_n\}$ since two lines L_i and L_j intersect: each element of $L_i \cap L_j$ is duplicated such that $L_i \cap L_j = \emptyset$: add then the corresponding ordering relations.
  2) A line L_i is not a full line. There exists $t \in \boldsymbol{T}$ such that $L_i \cup \{t\}$ is a full line. Add to each line $L_j, j \neq i$ a new element t_j such that $\{t\} \cup \bigcup_j \{t_j\}$ is a full column. Add then the corresponding ordering relations.
- Columns: same processes

Note that this repair process will be performed on an initial set of tokens to get better candidate lines and columns to obtain better tables from the documents.

At this stage, we have a clear model of lines, columns, and possibly resulting tables. To reach a higher level of abstraction and to manipulate lines and columns directly, we extend our ordering relations to lines and columns. Let us consider $\boldsymbol{L}^+ \subseteq \boldsymbol{L}$ (resp. $\boldsymbol{C}^+ \subseteq \boldsymbol{C}$) the set of full lines (resp. columns). According to the previous definitions, note that $\boldsymbol{L}^+$ and $\boldsymbol{C}^+$ are partitions of $\boldsymbol{T}$. We consider now the two orderings $(\boldsymbol{C}^+, \preceq_c)$ and $(\boldsymbol{L}^+, \preceq_l)$ that are the canonical extensions of $(\boldsymbol{T}, \preceq_c)$ and $(\boldsymbol{T}, \preceq_l)$. Note that here, because we consider full lines and full columns separately, these orders are total orders. Searching for a table corresponds to searching for a subset of lines and columns. According to previous definitions, a table T can be defined by a couple of sets $T = (L, C)$, such that $L \subseteq \boldsymbol{L}^+$ and $C \subseteq \boldsymbol{C}^+$. The structure of the table is then defined by the possible insertions between lines and columns (i.e. common tokens).

D. From orders to graphs

Our formalism allows us to get a clear and general abstract view of a table in a document, where information has been grouped into basic tokens. Now, since we want to get operational tools for computing tables, we turn our orders into a graph representation using the classic Hasse diagram (i.e. transitivity is not represented to simplify the graph).

We consider a first directed graph $(\boldsymbol{L}^+, E_L$ where $(l_i, l_j) \in E_L$ iff $l_i <_l l_j$ and $\nexists l_k \boldsymbol{L}^+, l_i <_l l_k <_l l_j$. We consider a similar graph ($\boldsymbol{C}^+, E_C$ for the columns with similar properties). We add an undirected graph ($\boldsymbol{L}^+ \cup \boldsymbol{C}^+, E_\cap$ where $[l_i, c_j] \in E_\cap$ iff $l_i \cap c_j \neq \emptyset$).

The final graph is thus a graph that gathers the three previous graphs whose edges have different types according to their semantics (lines and columns ordering or intersections) $(\boldsymbol{L}^+ \cup \boldsymbol{C}^+, E_L \cup E_C \cup E_\cap)$. Using this graph representation, a table corresponds to a particular sub-graph in the graph that represents the whole document. This graph is processed by a solver that is dedicated to sub-graph search [18], allowing our different types of edges. Note that a table may contain empty cells. Hence, we introduce possible empty nodes in our graph representation. Now we have to introduce patterns that correspond to general possible structures of tables that we search for in our graph.

IV. Search Using Patterns

Tables within a document can exhibit diverse formats, each with specific requirements (such as a complete first row or a specified corner cell). To represent these characteristics, we employ graph patterns based on the earlier introduced formalism. These patterns define sets of similar tables. The search for valid tables in a graph based on a given pattern involves solving a graph isomorphism problem, which is a well-known NP-intermediate problem. The solution process for this problem is described in detail in Section III-D.

A pattern represents a set of tables sharing the same characteristics. The number of rows and columns is necessary for pattern definition. Remind that according to Section III-D, the graph corresponding to a pattern is built:

- Each line and column is represented by a vertex in the pattern.
- Vertical and horizontal alignment relations are represented by directed arcs.
- The presence of a cell at the intersection of a row and a column is represented by a non-directed edge between the row vertex and the column vertex.

Figure 1 presents patterns with 4 lines and 4 columns respectively represented by vertices A, B, C, D and E, F, G, H. Each pattern is presented with its table diagram on the right and its corresponding graph on the left. These four patterns are issued from practical needs and correspond to common patterns that are encountered in invoices. They have some specific properties detailed below:

- **Corner Left Top:** We can see that an arc in the pattern induces the presence of a token at the intersection of the

first line (from top to bottom) and the first column (from left to right). No other common cell is required to be shared by the remaining of the table.
- **Full grid:** This table corresponds to a well-formed full table as formally defined in Section III-C.
- **Missing cells:** Some tables may contain empty cells. This pattern shows that some alignment relations may be missing. Nevertheless, such a table is still valid since it can be completed into a full table by introducing empty cells. In our graph model, we can generate a suitable pattern for searching for such a table.
- **Border Left Top:** This pattern is based on the assumption that there is less than one complete line to the top and one complete column to the right in our table.

An undirected edge between a line and a column indicates the requirement for a token at their intersection. However, the absence of an edge does not necessarily mean the absence of an intersection. In Figure 4's corresponding table schemes, the mandatory presence of a token is represented by a light grey circle. In this scheme, lines are depicted as blue rectangles, which may contain empty cells indicating the possible absence of tokens at those locations. Similarly, columns are represented by green rectangles, following the same principles.

Let us note that the pattern presented in Figure 1 can be defined for any number of lines and columns, resulting in vertices in our graph. These two parameters define the size of the pattern. As already explained, searching for a pattern in the whole document modeled as a graph corresponds to the solving of a sub-graph isomorphism problem [18]. Given a pattern type, our objective is to find the pattern of the largest size, i.e. a maximal number of lines/columns. To achieve this incremental pattern search, we use the algorithm proposed in [18] to find a pattern with a size and, as long as no pattern is found, we continue the search for progressively decreasing the size of the pattern.

The graph-based approach for table identification can be computationally intensive due to the complexity of the sub-graph search problem. However, in certain cases where table structures are clearly visible, the detection of horizontal and vertical lines in the document can be a cost-effective method. This detection can be accomplished using image processing tools. By combining the accuracy of our graph-based approach with the efficiency of visual detection, we enhance the robustness of our approach for processing various types of documents.

To further enhance the information extracted from processed documents, we propose integrating image processing tools. The utilization of visual elements, such as lines, is not employed in the approach that focuses on token placement. The OpenCV library [19] is widely utilized for information extraction from documents and offers various functionalities. Among these, the line and contour-based methods are particularly suitable for our requirements. By applying a sequence of treatments, such as RGB-to-Grayscale conversion, thresholding, and bounding rectangle calculations, we can detect graphical information effectively.

OpenCV provides us with local document information. We have developed a tool that utilizes OpenCV's data to extract visually corresponding areas that may represent tables. We use the output of OpenCV as a surrogate for evaluating the likelihood of a table's presence. We define several strategies based on this approximation:

- **Empty:** No visual information is returned, so the document is fully processed using the graph-based approach to check for table presence.
- **Area:** An area is identified visually, but no additional information is provided. Only this area is processed with the graph-based approach.
- **Column:** An area is detected, and vertical lines help identify columns. Only this area is processed with the graph-based approach, with pre-defined column positions.
- **Grid:**An area is detected with a complete grid, allowing table identification. Only this area is processed with the graph-based approach, with pre-defined column and row positions.

The strategy selection depends on the visual processing result. The table detection quality remains consistent across strategies, but computational time is minimized as visual processing is negligible compared to the full document's graph-based approach.

The process we used for our experiments includes the use of OpenCV, our OpenCV-based estimation tool, and our graph-based approach. Figure 2 describes this process graphically. The first step is OCR-independent since OpenCV and our estimation tool only use graphical information. As already mentioned, this step can be a surrogate function because it allows us to characterize the area to be treated and its associated features. The second step introduces more semantic information, using tokens, and it is very dependent on the quality of the OCR. The strategy of the graph-based approach depends thus on the first step, involving more or less computational resources.

V. Experiments

We use a typical dataset from literature for table extraction sci-TSR introduced by [20]. This dataset is composed of documents containing only tables. Note that our main goal is not only to locate a table but to extract tables from complete invoices. This point will be discussed later on. Note that, since most of the methods are based on neural networks, this dataset is decomposed into training and test subsets. For the evaluation, we consider the test subset with 3000 tables.

Concerning our method, we consider different table extraction processes based on the different stages and strategies previously defined. Running time is limited to 300 seconds for each table. Since the choice of the pattern is an important parameter, we have tested different patterns and, as expected, there is no overall better pattern for all the instances. Hence, we only use a pattern similar to the Border Left Top pattern presented in Figure 1 since all tables matching **Full grid** or **Missing cells** also match this pattern. Let us note that we could use **Corner Left Top** that matches all tables also

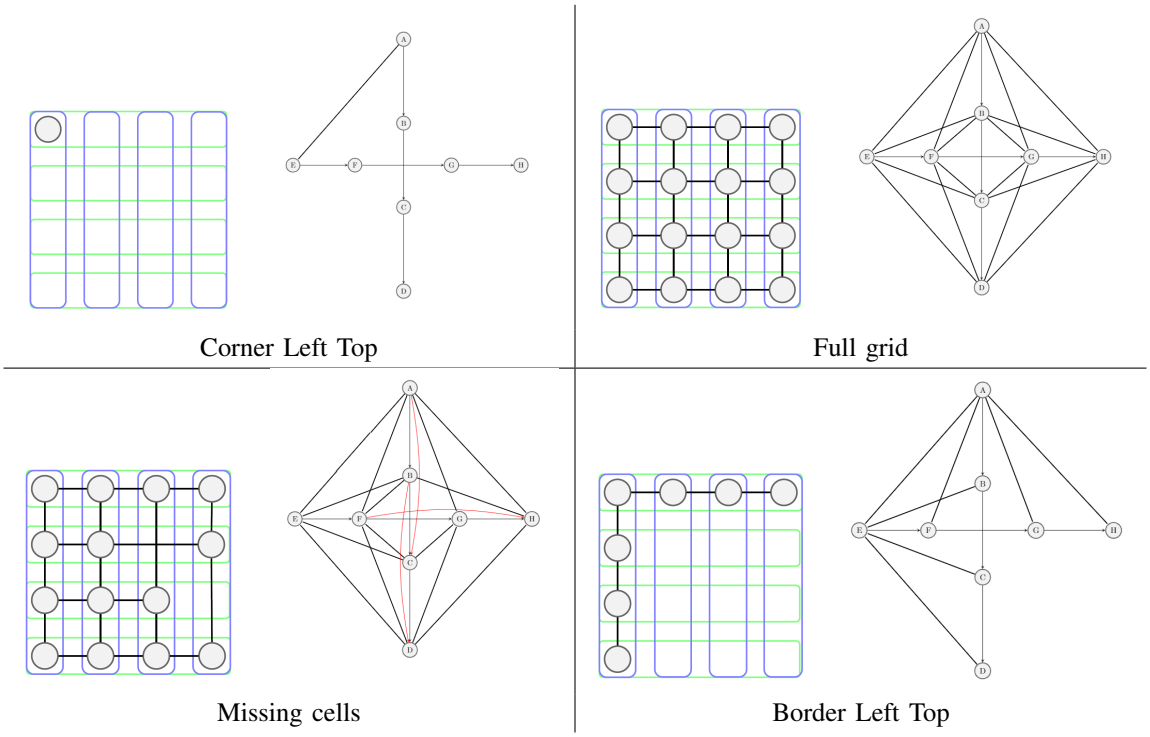

Fig. 1. Different patterns for a 4×4 table.

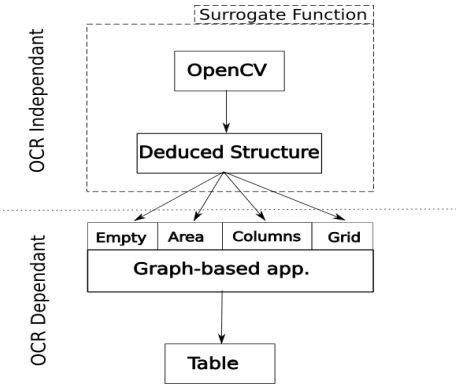

Fig. 2. The overall process of our approach.

recognized by the three other patterns. But this pattern induces a high execution cost. Our proposed method encompasses five versions, each aligned with a specific strategy (Empty, Area, Column, Grid). We also consider here an additional version called Autostrat, that uses OpenCV as a surrogate function to intelligently determine the most suitable strategy.

We introduce an Oracle to showcase the optimal efficiencies of our approach. The Oracle selects the best outcome from each of our strategies, similar to how an end-user would choose the most suitable table from a set of options. This oracle reveals the limitations of our AutoStrat method. This aspect could be improved thanks to a better classification of documents. As part of our future endeavors, we plan to introduce a learning method to enhance this classification process.

In Table I, we provide a comparison between our methods and state-of-the-art algorithms. The evaluation of the output is classically performed using accuracy and recall measures (e.g., see [20]). Basically, the accuracy evaluates the truth of the extracted table (i.e., retrieved elements are correct) and the recall evaluates the ability to retrieve expected elements. Hence, documents need to be labeled to get their ground truth. The classic F1 score combines recall and accuracy. For our evaluation process, we also consider a method proposed in [16]. This method counts the number of vertically and horizontally aligned elements according to the intended table to evaluate the accuracy of structure recognition.

We consider five methods based on deep learning (DL), involving thus a training stage. Some of these methods are based on graph structures (GraphBasedTSR and GFTE). The other methods use DL coupled with dedicated techniques. DeepDESRT is more data-driven. CATT-Net uses a conditional attention network. TabStrucNet combines cell detection and interaction modules to localize the cells and predict their row and column. We consider also a rule-based method Tabby (the references of the papers are given in Table I). Note that for the other methods we report here the results presented in the corresponding papers, since the codes were not available or difficult to run to get same results (hence the alignment measures are only available for GFTE) . The articles that introduce these methods suggest conducting experiments on a specific portion of the sci-TSR test subset without providing precise details about the instances chosen. Here, for the sake of

fairness, we choose to analyze the entire test subset to prevent any potential advantage of our method on specific subsets.

TABLE I
COMPARISONS ON THE SCI-TSR TEST SUBSET

	horizontal alignment	vertical alignment	accuracy	recall	F1
Tabby [14]	-	-	0,914	0,91	0,912
GraphBasedTSR [12]	-	-	0,936	0,931	0,934
DeepDeSRT [21]	No Code		0,898	0,897	0,897
CATT-Net [22]	No Code		0,956	0,965	0,961
TabStrucNet [23]	-	-	0,927	0,913	0,92
GFTE [16]	0,954	0,922	-	-	-
Empty	0,782	0,819	0,852	0,681	0,736
Area	0,867	0,891	0,846	0,702	0,750
Column	0,775	0,802	0,782	0,583	0,641
Grid	0,740	0,755	0,724	0,529	0,584
AutoStrat	0,786	0,769	0,773	0,631	0,678
Oracle	0,940	0,944	0,933	0,834	0,871

Let us note that our rule-based approach achieves very interesting results compared to trained neural networks. Autostrat is not really efficient since the document includes a single table, while a basic Area strategy seems to be the best choice here. The surrogate function that helps us to handle full invoices makes some errors on some documents here. Concerning our approach, the recall score presents some limitations due to the nature of the input documents. In fact poor recall scores are due to 50 documents that reached timeout.

VI. CONCLUSION

Our method based on a graph representation and constraint-based exploration process can be competitive with regard to ML-based methods and it significantly reduces the need for extensive training on large datasets (9000 in the training set that we have considered). In a practical application of our approach, we have tested our method on invoices that include tables and other data. Clearly, table-dedicated methods such as Tabby provide poor results compared to our solution on these benchmarks.

REFERENCES

[1] H. T. Ha and A. Horák, "Information extraction from scanned invoice images using text analysis and layout features," *Signal Process. Image Commun.*, vol. 102, p. 116601, 2022. [Online]. Available: https://doi.org/10.1016/j.image.2021.116601

[2] A. Hamdi, E. Carel, A. Joseph, M. Coustaty, and A. Doucet, "Information extraction from invoices," in *ICDAR 2021, Proceedings, Part II*, ser. Lecture Notes in Computer Science, J. Lladós, D. Lopresti, and S. Uchida, Eds., vol. 12822. Springer, 2021, pp. 699–714. [Online]. Available: https://doi.org/10.1007/978-3-030-86331-9_45

[3] M. Köppen, D. Waldöstl, and B. Nickolay, "A system for the automated evaluation of invoices," in *DAS 1996*, ser. Series in Machine Perception and Artificial Intelligence, J. J. Hull and S. L. Taylor, Eds., vol. 29. WorldScientific, 1996, pp. 223–241. [Online]. Available: https://doi.org/10.1142/9789812797704_0012

[4] F. Cesarini, E. Francesconi, S. Marinai, J. Sheng, and G. Soda, "Conceptual modelling for invoice document processing," in *DEXA*, R. R. W., Ed. IEEE Computer Society, 1997. [Online]. Available: https://doi.org/10.1109/DEXA.1997.617381

[5] A. Dengel and B. Klein, "smartfix: A requirements-driven system for document analysis and understanding," in *5th DAS*, ser. Lecture Notes in Computer Science, D. P. Lopresti, J. Hu, and R. S. Kashi, Eds., vol. 2423. Springer, 2002, pp. 433–444. [Online]. Available: https://doi.org/10.1007/3-540-45869-7_47

[6] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 50–70, 2022. [Online]. Available: https://doi.org/10.1109/TKDE.2020.2981314

[7] V. P. D'Andecy, E. Hartmann, and M. Rusiñol, "Field extraction by hybrid incremental and a-priori structural templates," in *13th IAPR, DAS 2018*. IEEE Computer Society, 2018, pp. 251–256. [Online]. Available: https://doi.org/10.1109/DAS.2018.29

[8] F. Shafait and R. Smith, "Table detection in heterogeneous documents," in *The Ninth IAPR, DAS 2010*, D. S. D., V. G., D. P. L;, and P. N., Eds. ACM, 2010. [Online]. Available: https://doi.org/10.1145/1815330.1815339

[9] R. Zanibbi, D. Blostein, and J. R. Cordy, "A survey of table recognition," *Int. J. Document Anal. Recognit.*, 2004. [Online]. Available: https://doi.org/10.1007/s10032-004-0120-9

[10] L. Gao, Y. Huang, H. Déjean, J. Meunier, Q. Yan, Y. Fang, F. Kleber, and E. M. Lang, "ICDAR 2019 competition on table detection and recognition (ctdar)," in *2019 ICDAR*. IEEE, 2019, pp. 1510–1515. [Online]. Available: https://doi.org/10.1109/ICDAR.2019.00243

[11] T. Kashinath, T. Jain, Y. Agrawal, T. Anand, and S. Singh, "End-to-end table structure recognition and extraction in heterogeneous documents," *Appl. Soft Comput.*, vol. 123, p. 108942, 2022. [Online]. Available: https://doi.org/10.1016/j.asoc.2022.108942

[12] E. Lee, J. Park, H. I. Koo, and N. I. Cho, "Deep-learning and graph-based approach to table structure recognition," *Multim. Tools Appl.*, vol. 81, no. 4, pp. 5827–5848, 2022. [Online]. Available: https://doi.org/10.1007/s11042-021-11819-7

[13] S. C. Seth and G. Nagy, "Segmenting tables via indexing of value cells by table headers," in *12th ICDAR 2013*. IEEE Computer Society, 2013, pp. 887–891. [Online]. Available: https://doi.org/10.1109/ICDAR.2013.181

[14] A. O. Shigarov, A. Altaev, A. A. Mikhailov, V. Paramonov, and E. A. Cherkashin, "Tabbypdf: Web-based system for PDF table extraction," in *ICIST 2018, Proceedings*, ser. Communications in Computer and Information Science, R. D. and G. Vasiljeviene, Eds., 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-99972-2_20

[15] K. Santosh and A. Belaïd, "Pattern-based approach to table extraction," in *Iberian Conference on Pattern Recognition and Image Analysis*. Springer, 2013, pp. 766–773.

[16] Y. Li, Z. Huang, J. Yan, Y. Zhou, F. Ye, and X. Liu, "Gfte: graph-based financial table extraction," in *International Conference on Pattern Recognition*. Springer, 2021, pp. 644–658.

[17] B. Yildiz, K. Kaiser, and S. Miksch, "pdf2table: A method to extract table information from pdf files," in *IICAI*, 2005, pp. 1773–1785.

[18] C. McCreesh, P. Prosser, C. Solnon, and J. Trimble, "When subgraph isomorphism is really hard, and why this matters for graph databases," *Journal of Artificial Intelligence Research*, vol. 61, pp. 723–759, 2018.

[19] G. R. Bradski and V. Pisarevsky, "Intel's computer vision library: applications in calibration, stereo segmentation, tracking, gesture, face and object recognition," in *CVPR 2000 (Cat. No. PR00662)*, vol. 2. IEEE, 2000, pp. 796–797.

[20] Z. Chi, H. Huang, H.-D. Xu, H. Yu, W. Yin, and X.-L. Mao, "Complicated table structure recognition," *arXiv preprint arXiv:1908.04729*, 2019.

[21] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, "Deepdesrt: Deep learning for detection and structure recognition of tables in document images," in *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 1162–1167.

[22] B. Xiao, M. Simsek, B. Kantarci, and A. A. Alkheir, "Table structure recognition with conditional attention," *arXiv preprint arXiv:2203.03819*, 2022.

[23] S. Raja, A. Mondal, and C. Jawahar, "Table structure recognition using top-down and bottom-up cues," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*. Springer, 2020, pp. 70–86.