SN Computer Science (2023) 4:246
https://doi.org/10.1007/s42979-022-01659-z

ORIGINAL RESEARCH

Flexible Hybrid Table Recognition and Semantic Interpretation System

Marcin Namysł[1,2] · Alexander M. Esser[3] · Sven Behnke[1,2] · Joachim Köhler[1,2]

Received: 18 June 2022 / Accepted: 30 December 2022 / Published online: 4 March 2023

Abstract
Extracting information from documents containing quantitative data in tabular format is an important but still unsolved task due to the heterogeneity of document layouts. This work aims to take a step toward developing a solution to this problem. This paper proposes a flexible, hybrid table extraction system consisting of a deep learning-based table detection module, a heuristic-based structure recognition method, and a graph-based semantic interpretation component. The proposed system is modular and supports the most frequent table layouts. Moreover, it handles both the documents in image format and PDF files with embedded text. The proposed system outperforms the baseline method and achieves results on par with state-of-the-art approaches on the challenging benchmarks from ICDAR 2013 and ICDAR 2019 table interpretation competitions. Moreover, we correct an issue with the evaluation script used in the latter competition and report extended results of the proposed method in comparison with a leading commercial product. Finally, our table extraction system achieves a high F_1 score in the scenario where raw documents are given as input and the targeted information is contained in a subset of table columns. The presented system achieves results competitive with leading methods in the field. It has already been evaluated on general-purpose data and biomedical benchmarks. We intend to continuously improve our approach and process data from other domains, e.g., financial documents. To support future research on information extraction from documents, we make the evaluation scripts and results from our experiments publicly available at https://github.com/mnamysl/tabrec-sncs.

Keywords Information extraction · Document understanding · Table detection · Table structure recognition · Table interpretation

Introduction

Automatic table extraction is a challenging task due to the heterogeneity of document types and layouts. Tables in the scientific literature are formatted and typeset differently than tables presenting financial data, tables used in business documents, or tables in advertising materials. Tables are designed to present compressed information to the reader in a way that is easy to comprehend [35]. Nevertheless, automatic table extraction, although widely studied before, has not been completely solved yet.

This work includes results of the doctoral thesis by Marcin Namysł [43] and presents an extended version of the table extraction approach that was previously published in Namysl et al. [22], where a flexible, holistic method that combines table recognition and table interpretation modules was proposed. In this method, two rule-based table recognition heuristics perform table detection and table structure recognition (TSR) in one step. Specifically, for partially bordered tables, a book tabs-based heuristic was developed,

This article is part of the topical collection "Advances on Computer Vision, Imaging and Computer Graphics Theory and Applications" guest edited by Kadi Bouatouch, Augusto Sousa, Mounia Ziat and Helen Purchase.

✉ Marcin Namysł
marcin.namysl@iais.fraunhofer.de

Alexander M. Esser
aesser22@smail.uni-koeln.de

Sven Behnke
behnke@cs.uni-bonn.de

Joachim Köhler
joachim.koehler@iais.fraunhofer.de

1 NetMedia, Fraunhofer IAIS, Sankt Augustin, Germany

2 Autonomous Intelligent Systems, University of Bonn, Bonn, Germany

3 University of Cologne, Cologne, Germany

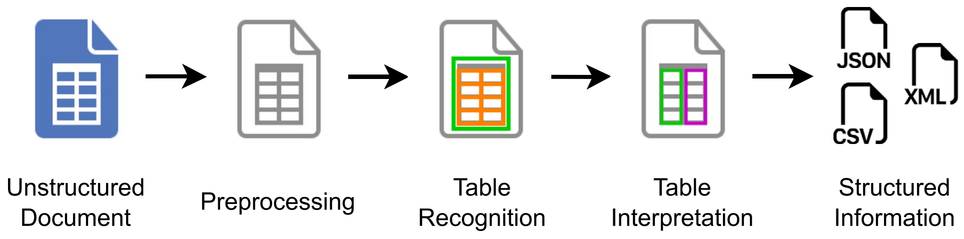


Fig. 1 Diagram of the baseline information extraction system [22]. An unstructured document, either an image or a PDF file, is given as input. Preprocessing is performed prior to table recognition, which detects the table objects and recognizes their building blocks: rows, columns, and individual cells. Table interpretation links the extracted structural elements with predefined semantic concepts. As a result, the layout and the semantic interpretation of a table are written in a structured format. Adapted from Namysł [43]

which recognizes tables that are typeset with a commonly used LaTeX package.[1] For fully bordered tables, a solid separator-based heuristic was implemented. We refer to this table recognition approach as baseline method.

Second, in Namysl et al. [22], the basic formulation of the table recognition task is complemented by a table interpretation module implemented as a rule-based method that leverages regular expressions (RegEx) and an approximate string matching algorithm. It is worth noting that this method was also previously employed to extract and structure quantitative information from a vast number of biomedical articles, as presented by Adams et al. [1] and Lage-Rupprecht et al. [18].

In this work, we address issues identified in these studies by focusing on decreasing the precision-recall gap. To this end, we improve the table recognition component by incorporating a deep learning-based table detection module and combining it with the adapted version of the baseline TSR component. We compare the proposed approach with the baseline method as well as with the state-of-the-art approaches in this field by performing experiments on two challenging benchmarks: the ICDAR 2013 [9] and ICDAR 2019 [7] data sets from the table recognition competitions hold at the International Conference on Document Analysis and Recognition (ICDAR). Our results demonstrate that the proposed hybrid table recognition method achieves better recall and consequently higher F_1 scores, compared to the baseline method. Our approach exhibits recognition accuracy competitive with state-of-the-art approaches on both examined benchmarks (Fig. 1).

Moreover, in the course of our experiments, we found and corrected an issue with the official evaluation tool[2] employed in ICDAR 2019 Table Competition [7]. We published the repository with the corrected script[3] and submitted our changes to the official evaluation tool.[4] Furthermore, we also noticed that the annotations used in this competition were updated recently.[5] To facilitate comparison with previous and future work reporting the results on this benchmark, we include the scores of our method in all scenarios: with and without the corrected script, as well as using the previously used and the recently revised annotations.

To facilitate the reproducibility and fair comparison of the results obtained by different methods on the ICDAR 2013 Table Competition benchmark [9], we release the evaluation script employed in our work. Our script parses the output produced by the official evaluation tool[6] and accumulates them to produce the final document-level scores. It also includes the adjacency relations from the false-positively detected tables to give a better perspective on the actual performance of the table recognition approaches.

We present a formal definition of the table interpretation task and explain the workflow of our method in more detail. For completeness, we also thoroughly describe the table detection and TSR tasks studied in this work.

Figure 2 gives an overview of our approach. Our system is modular and flexible: We are able to easily adapt particular modules to a specific scenario, as different components need to be optimized, depending on layout and type of the input. Our system supports both documents in image format and PDF files. Note that few table recognition methods support both types of input. Most approaches require PDF documents with embedded text.

In summary, this work makes the following contributions:

- We present a formal definition of the table extraction task and its main components: table detection, structure recognition, and interpretation.

[1] https://ctan.org/pkg/booktabs.

[2] https://github.com/cndplab-founder/ctdar_measurement_tool.

[3] https://github.com/mnamysl/ctdar_measurement_tool/tree/table_mapping_fix.

[4] https://github.com/cndplab-founder/ctdar_measurement_tool/pull/1.

[5] https://github.com/cndplab-founder/ICDAR2019_cTDaR.

[6] https://roundtrippdf.com/en/data-extraction/table-recognition-dataset-tools.

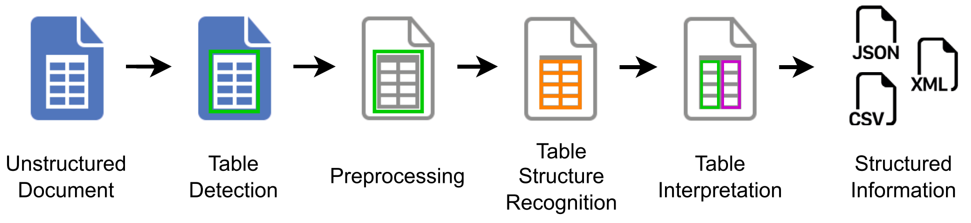


Fig. 2 Overview of our information extraction system. An unstructured document, either an image or a PDF file, is given as input. Table detection locates all tables within an input document. Preprocessing is performed prior to TSR, which recognizes the building blocks of a table: rows, columns, and individual cells. Table interpretation links the extracted structural elements with predefined semantic concepts. As a result, the layout and the semantic interpretation of a table is written in a structured format. Adapted from Namysł [43]

- We extend our table recognition approach by integrating a deep learning-based table detection module and adapting the TSR component from our previous work [22].
- We thoroughly evaluate the proposed method on two widely adopted table recognition benchmarks. Our method outperforms the baseline approach from our previous work [22] and performs on par with the state-of-the-art approaches in the field.
- We propose a fix of an issue with the evaluation script employed in the recent competition on table recognition and report the scores of our method in all scenarios that involve the original and the corrected script as well as the previously used and recently revised annotations.
- To facilitate reproducibility and fair comparison of the results obtained by different table recognition methods, we release the resources from our experiments and the evaluation script employed in our ICDAR 2013 experiment publicly.[7]

Table Extraction Task

Table extraction can be considered as a three-step process consisting of table detection, structure recognition, and interpretation (Fig. 2).

The goal of the table detection task is to locate all table regions within the input document. Subsequently, table structure recognition (TSR) aims to recognize the structure of each detected table. Note that both tasks can be performed on different input levels: text lines, words, characters, or pixels.[8] Moreover, although table detection and TSR aim to solve different problems, some approaches cover these two tasks jointly. In this case, we refer to joint table detection and structure recognition as the table recognition process.

Finally, the goal of table interpretation is to link the recognized cells with their semantic representation. This step strongly depends on the actual use case and no method fits all scenarios. In this work, this problem is formulated as maximum weight matching [5] on a graph with nodes that correspond to table cells and predefined semantic concepts.

In the following, the table detection, TSR, and table interpretation tasks, that are studied in this work, are described in more detail.

Table Detection

Table detection aims to locate all tables within an input document and can be considered a single-class object detection problem. Moreover, it can be split into two subtasks: (1) classify every input element, e.g., every pixel, as being part of a table or not (image segmentation) and (2) merge homogeneous input elements into distinct table regions (region growing and splitting).

In particular, region growing and splitting approaches make use of a heterogeneity criterion that specifies how similar two inputs are [10]. Specifically, keyword-based approaches look for specific words (like table or figure) and consider all elements within a specific distance to the keyword as being part of the same table region. In contrast, whitespace-based approaches detect large blank areas around the table and consider all enclosed pixels as a homogeneous table region [32].

Table detection can be performed on different input levels. For instance, on text line level, region growing and splitting becomes, geometrically, a one-dimensional problem. For a text line, one has to decide whether the lines above and below are similar enough or not to form a common table region.

Table Structure Recognition

During TSR, the structure of a table, i.e., rows, columns, and cells, is recognized.

[7] https://github.com/mnamysl/tabrec-sncs.

[8] Oro and Ruffolo [24] speak of so-called content elements that form a table.

Given a set of input elements E belonging to the table object, TSR aims to map these elements to a regular table grid. Formally, we are looking for a mapping:

$$\begin{aligned} \mu &: \{1, \dots, K\} \times \{1, \dots, N\} \longrightarrow \mathcal{P}(E) \\ \mu(i,j) &= E_{i,j} \subseteq E, \end{aligned} \tag{1}$$

which maps each position within a $K \times N$ table to a content element $e \in E$ or a set of multiple elements $E_{i,j} \subseteq E$, with $\mathcal{P}(E)$ denoting the power set of E.

In simple cases, one coordinate (i, j) is mapped to one single element e. Alternatively, elements can be merged at this step—multiple text lines to one text region—so that they collectively form a cell. Thus, μ generally points to a subset $E_{i,j}$ of elements in E.

To allow cells that span more than one row or column, it is valid that two neighboring coordinates $((i, j)$ and $(i, j+1))$ or $((i, j)$ and $(i+1, j))$ both point to the same element. A resulting table cell consists of all neighboring grid points mapped to the same element. Finally, it is also allowed that a grid point is empty and that μ points to an empty set.

Table Interpretation

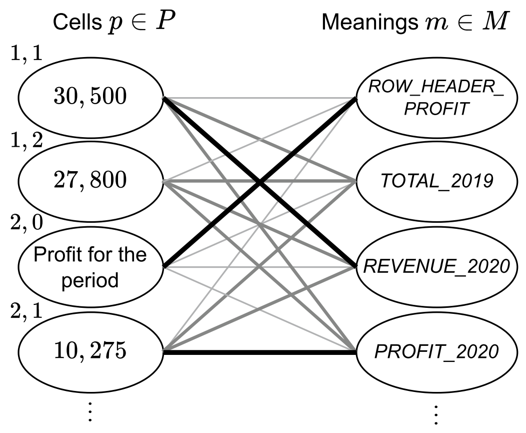
In the final table interpretation step, the semantic meaning of the table cells is understood. Formally, there exists a set of cells P and a set of meanings M, so that a cell $p \in P$ is mapped to a meaning $m \in M$.

The matching between cells P and meanings M is not necessarily a perfect matching. If a table contains additional columns that are not foreseen in the table model, the cells in these columns cannot be assigned a meaning. On the other hand, when the table model provides optional meanings, some of them cannot always be matched.

In Fig. 3, for instance, there exists a meaning REVENUE_2020, which specifies the revenue for the fiscal year 2020.[9] During the table interpretation step, one aims to map cell (1, 1) with the content 30,500 to this meaning.

	2020 *($'000s)*	**2019** *($'000s)*
Revenue	**30,500**	27,800
Profit for the period	**10,275**	6,900
Other comprehensive income	**1,125**	1,250
Total comprehensive income	**12,250**	8,633

(a) Balance sheet in tabular form



(b) A table interpretation graph constructed from Figure 3a

Fig. 3 Table interpretation example. **a** A financial statement (balance sheet in tabular form). **b** The corresponding table interpretation graph. Cells $p \in P$ are mapped to possible meanings $m \in M$. For each mapping, an affinity value is calculated, indicated by the thickness of the lines. Adapted from Namysł [43]

Related Work

Complete Table Recognition Approaches

In this section, we summarize recent approaches that perform complete table recognition (CTR). We describe both heuristic-based and learning-based approaches performing CTR. For a thorough review of the approaches formerly used for this task, please refer to a comprehensive review presented by Silva et al. [33].

Heuristic-Based CTR Methods

Heuristic-based methods were mainly designed to handle PDF files with embedded text. These methods perform fairly accurate, given that the format of the tables is compatible with the designed heuristics.

Hassan and Baumgartner [11] describe a system that extracts word boxes from PDF files, groups them into tables by analyzing their spatial features and ruling lines, if present, and outputs the identified tables in HTML format. It is worth noting that their system can detect cells that span multiple rows or columns as well as partially bordered and borderless tables.

Oro and Ruffolo [24] proposed PDF-TREX, a heuristic, bottom-up approach for table recognition in single-column PDF documents. To identify tabular arrangements of page elements, their method aligns and groups them by considering their spatial features, e.g., white spaces, the distribution of horizontal and vertical distances between the blocks, vertical overlapping ratios, etc. Their method obtains table cells from the intersections of rows and columns and is able to recognize multiple-line row and column headers.

Nurminen [23] developed the Tabler system that implements a set of heuristics for table detection and structure

[9] The meanings are denoted in capital letters.

recognition. Tabler takes PDF files with embedded text as input and outputs the recognized tables in a structured HTML or XML format. Tabler combines the information extracted directly from PDF files with raster image processing techniques.

Rastan et al. [28] proposed TEXUS, a task-based table extraction method from PDF documents with embedded text. To detect table positions, they locate table lines and use transitions between them and main text lines. To identify columns, they look for spatial alignments of text chunks inside the table region. Moreover, the rows are located by finding a dominant table line pattern. In addition, they also implemented functional and structural analysis components that are used to identify the role of each cell in a table and to detect the logical relationships between table cells, respectively.

Shigarov et al. [32] presented TabbyPDF, a heuristic-based method for table detection and structure recognition from PDF documents. Their system uses textual information and graphical features embedded into PDF files such as horizontal and vertical distances, font properties, and ruling lines. In addition, they propose to exploit the feature of the appearance of text printing instructions and the positions of a drawing cursor. Their system can detect borderless tables by exploiting ruling lines embedded in a PDF file. Alternatively, implicitly defined tables are recognized by analysis of white spaces between cells.

Learning-Based CTR Methods

Recently, many deep learning-based methods were proposed to solve the image-based table recognition problem. These approaches were often combined with heuristics implementing the missing functionality or used as postprocessing.

Schreiber et al. [31] proposed DeepDeSRT that employs the Faster R-CNN model for table row and column detection followed by a semantic segmentation approach for TSR. They fine-tune a general-purpose object detection model for the target task. Before structure recognition, they stretch the images vertically and horizontally to facilitate the separation of rows and columns by the model. Moreover, they apply postprocessing to fix problems with spurious detection fragments and conjoined regions.

Reza et al. [30] applied conditional generative adversarial networks for table localization and an encoder decoder-based architecture for TSR. Their detection model was trained from scratch using a large augmented data set composed of documents from both their private collection and publicly available sources. In contrast, their encoder-decoder architecture was initialized from a pretrained model and fine-tuned using the data annotated with table row and column positions.

Paliwal et al. [26] proposed TableNet, a multi-task, encoder-decoder architecture for table detection and structure recognition. They initialize the encoder using the weights of a general-purpose object detection model and share it between the table region detection and column segmentation decoders. Their model takes an image as input and produces two semantically labeled images for table and column regions, respectively. Subsequently, they use the output of the Tesseract OCR engine [34] to find table rows by locating words that are aligned horizontally.

Prasad et al. [27] described the CascadeTabNet model that uses the instance segmentation technique to detect table regions and that recognizes their structure in a single inference step. They use a CNN-based architecture and demonstrate effective use of transfer learning and image augmentation techniques. Their model additionally classifies tables into two classes: bordered and borderless types. Their model predicts the location of cells only for the borderless tables. In contrast, they employ rule-based text and line detection heuristics to extract the cells from bordered tables.

Inspired by the method proposed by Prasad et al. [27], Fischer et al. [6] presented Multi-Type-TD-TSR, a multi-stage end-to-end table recognition approach combining a deep learning-based table detection model with heuristic-based TSR. To improve the robustness of geometrical and pixel-level noise, they apply skew angle correction, noise filtering, and color normalization prior to the TSR method.

Table Interpretation Approaches

Table interpretation can be regarded as a strongly used case-specific task. Therefore, a variety of approaches from the area of natural language processing is used such as edit distance-based techniques [19] and RegEx matching algorithms [16], which are applied, e.g., for matching column titles or data types [38]. Semantic interpretation of the table content can also be performed using word embeddings or large, pretrained language models [12], relation extraction [21], or semantic parsing methods [39].

Semantic Type Detection

Semantic type detection is a related task that aims to find the correspondence between columns and real-world entities, e.g., locations, person names, and organizations. This task is often performed by using dictionary lookup and RegEx matching of column headers and values. There exists a large variety of data preparation and visualization tools that incorporate this method.[10]

[10] Popular data analysis tools: https://powerbi.microsoft.com, https://www.trifacta.com, https://datastudio.google.com.

A noteworthy deep learning-based approach was recently proposed by Hulsebos et al. [15]. Their method finds a correspondence between column headers of a table and 78 semantic types from a knowledge base [2]. Different features are employed to describe the content of a column, including distribution of characters, semantic meaning of the words, and global statistics like cardinality or uniqueness. In a follow-up study, Zhang et al. [40] proposed to additionally exploit the context of a column within a table to predict the underlying semantic types. Their hybrid machine learning model combines single-column type prediction with topic modeling and structured prediction techniques; thereby they achieve improvements in recognition accuracy in comparison to the baseline model.

Proposed Method

In Fig. 2, the architecture of the proposed information extraction system is presented. This section describes the components of this system in detail. In particular, "Table Detection" presents the table detection module. "Preprocessing" describes the preprocessing routines employed by our method. "Fully Bordered Tables" and "Partially Bordered and Borderless Tables" explain the proposed TSR methods. Finally, "Table Interpretation Method" details the proposed table interpretation approach. For a thorough description of the baseline system (Fig. 1), please refer to Namysl et al. [22].

Table Detection

Table detection aims to locate all tables within an input document ("Table Detection"). Recent advances in deep learning-based object recognition [36, 37] allow to perform a highly accurate and reliable detection process.

The main advantage of deep learning-based object detection methods is the possibility to apply the transfer learning technique, which gives us the ability to use the knowledge gained from learning one task to solve a related problem. In the case of deep learning-based table detection methods, an object detection model that was pretrained on a large, general-purpose object detection benchmark (usually on the ImageNet data set [4]) is fine-tuned using a smaller-scale data set for the target table detection task, e.g., TableBank [20].

Therefore, in this work, we exploit an existing deep learning-based table localization method and combine it with the TSR module from our previous work, resulting in an efficient, hybrid table recognition approach.

The detection method is required to take either an image or a PDF file as input and to return a list of bounding boxes, each corresponding to a single table object. The choice of method is rather arbitrary, as long as the aforementioned requirements are met.

Using the results provided by the table detection component, all identified tables are cropped from the original input document and passed to the preprocessing module as either images or PDF files, depending on the format of the original document.

Preprocessing

Preprocessing transforms the input document containing a single table object identified in the previous step into a semi-structured representation that is exploited by the subsequent components of our system. We employ the layout analysis module described in Konya [17] to extract ruling lines (hereafter referred to as solid separators) and textual page regions from an input document.

In particular, if the input document is in PDF format, it is rendered as an image. The input image is then binarized using the global thresholding method proposed by Otsu [25]. Subsequently, the solid separators are detected on the binary image using a combination of methods described by Zheng et al. [41] and Gatos et al. [8]. In the case of PDF files with embedded text, the text is directly extracted using a PDF parsing method.[11] Otherwise, OCR is performed using the Tesseract library [34] to extract the textual content from the image.

Table Structure Recognition

Although deep learning-based table detection has already reached very high accuracy on popular table recognition benchmarks, deep learning-based table structure recognition is still far from being solved [7]. Previous methods approached this problem by adapting the standard object detection framework to this task by detecting rows and columns independently. Postprocessing heuristics were then used to merge the results and output the final table grid (see "Learning-Based CTR Methods"). Obviously, these methods struggled with the recognition of table cells spanning multiple rows or columns. The recent advent of hybrid deep learning-based methods coupled with heuristics enabled surpassing the current state-of-the-art performance on widely adopted table recognition benchmarks [6, 27].

Motivated by the success of hybrid table recognition approaches, we combine a deep learning-based table detection module with a heuristic-based TSR method. This section describes the TSR methods employed in this work: heuristics for fully bordered tables ("Fully Bordered Tables")

[11] We employ Poppler (https://poppler.freedesktop.org) for both rendering and text extraction.

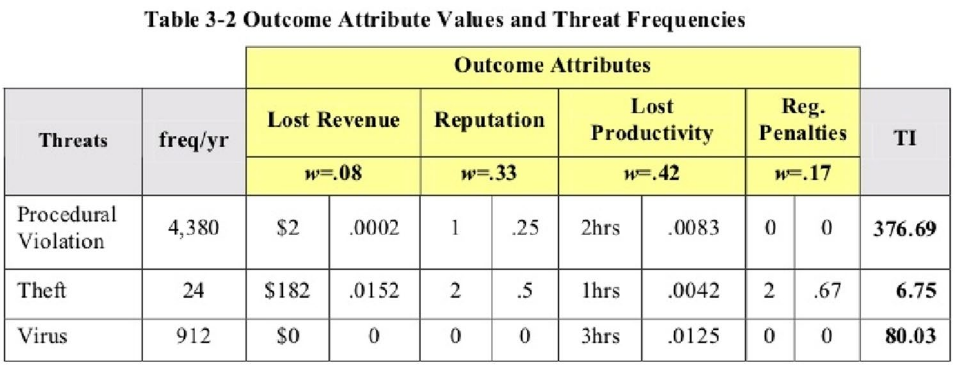


Table 3-2 Outcome Attribute Values and Threat Frequencies

Threats	freq/yr	Outcome Attributes								TI
		Lost Revenue		Reputation		Lost Productivity		Reg. Penalties		
		w=.08		*w*=.33		*w*=.42		*w*=.17		
Procedural Violation	4,380	$2	.0002	1	.25	2hrs	.0083	0	0	**376.69**
Theft	24	$182	.0152	2	.5	1hrs	.0042	2	.67	**6.75**
Virus	912	$0	0	0	0	3hrs	.0125	0	0	**80.03**

Fig. 4 An example of a fully bordered table. The image was cropped from the `cTDaR_t10047.jpg` file contained in the ICDAR 2019 Table Competition benchmark [7]. In this example, TI (threat index) indicates the relative significance of each threat and *w* is the attribute weight. TI of each type of an attack is computed by multiplying the threat frequency by the sum of the values in the right-hand columns under the outcome attributes weighted by the corresponding attribute weights. Adapted from Namysl et al. [22]

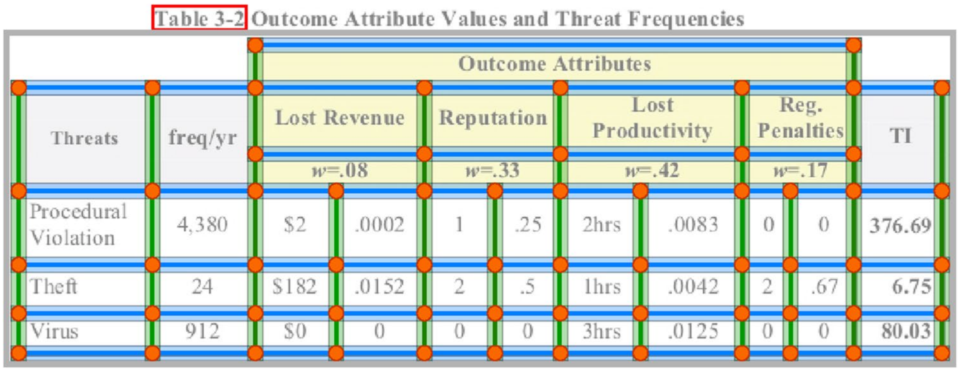


Fig. 5 Separator merging stage of the fully bordered TSR method. Vertical and horizontal separator regions are marked green and blue, respectively. Orange circles correspond to the intersection points. Adapted from Namysl et al. [22]

and for partially bordered or borderless tables ("Partially Bordered and Borderless Tables").

It is worth noting that the proposed TSR algorithms can be easily applied to both horizontal and vertical page layouts. For the sake of clarity, we describe how our method works in the case of the horizontal layout. For the vertical layout, all steps are identical, except that we swap the horizontal and the vertical separators with each other.

As a first step that is common for both proposed heuristics, we calculate an average character size within the input image, denoted as $\overline{S}_x$ and $\overline{S}_y$ for the width and height dimensions, respectively, using the semi-structured data provided by the preprocessing component. These values are then exploited in the subsequent steps of the proposed algorithm.

Fully Bordered Tables

Figure 4 shows an example of a fully bordered table, which is handled by the rule-based method described in this section.

Separator Merging

Our heuristic that recognizes fully bordered tables starts by sorting the horizontal and the vertical separators by the top and the left position, respectively. All separator boxes are first expanded by $\delta_x = \max(5, \overline{S}_x/2)$ and $\delta_y = \max(5, \overline{S}_y/2)$ pixels to increase the chance of intersection with the neighboring solid separators. Then, we iteratively merge all intersecting separators, forming clusters of separators, as depicted in Fig. 5. Finally, all clusters that contain less than one separator with each orientation (vertical and horizontal) are pruned from the list. The remaining, distinct separator clusters found by this procedure correspond to the identified table object.

Table Grid Estimation and Refinement

Subsequently, for each separator cluster, a rough grid of cells is derived as follows: Each pair of subsequent vertical and horizontal separators forms a table column or table row region, respectively. The regions of intersection between the column and row boxes define the rough grid of cells.

Note that some cells in the roughly estimated grid need to be refined by merging them with the neighboring cells to recover the cells that span multiple rows or columns. To this end, we employ an approach inspired by the union-find algorithm proposed by Hoshen and Kopelman [13] and illustrated in Fig. 6.

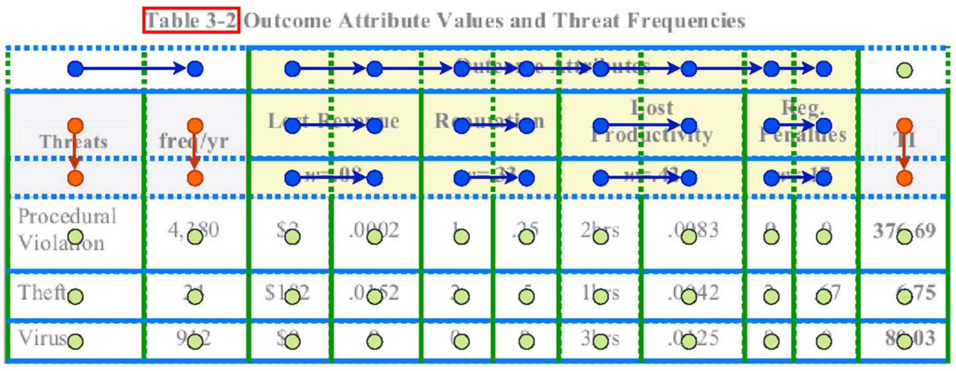


Fig. 6 Cell merging stage of the fully bordered TSR method. Blue and orange circles are the centers of the cells that were merged horizontally and vertically, respectively. Green circles are the centers of fully bordered cells. Arrows show the scanning direction. Adapted from Namysl et al. [22]

Specifically, we perform a raster scan through the rough grid of cells in the left-to-right direction. For each cell, we check whether the area near the right border of the cell overlaps any vertical separator assigned to the current separator cluster. If this is not the case, we merge the current cell with its right neighbor and proceed to the next cell. We use a margin around the border of a cell calculated as $m_x = \overline{S}_x$. This procedure is then repeated in the top-to-down direction. In this case, we use the margin $m_y = \overline{S}_y$. Note that the column spans of the cells that need to be merged must be equal.

Postprocessing

During the post-processing phase, all textual page regions are assigned to the corresponding table cells based on their overlap ratios; cells that do not contain any assigned page regions are marked as empty. Subsequently, the rows and columns that contain exclusively empty cells are removed from the table.

In an additional step, those tables are identified that predominantly exhibit a bordered layout but also contain many rows that are separated by white spaces instead of solid separators. An example of such a table is presented in Fig. 7. As such tables would preferably be recognized using our heuristic for partially bordered tables, we give an option to discard them from the list of candidates found by the heuristic for fully bordered tables. For each table, we calculate the ratio H_{ratio} of the highest row to the median row height. We discard a table if its H_{ratio} is greater than a predefined threshold $H^{\text{max}}_{\text{ratio}}$.

Finally, all table candidates that have less than a predefined number of rows, columns, and cells are pruned from the list of candidates. Figure 8 shows an example of a table recognized by our heuristic for bordered tables.

Partially Bordered and Borderless Tables

Booktabs[12] is a popular LaTeX package used to typeset tables in scientific articles. An example of a table in this format is shown in Fig. 9. It consists of three main components: top, middle, and bottom rule. The middle rule separates the table header and the table body region. In addition, multiple-level header structure can be represented using shorter cmidrules that span multiple columns aggregated under the same higher-level header (see Fig. 15a).

Our heuristic recognition method for partially bordered and borderless tables uses horizontal separators for documents with standard orientation. As noted in "Fully Bordered Tables", the pages with vertical orientation can be easily handled by swapping horizontal and vertical separators with each other.

Table Region Detection

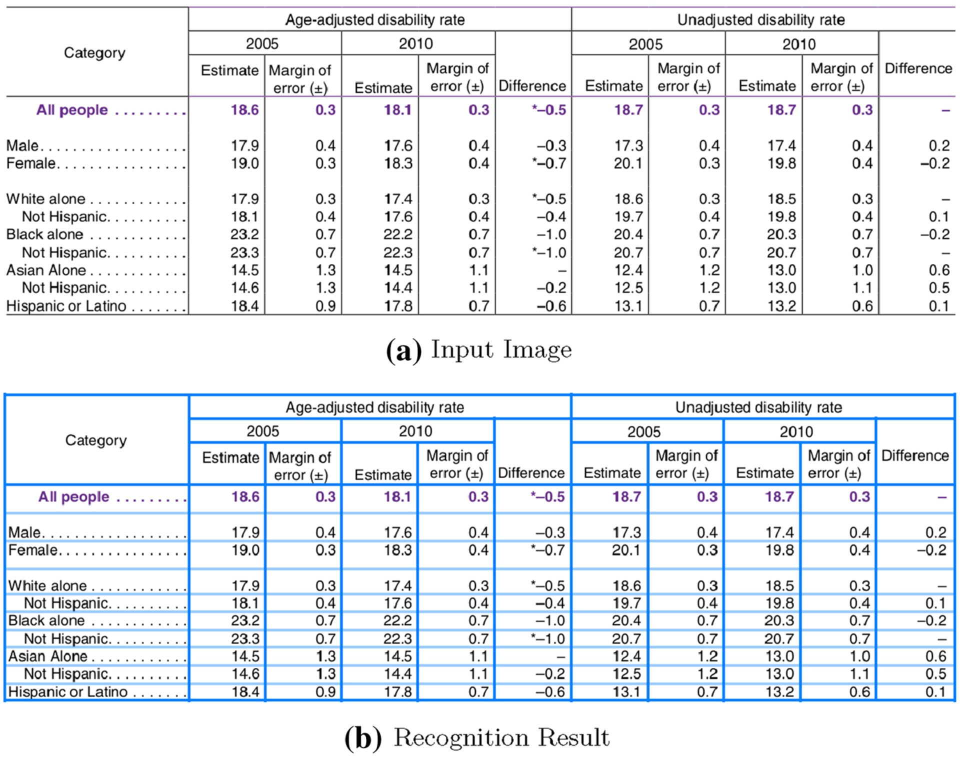
In the first step, we perform separator filtering. Specifically, we discard all thick lines wider than $\overline{S}_y$ and sort all remaining separators by the top position. Moreover, if multiple separators are located within the margin of $D = 2\overline{S}_y$, only the element with the lowest y-position is kept.

Given the above-described filtering routines and assuming that the input image contains a single table object, we can add virtual top and bottom rule lines at the top and the bottom of the image, respectively, without negatively influencing the recognition process. In this method, we stretch these ruling lines across the entire width of the image. This step should improve the results in case tables do not strictly follow the booktabs format, e.g., by missing a top or a bottom rule.

To detect table objects, we search for triples of consecutive separators, for which the difference between their left and right coordinates is lower than $\overline{S}_x$. Each triple forms a table candidate that is passed to the subsequent filtering step, where all candidates narrower than 90% of the image width are discarded.

If this process does not output any valid table candidate, we discard all spurious candidates and proceed as follows: We add an additional virtual ruling line at the position $-\overline{S}_y$, and we repeat the searching process described in the previous paragraph. This step ensures that at least one table candidate is found, even in the case of borderless tables. Figure 10 illustrates a case that benefits from adding virtual ruling lines and filtering narrow tables.

[12] https://ctan.org/pkg/booktabs.


(a) Input Image
(b) Recognition Result

Category	Age-adjusted disability rate					Unadjusted disability rate				
	2005		2010			2005		2010		
	Estimate	Margin of error (±)	Estimate	Margin of error (±)	Difference	Estimate	Margin of error (±)	Estimate	Margin of error (±)	Difference
All people	**18.6**	**0.3**	**18.1**	**0.3**	***–0.5**	**18.7**	**0.3**	**18.7**	**0.3**	**–**
Male.	17.9	0.4	17.6	0.4	–0.3	17.3	0.4	17.4	0.4	0.2
Female.	19.0	0.3	18.3	0.4	*–0.7	20.1	0.3	19.8	0.4	–0.2
White alone	17.9	0.3	17.4	0.3	*–0.5	18.6	0.3	18.5	0.3	–
Not Hispanic.	18.1	0.4	17.6	0.4	–0.4	19.7	0.4	19.8	0.4	0.1
Black alone	23.2	0.7	22.2	0.7	–1.0	20.4	0.7	20.3	0.7	–0.2
Not Hispanic.	23.3	0.7	22.3	0.7	*–1.0	20.7	0.7	20.7	0.7	–
Asian Alone	14.5	1.3	14.5	1.1	–	12.4	1.2	13.0	1.0	0.6
Not Hispanic.	14.6	1.3	14.4	1.1	–0.2	12.5	1.2	13.0	1.1	0.5
Hispanic or Latino	18.4	0.9	17.8	0.7	–0.6	13.1	0.7	13.2	0.6	0.1

Category	Age-adjusted disability rate					Unadjusted disability rate				
	2005		2010			2005		2010		
	Estimate	Margin of error (±)	Estimate	Margin of error (±)	Difference	Estimate	Margin of error (±)	Estimate	Margin of error (±)	Difference
All people	**18.6**	**0.3**	**18.1**	**0.3**	***–0.5**	**18.7**	**0.3**	**18.7**	**0.3**	**–**
Male.	17.9	0.4	17.6	0.4	–0.3	17.3	0.4	17.4	0.4	0.2
Female.	19.0	0.3	18.3	0.4	*–0.7	20.1	0.3	19.8	0.4	–0.2
White alone	17.9	0.3	17.4	0.3	*–0.5	18.6	0.3	18.5	0.3	–
Not Hispanic.	18.1	0.4	17.6	0.4	–0.4	19.7	0.4	19.8	0.4	0.1
Black alone	23.2	0.7	22.2	0.7	–1.0	20.4	0.7	20.3	0.7	–0.2
Not Hispanic.	23.3	0.7	22.3	0.7	*–1.0	20.7	0.7	20.7	0.7	–
Asian Alone	14.5	1.3	14.5	1.1	–	12.4	1.2	13.0	1.0	0.6
Not Hispanic.	14.6	1.3	14.4	1.1	–0.2	12.5	1.2	13.0	1.1	0.5
Hispanic or Latino	18.4	0.9	17.8	0.7	–0.6	13.1	0.7	13.2	0.6	0.1

Fig. 7 An example of a table cropped from the `us-001.jpg` file contained in the ICDAR 2013 Table Competition benchmark [9]. Solid blue lines represent the borders between the cells that were detected by the TSR heuristic for fully bordered tables. In contrast, light blue lines correspond to the row borders that are not outlined with solid ruling lines and therefore could not be recognized by this method. Adapted from Namysł [43]

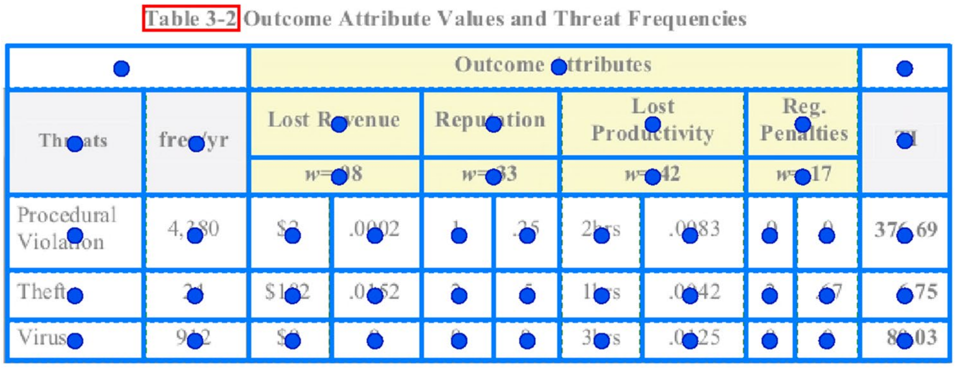
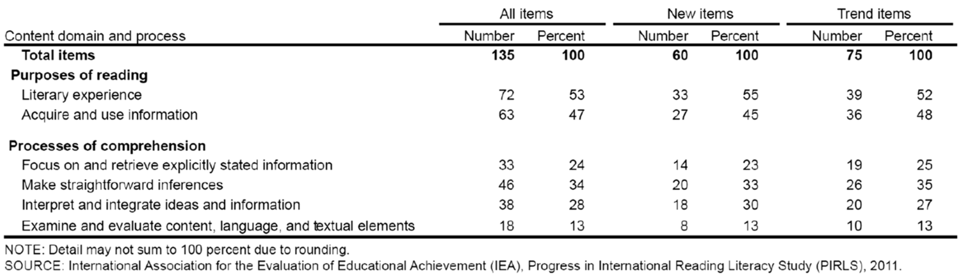


Table 3-2 Outcome Attribute Values and Threat Frequencies

		Outcome Attributes								
Threats	freq/yr	Lost Revenue		Reputation		Lost Productivity		Reg. Penalties		TI
		w=.08		w=.33		w=.42		w=.17		
Procedural Violation	4,380	$2	.0002	1	.25	2hrs	.0083	0	0	376.69
Theft	24	$182	.0152	[illegible]	5	1hrs	.0042	[illegible]	.67	6.75
Virus	912	$0	0	0	0	3hrs	.0125	0	0	80.03

Fig. 8 Recognition result obtained by the bordered TSR method. Blue circles represent the centers of the recognized cells. Adapted from Namysl et al. [22]



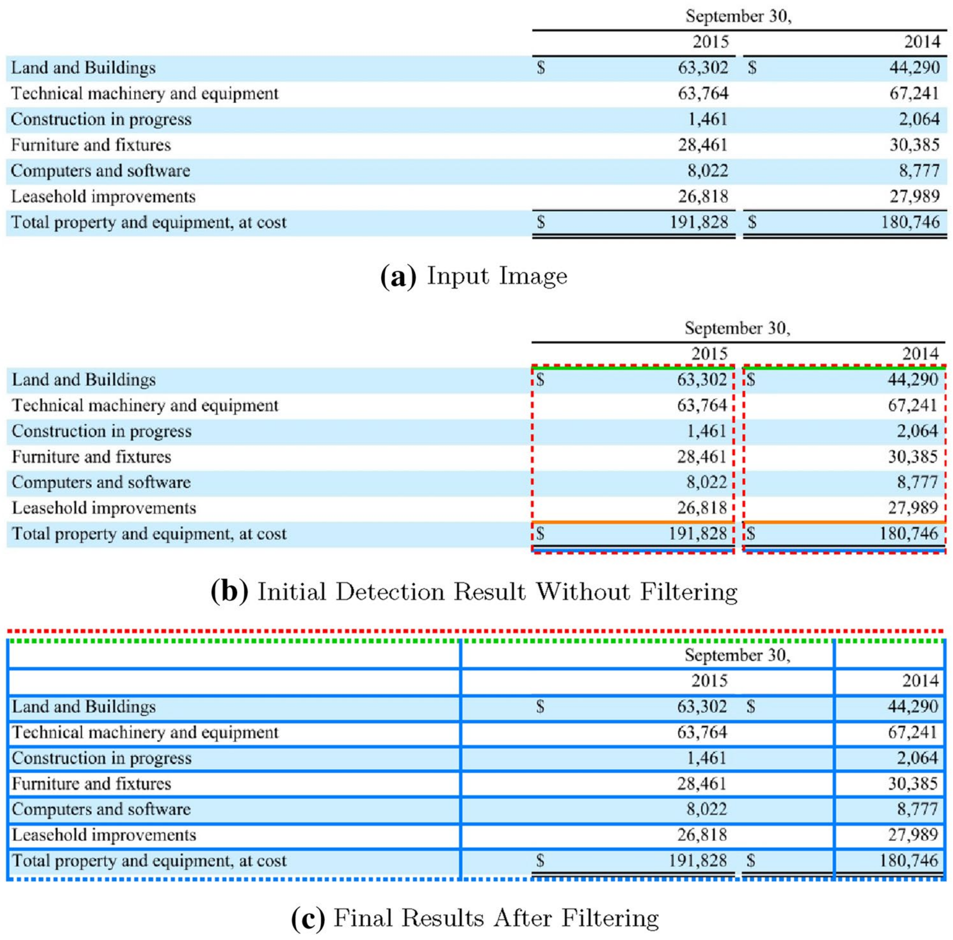
Content domain and process	All items		New items		Trend items	
	Number	Percent	Number	Percent	Number	Percent
Total items	**135**	**100**	**60**	**100**	**75**	**100**
Purposes of reading						
Literary experience	72	53	33	55	39	52
Acquire and use information	63	47	27	45	36	48
Processes of comprehension						
Focus on and retrieve explicitly stated information	33	24	14	23	19	25
Make straightforward inferences	46	34	20	33	26	35
Interpret and integrate ideas and information	38	28	18	30	20	27
Examine and evaluate content, language, and textual elements	18	13	8	13	10	13

NOTE: Detail may not sum to 100 percent due to rounding.
SOURCE: International Association for the Evaluation of Educational Achievement (IEA), Progress in International Reading Literacy Study (PIRLS), 2011.

Fig. 9 An example of a table in booktabs format from the `us-021.pdf` file contained in the ICDAR 2013 Table Competition benchmark [9]. Adapted from Namysl et al. [22]

Merging Overlapping Table Candidates

Moreover, in the case of a tabular layout that uses solid separators for the separation of rows, the above-described method will output multiple overlapping candidates. Therefore, if the vertical overlap between two table candidates is greater than $\overline{S}_y$, these candidates are merged together, i.e., the top and the middle rule with a lower y-position as well as the bottom rule with a higher y-position are retained in the merged table candidate, as illustrated in Fig. 11.


(a) Input Image
(b) Initial Detection Result Without Filtering
(c) Final Results After Filtering

	September 30,			
	2015		2014	
Land and Buildings	$	63,302	$	44,290
Technical machinery and equipment		63,764		67,241
Construction in progress		1,461		2,064
Furniture and fixtures		28,461		30,385
Computers and software		8,022		8,777
Leasehold improvements		26,818		27,989
Total property and equipment, at cost	$	191,828	$	180,746

Fig. 10 Illustration of the filtering based on the table width employed by the TSR heuristic for partially bordered and borderless tables. **a** An example of a table cropped from the `cTDaR_t10005.jpg` file contained in the ICDAR 2019 Table Competition benchmark [7]. **b** An initial result before the filtering: two spurious candidates were identified. Green, orange, and blue lines correspond to the top, middle, and bottom rule lines, respectively. **c** The result after filtering. Dotted green and blue lines correspond to the virtual top and bottom ruling lines, respectively. Dotted red line is the virtual ruling line added above the top ruling line. Note that the row between two virtual ruling lines at the top is discarded as it does not contain any textual content. Adapted from Namysł [43]

Finally, for each valid table candidate, we collect all cmidrule lines that are located between the top and the middle rule, so they can be used to recognize a multiple-level header structure in the subsequent processing step. To this end, they are grouped by their y-position to isolate different levels of the header's hierarchy and to separate header rows.

Table Row and Column Detection

The borders for the rows in the body region are determined using the horizontal profile, which is calculated by projecting all words within the body region of a table, as illustrated in Fig. 12. The row borders can then be easily estimated by taking center positions of the gaps in the resulting profile.

To recognize borders between columns, we first project all page regions within the body region and the lowest-level header row vertically and we analyze the resulting projection to find all gaps with a length above $\mathcal{D}_{\text{column}} = \gamma \overline{S}_x$, where γ is a hyperparameter. The center positions of these intervals correspond to the column borders, as illustrated in Fig. 13.

In contrast, all gaps with length below $\mathcal{D}_{\text{column}}$ correspond to vertically aligned words that form spurious columns. Note that we exclude the higher-level headers, as they contain multiple-column cells that would otherwise distort the calculated vertical projection.

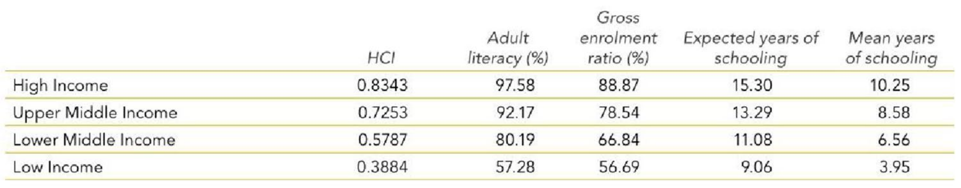
Table Grid Estimation and Refinement

Given the row and column borders calculated in the previous stages, we compute the grid of cells from the intersections between the row and the column borders, which results in a partial table segmentation illustrated in Fig. 14.

Moreover, the structure of the remaining, higher-level headers is recognized as follows: The rough grid of cells calculated in the previous step is extended to the higher-level headers and all cells that intersect the same cmidrule segment are merged together, as illustrated in Fig. 15.

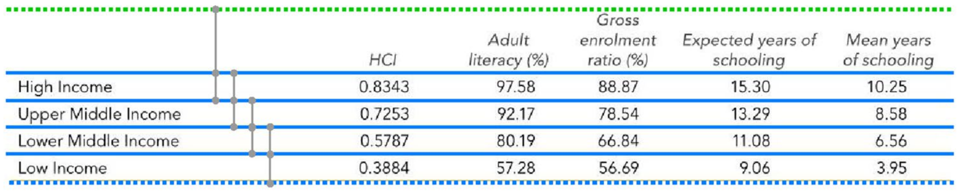
Postprocessing

Finally, all textual page regions are assigned to the corresponding table cells based on their overlap ratios and the cells that do not contain any assigned page regions are marked as empty. Subsequently, the rows and columns that contain exclusively empty cells are removed from the table. Moreover, all table candidates that have less than a predefined number of rows, columns, and cells are pruned from the list of candidates.



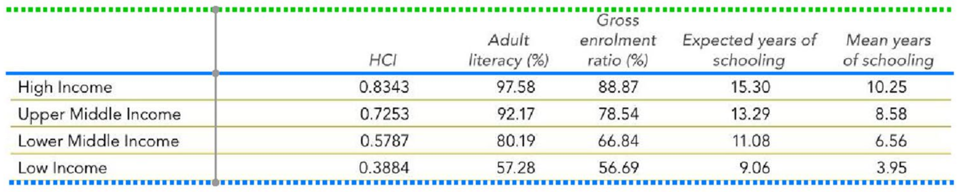
	HCI	Adult literacy (%)	Gross enrolment ratio (%)	Expected years of schooling	Mean years of schooling
High Income	0.8343	97.58	88.87	15.30	10.25
Upper Middle Income	0.7253	92.17	78.54	13.29	8.58
Lower Middle Income	0.5787	80.19	66.84	11.08	6.56
Low Income	0.3884	57.28	56.69	9.06	3.95

(a) Input Image



	HCI	Adult literacy (%)	Gross enrolment ratio (%)	Expected years of schooling	Mean years of schooling
High Income	0.8343	97.58	88.87	15.30	10.25
Upper Middle Income	0.7253	92.17	78.54	13.29	8.58
Lower Middle Income	0.5787	80.19	66.84	11.08	6.56
Low Income	0.3884	57.28	56.69	9.06	3.95

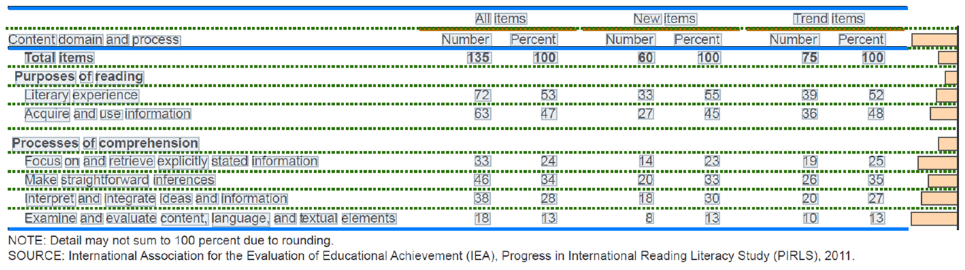
(b) Table Candidates Before Merging



	HCI	Adult literacy (%)	Gross enrolment ratio (%)	Expected years of schooling	Mean years of schooling
High Income	0.8343	97.58	88.87	15.30	10.25
Upper Middle Income	0.7253	92.17	78.54	13.29	8.58
Lower Middle Income	0.5787	80.19	66.84	11.08	6.56
Low Income	0.3884	57.28	56.69	9.06	3.95

(c) Table Candidates After Merging

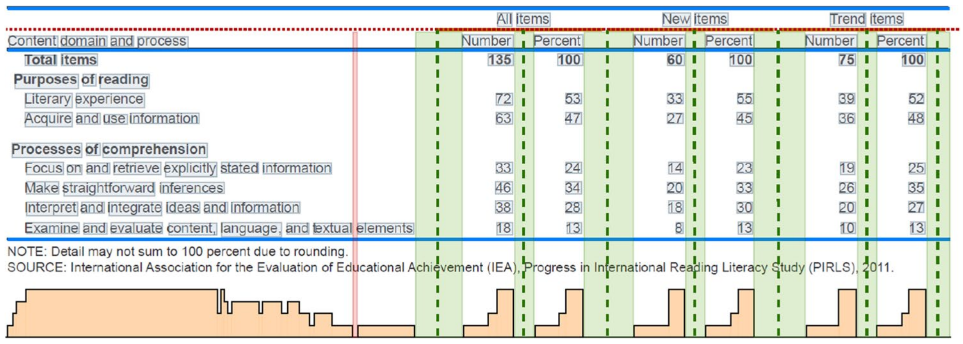
Fig. 11 Merging overlapping table candidates, as employed by the TSR heuristic for partially bordered and borderless tables. **a** An example of a table cropped from the `cTDaR_t10058.jpg` file contained in the ICDAR 2019 Table Competition benchmark [7]. **b** and **c** Overlapping tables. Each triple of consecutive separators, marked with gray lines, represents one table candidate. Using solid lines as row separators causes that a common line is included in the subsequent candidates. Merging the overlapping elements allows us to mitigate this problem. Dotted green and blue lines correspond to virtual top and bottom rules, respectively. Solid blue lines represent the remaining solid separators. Adapted from Namysł [43]



	All items		New items		Trend items	
Content domain and process	Number	Percent	Number	Percent	Number	Percent
Total items	135	100	60	100	75	100
Purposes of reading						
Literary experience	72	53	33	55	39	52
Acquire and use information	63	47	27	45	36	48
Processes of comprehension						
Focus on and retrieve explicitly stated information	33	24	14	23	19	25
Make straightforward inferences	46	34	20	33	26	35
Interpret and integrate ideas and information	38	28	18	30	20	27
Examine and evaluate content, language, and textual elements	18	13	8	13	10	13

NOTE: Detail may not sum to 100 percent due to rounding.
SOURCE: International Association for the Evaluation of Educational Achievement (IEA), Progress in International Reading Literacy Study (PIRLS), 2011.

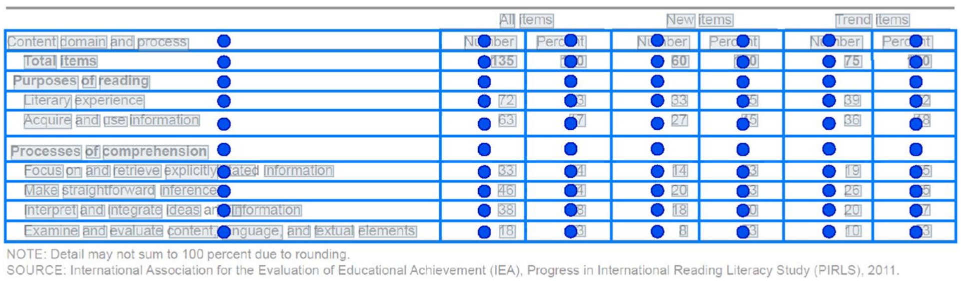
Fig. 12 Row segmentation process employed by the proposed TSR method for partially bordered and borderless tables. Blue lines represent the top, middle, and bottom ruling lines. Orange lines mark the cmidrule lines. Orange bars to the right correspond to the horizontal profile (running sum of pixels in the text regions in each row). Green dotted lines correspond to the row borders. Adapted from Namysl et al. [22]



	All items		New items		Trend items	
Content domain and process	Number	Percent	Number	Percent	Number	Percent
Total items	135	100	60	100	75	100
Purposes of reading						
Literary experience	72	53	33	55	39	52
Acquire and use information	63	47	27	45	36	48
Processes of comprehension						
Focus on and retrieve explicitly stated information	33	24	14	23	19	25
Make straightforward inferences	46	34	20	33	26	35
Interpret and integrate ideas and information	38	28	18	30	20	27
Examine and evaluate content, language, and textual elements	18	13	8	13	10	13

NOTE: Detail may not sum to 100 percent due to rounding.
SOURCE: International Association for the Evaluation of Educational Achievement (IEA), Progress in International Reading Literacy Study (PIRLS), 2011.

Fig. 13 Column segmentation process employed by the proposed TSR method for partially bordered and borderless tables. The dotted red line is a border of the lowest-level header. Orange bars at the bottom correspond to the vertical profile (running sum of pixels in the word regions in each column). We clip the values in the profile for better visualization. The column gaps that are wider and narrower than $\mathcal{D}_{\text{column}}$ are highlighted in green and red, respectively. Green vertical dotted lines represent the detected column borders. Adapted from Namysl et al. [22]



	All items		New items		Trend items	
Content domain and process	Number	Percent	Number	Percent	Number	Percent
Total items	135	[illegible]	60	[illegible]	75	[illegible]
Purposes of reading						
Literary experience	72	[illegible]	33	[illegible]	39	[illegible]
Acquire and use information	63	[illegible]	27	[illegible]	36	[illegible]
Processes of comprehension						
Focus on and retrieve explicitly stated information	33	[illegible]	14	[illegible]	19	[illegible]
Make straightforward inferences	46	[illegible]	20	[illegible]	26	[illegible]
Interpret and integrate ideas and information	38	[illegible]	18	[illegible]	20	[illegible]
Examine and evaluate content, language, and textual elements	18	[illegible]	8	[illegible]	10	[illegible]

NOTE: Detail may not sum to 100 percent due to rounding.
SOURCE: International Association for the Evaluation of Educational Achievement (IEA), Progress in International Reading Literacy Study (PIRLS), 2011.

Fig. 14 The resulting segmentation grid obtained by the proposed TSR method for partially bordered and borderless tables. Blue lines and circles are the borders and the centers of the cells, respectively. Gray boxes outline the words within the table. Adapted from Namysl et al. [22]

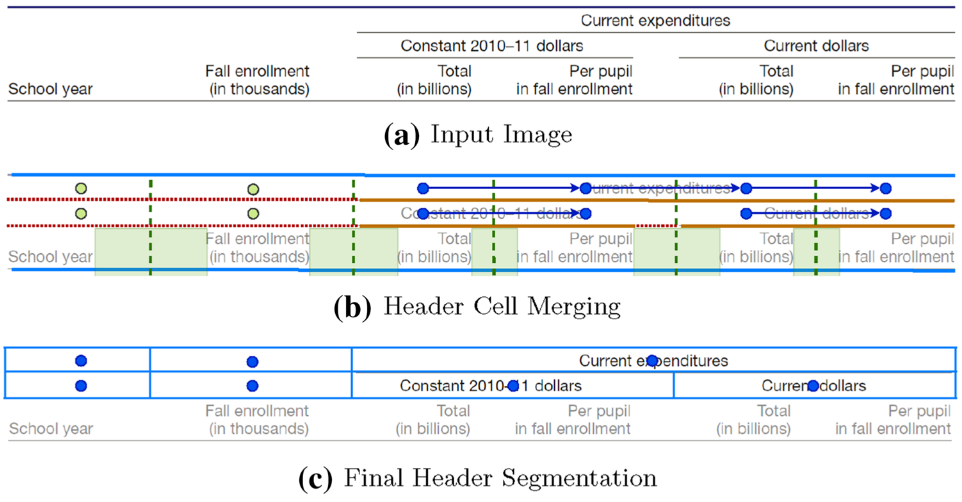


Fig. 15 Higher-level header segmentation of the proposed TSR method for partially bordered and borderless tables. **a** The top part of a table extracted from the `us-018.pdf` file from the ICDAR 2013 Table Competition benchmark [9]. **b** Header cell merging. Orange lines correspond to the cmidrule lines. Green areas and lines represent column white spaces and borders, respectively. Blue circles are the centers of the cells intersecting a cmidrule line. The cells that intersect the same cmidrule line are merged. In contrast, other cells (marked with green circles) remain unchanged. **c** Header segmentation. Blue lines and circles correspond to the borders and the centers of the cells in the final grid, respectively. Adapted from Namysl et al. [22]

Table Interpretation Method

Instead of matching the cells $p \in P$ to the meanings $m \in M$ directly, as foreseen by the general formulation of the table interpretation task presented in "Table Interpretation", the proposed table interpretation method first assigns meanings to the columns $c \in C$ of a table t. Subsequently, for a column c that was matched with a meaning m_j, it extracts the tuples $x_{i,j}$ by associating the cells in the body part of the column c with the meaning m_j, where i is a row index, and j is the index of a matched meaning.

Our algorithm takes the set of recognized tables T as input and, for each table $t \in T$, it assigns meanings $m \in M$ to the columns $c \in C$. We define a set of affinity rules that describe a column that is likely to be matched with the meaning m that includes:

(1) *Title Keyword Score*: implemented as approximate string matching between the title of a column and the predefined keywords.
(2) *Title RegEx Score*: computed as exact matching of the title of a column with customized RegEx.
(3) *Data Type Score*: computed as exact matching of the content of the cells in a column with RegEx for some common types (e.g., integer, date, etc.).
(4) *Content RegEx Score*: implemented as exact matching of the content of the cells in a column with customized RegEx.

Approximate string matching corresponds to the Levenshtein distance [19] calculated between two strings and divided by the length of the longer string. The exact RegEx score returns 1.0 if the matching succeeds and 0.0 otherwise. Moreover, we average the values of the content and data type scores over the cells in the corresponding column. We compute the final affinity score S for a column c with a meaning m by:

$$S(c,m) = \frac{w_c \max\left(S_c^{Rx}, S_c^{DT}\right) + w_t \max\left(S_t^{Rx}, S_t^{KW}\right)}{w_c + w_t}, \qquad (2)$$

```
[
  {
    "id": "compound",
    "keywords": ["compound", "name", "sample"],
    "titleRegex": "",
    "datatype": "string",
    "weightTitle": 0.9,
    "weightContent": 0.1,
    "minAffinityScore": 0.5
  },
  {
    "id": "hdac6_gene",
    "keywords": ["HDAC6"],
    "titleRegex": "^HDAC[-]{0,1}6[^\\d].*$",
    "datatype": ["double", "range", "integer"],
    "weightTitle": 0.7,
    "weightContent": 0.3,
    "minAffinityScore": 0.85
  }
]
```

Fig. 16 An example of a configuration file used by the proposed table interpretation method. It defines the meanings COMPOUND and HDAC6 GENE, as well as the rules for matching table columns to these meanings. The file is stored in JSON format. Adapted from Namysl et al. [22]

where w_t and w_c are the weights of the title and the content scores, respectively, S_c^{Rx} and S_c^{DT} are the affinity scores of the content RegEx and the data type, respectively. Moreover, S_t^{Rx} and S_t^{KW} are the scores of the title RegEx and the approximate matching with the keywords, respectively. In Eq. (2), the sum of weights must be a positive number. Note that, if a particular rule is not defined for a meaning m, the corresponding score is set to zero. All rules are defined in a configuration file, as presented in an example in Fig. 16.

Given a list of recognized tables and a set of predefined meanings, we perform the matching between the meanings and the columns in each table. To this end, we construct a weighted bipartite graph with two sets of vertices, each representing the meanings on one side and the columns on the other side, as illustrated in Fig. 17b. We link each column with each meaning with an edge weighted by the affinity score that specifies how likely a column matches with a certain meaning. To improve performance, we prune the connections that do not reach a predefined required minimum affinity value S_{min}.

Subsequently, we perform maximum weight matching, as defined by Edmonds [5], on the created bipartite graph to find the best assignment of the columns to the meanings. Finally, we extract the tuples $x_{i,j}$, where i is a row index and j is the index of a meaning, as shown in Fig. 17c.

Table Recognition Experiments

To evaluate our method we perform extensive experiments on two widely adopted table recognition benchmarks. In both cases, we evaluate the complete table recognition (CTR) process, i.e., end-to-end table detection and recognition.

Data Sets

The data set used in the ICDAR 2013 Table Competition by Göbel et al. [9] contains born-digital business and government PDF documents with 156 tables in total. Ground-truth annotations for both table detection and segmentation tasks are available.

The ICDAR 2019 Table Detection and Recognition data set by Gao et al. [7] is a collection of modern and archival document images. We employed only the former part, as the latter consists of handwritten documents and the analysis of hand-drawn tables is outside the scope of this work. We focused on track B2 in this competition as it corresponds to the CTR process.

Table Detection Setup

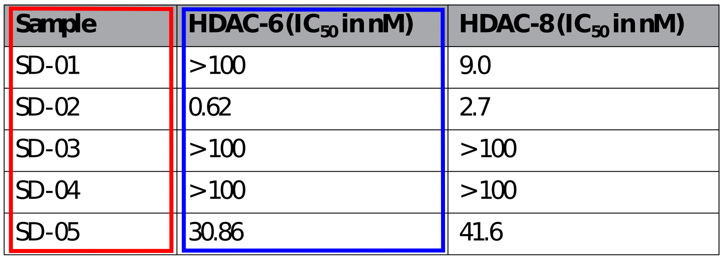
In the case of the ICDAR 2013 data set, all pages of a PDF document are first rendered as images with a resolution of 300 DPI and the detection is performed for each rendered image separately. In the case of the ICDAR 2019 benchmark, the original images are used as input to the detection model.

Table Detection Models

In this work, we combine our TSR method with two previously released table detection models. Nevertheless, other models can readily be used instead ("Table Detection").

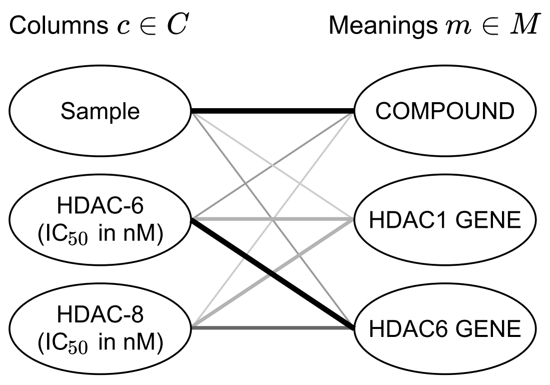
The first variant is the table detection model released by Prasad et al. [27]. Their CascadeTabNet model uses an instance segmentation technique and performs pixel-level table identification. In the experiment in "ICDAR 2013 Evaluation", we use the model fine-tuned on the ICDAR 2013 benchmark and in "ICDAR 2019 Evaluation" we employ the model tuned on the ICDAR 2019 data set. Please refer to Prasad et al. [27] for details about the architecture of the table detection model, the composition of the data used for training, and the employed training setup. Hereinafter we refer to the variant of our system that employs this model as the domain-specific table detection model.

In contrast, the second variant of our system employs the table detection model proposed by Li et al. [20], which is based on the Faster R-CNN architecture [29] with the ResNeXt-152 model as backbone [37]. Their model was pretrained on the ImageNet data set [4] and fine-tuned on the TableBank data [20], which contains a large number of Word and LaTeX documents crawled from the internet. Note that this model was not fine-tuned on the examples from the benchmarks employed for evaluation. Therefore, we refer to this model as a general-purpose table detection model.

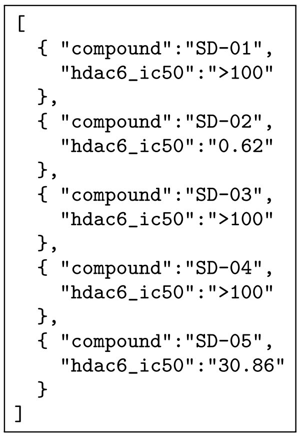


Sample	HDAC-6 (IC_{50} in nM)	HDAC-8 (IC_{50} in nM)
SD-01	>100	9.0
SD-02	0.62	2.7
SD-03	>100	>100
SD-04	>100	>100
SD-05	30.86	41.6

(a) Input Table. Adapted from Debnath et al. [3]



(b) Interpretation Graph. Adapted from Namysl et al. [22]



```
[
  { "compound":"SD-01",
    "hdac6_ic50":">100"
  },
  { "compound":"SD-02",
    "hdac6_ic50":"0.62"
  },
  { "compound":"SD-03",
    "hdac6_ic50":">100"
  },
  { "compound":"SD-04",
    "hdac6_ic50":">100"
  },
  { "compound":"SD-05",
    "hdac6_ic50":"30.86"
  }
]
```

(c) Extracted Tuples

Fig. 17 Illustration of the proposed table interpretation method: **a** An image of a table containing the inhibitory activity of some representative compounds toward the HDAC gene. The columns corresponding to the meanings COMPOUND and HDAC6 GENE (see Fig. 16) are marked with red and blue boxes, respectively. **c** Table interpretation graph: Columns $c \in C$ are mapped to the meanings $m \in M$. For each mapping, an affinity value is calculated, symbolized by the thickness of the lines. **d** The extracted tuples that represent the inhibitory activity of each compound towards the HDAC6 gene. The resulting file is stored in JSON format

Filtering Rules

Note that the employed table detection method takes an image as input and returns, for each detected table, a bounding box and a confidence value. We keep all detection results with a confidence greater than or equal to 0.85 and to 0.1, respectively, in the case of the domain-specific and the general-purpose table detector. Moreover, if some detection results overlap with each other by more than 50%, we keep only the result with a higher confidence value.

TSR Setup

All detected tables are cropped from the input documents based on the returned bounding box coordinates and fed to the preprocessing module, followed by the TSR component, one table at a time. In the case that the input document is in PDF format, we use the PyPDF2 library[13] to crop a region from a PDF file. Therefore, the preprocessing module can extract the text embedded in the PDF files, which is essential to obtain competitive results on the ICDAR 2013 benchmark.

The Order of Applying TSR Heuristics

Regarding the TSR heuristics, it is worth noting that a table candidate that overlaps any other table that was already detected by the previous heuristic is automatically discarded. Therefore, the order in which we apply our methods impacts the final results. As the heuristic for partially bordered and borderless tables could generate spurious candidates from fully bordered tables, we first apply the method for fully bordered tables followed by the other heuristic.

Hyperparameters

In Table 1, we present the hyperparameter values of the TSR methods used in the experiments. We empirically estimated these values based on the results on a practice data set from ICDAR 2013 Table Competition that consists of 58 PDF documents and the data from the remaining tracks in ICDAR 2019 Table Competition.

Postprocessing

In the case of the ICDAR 2019 benchmark, the results for all tables on a page are gathered to produce the output XML

[13] https://pypi.org/project/PyPDF2.

Table 1 Values of the hyperparameters used in the experiments

Name	ICDAR 2013	ICDAR 2019
Min. number of rows	2	2
Min. number of columns	2	2
Min. number of cells	7	7
H^{max}_{ratio} threshold	10.0	not used
Column gap threshold (γ)	1.5	1.0

We present the values used both in the experiment on the ICDAR 2013 ("ICDAR 2013 Evaluation") and the ICDAR 2019 ("ICDAR 2019 Evaluation") benchmark. Note that, in the case of ICDAR 2019, we do not discard table candidates based on H_{ratio}

file in a format[14] that is supported by this competition. Similarly, in the ICDAR 2013 setup, the results of TSR from all pages are gathered to produce the final XML file in a format[15] exploited by the evaluation tools employed in this competition.

ICDAR 2013 Evaluation

In this section, we present the results obtained by the presented approach on the table recognition benchmark from ICDAR 2013 Table Competition.[16]

Evaluation Setup

We developed a Python wrapper for the competition's evaluation tool[17] that computes the document-level metrics—precision, recall, and F_1 score—which were used to compare the accuracy of the examined methods. Our script parses the output produced by the official evaluation tool written in Java programming language and accumulates them to produce the final, per document average scores. It also includes the adjacency relations from the false-positively detected tables to give a better perspective on the actual performance of the table recognition approaches and utilizes alternative ground-truth annotations prepared by the organizers for several documents in this data sets. To facilitate reproducibility and fair comparison of research results, we include our script in a public repository associated with this paper.[18]

[14] https://cndplab-founder.github.io/cTDaR2019/dataset-description.html.

[15] https://roundtrippdf.com/en/data-extraction/dataset-format.

[16] https://roundtrippdf.com/en/data-extraction/icdar-2013-table-competition.

[17] https://roundtrippdf.com/en/data-extraction/table-recognition-dataset-tools.

[18] https://github.com/mnamysl/tabrec-sncs.

Table 2 Evaluation results on the ICDAR 2013 benchmark. We report the precision, recall, and F_1 score (per document averages) for the CTR process

Method	Precision	Recall	F_1
FineReader v11 [9]	0.8710	**0.8835**	**0.8772**
This work[a]	0.8714	0.8468	0.8589
This work[b]	0.8483	0.8397	0.8439
OmniPage 18 [9]	0.8460	0.8380	0.8420
Nurminen [9]	0.8693	0.8078	0.8374
Baseline [22]	**0.9179**	0.7616	0.8325
TabbyPDF [32]	0.8339	0.8298	0.8318
TEXUS [28]	0.8071	0.7823	0.7945

Bold values indicate the best results (the highest precision, recall, and F_1 score)

[a] Uses a domain-specific table detection model from [27]

[b] Uses a general-purpose table detection model from [20], i.e., the X152 (ResNeXt-512) Latex+Word model

Evaluation Results

Table 2 reports the results obtained by our method for the CTR task. For comparison, we present the best previously published results on this data set.[19]

The proposed approach outperformed the method presented in our previous work [22]. Moreover, we achieved the F_1 score better than all previously reported results except for the commercial FineReader method that won the original competition. The precision and recall scores obtained by our method are well balanced. In particular, the recall was improved significantly in comparison with the results of the baseline method from our previous work.

Moreover, comparing the results of two variants of our system that employed either the general-purpose or the domain-specific table detection model ("TSR Setup") we can see a clear advantage of the latter. Nevertheless, the general-purpose variant is still very competitive, outperforming other competitors, except for the FineReader engine.

ICDAR 2019 Evaluation

In this section, we present the results obtained by our method on the benchmark from the ICDAR 2019 Table Competition.[20]

[19] For fair comparison, we only include the prior work that reported the results of the CTR process and we exclude the methods that used only a subset of the data for evaluation.

[20] https://cndplab-founder.github.io/cTDaR2019/index.html.

Evaluation Setup

For this experiment, we employed the official tools and metrics used in the original competition.[21] We performed the evaluation of the track B2, which corresponds to the CTR process.

The organizers of this competition adopted the metrics employed in the ICDAR 2013 Table Competition, except that the textual content of the cells is not used for the comparison of adjacency relations, i.e., relations between the neighboring cells in a table, and the evaluation focuses on the geometrical proximity between the ground-truth and the recognized cells. The main metric used to compare the results of the examined methods is the weighted average F_1 score, abbreviated as WAvg. F_1, which is computed as a weighted sum of the F_1 scores obtained using different Intersection over Union (IoU) thresholds for the cell matching procedure. IoU is defined as the ratio between the area of the overlap and the union of two bounding boxes.

Evaluation Results

Table 3 reports the results of our method in comparison with the baseline approach and the best-reported scores.[22] In addition, we also evaluated ABBYY FineReader Engine,[23] a commercial solution that facilitates information extraction from documents and also provides a table recognition module. We used the method employed in Adams et al. [1] that parses all table blocks from the output in the ABBY-XML format and converts them to the XML format supported by the ICDAR 2019 evaluation tool.

Both variants of our system improved upon the baseline approach from our previous work. The variant that exploited the domain-specific table detection model performed substantially better than the general-purpose variant. Nevertheless, our system was outperformed by the state-of-the-art methods in terms of WAvg. F_1, although we performed on par with the FineReader engine. Interestingly, both our method and FineReader engine perform better than the other methods at the highest IoU threshold.

Correcting an Issue in the Evaluation Script

We carried out a thorough investigation of the results of our method but we could not explain the low recognition scores in the case of some test examples. Therefore, we performed a simple sanity check and fed the ground-truth data as the input to the evaluation script, evaluating it against itself and expecting to get perfect WAvg. F_1 score of 1.0. Surprisingly, we obtained a score of 0.793, which suggested some issues with the evaluation script. In fact, we located the problem in the code and fixed it. The problem caused incorrect table matching in the case when there are two or more tables in an image. After correcting the issue with the evaluation script, the sanity check passed.

Table 3 ICDAR 2019 evaluation (track B2—modern documents)

Method	IoU				WAvg. F_1
	0.6	0.7	0.8	0.9	
CascadeTabNet [27]	**0.438**	**0.354**	0.190	0.036	**0.232**
NLPR-PAL [7]	0.365	0.305	0.195	0.035	0.206
This work[a]	0.288	0.249	0.181	0.089	0.191
FineReader v12	0.270	0.239	**0.198**	**0.092**	0.190
This work[b]	0.267	0.237	0.173	0.086	0.181
Baseline [22]	0.233	0.213	0.164	0.064	0.159

Bold values indicate the highest WAvg. F_1 score overall and the best F_1 scores at different IoU thresholds

We report F_1 scores for the reference values of the IoU threshold between the ground-truth and the recognized cells. For comparison, we include the best results reported in previous works. *WAvg.* F_1 denotes the average F_1 score weighted by the IoU threshold for IoU $\in \{0.6, 0.7, 0.8, 0.9\}$

[a] Uses a domain-specific table detection model from [27]

[b] Uses a general-purpose table detection model from [20], i.e., the X152 (ResNeXt-512) Latex+Word model

The Case of Revised Annotations

Moreover, we noticed that the annotations available in the official repository hosting the data for this competition have recently been updated. To estimate the expected difference in recognition scores obtained using the revised versus the old annotations, we fed the old annotations as input and evaluated them against the revised annotations using the corrected evaluation script. The resulting WAvg. F_1 score of 0.647 suggested that the results obtained by evaluating against the old annotations could substantially differ from the results obtained by employing the revised annotations. Furthermore, as noted in our previous work [22], the WAvg. F_1 score employed in this competition is biased toward high overlap ratios between the cells, strongly penalizing lower IoU scores.

[21] https://github.com/cndplab-founder/ctdar_measurement_tool.

[22] For fair comparison, we include the prior work that employed the official tools and annotations for evaluation. For instance, the results of the Multi-Type-TD-TSR approach [6] are omitted because the authors reannotated the test data and used images of cropped tables as input.

[23] https://www.abbyy.com/ocr-sdk (SDK v12).

Re-evaluation Using a Corrected Script and Revised Annotations

Motivated by these observations, we performed a further evaluation of our method in four different scenarios reflecting the two above-mentioned observations: (1) Using either the corrected or the original evaluation script and (2) Using either the old or the revised annotations. As a baseline method in this experiment, we employ ABBYY FineReader Engine.

The results of the extended evaluation are presented in Table 4. The proposed method considerably outperformed ABBYY FineReader in the scenarios, where the revised annotations were used. Otherwise, both methods exhibited comparable scores. Consistent with the previous results, the variant of our method that employed the domain-specific table detection model outperformed the general-purpose variant. The best results were obtained when both the corrected script and the revised annotations were employed (Table 4).

Table 4 Extended ICDAR 2019 evaluation (track B2 — modern documents)

Corrected script	Revised annotations	Method	WAvg. F_1
×	×	This work[a]	**0.1908**
		FineReader v12	0.1904
		This work[b]	0.1806
✓	×	FineReader v12	**0.2481**
		This work[a]	0.2436
		This work[b]	0.2203
×	✓	This work[a]	**0.2773**
		This work[b]	0.2545
		FineReader v12	0.2351
✓	✓	This work[a]	**0.3446**
		This work[b]	0.3092
		FineReader v12	0.2903

Bold values indicate the best results within each evaluation scenario

We include the results of our method and ABBY FineReader Engine. We report the results of four different variants of the evaluation — using the evaluation script either before or after the correction as well as using either old or revised annotations. WAvg. F_1 denotes the average F_1 score weighted by the IoU threshold for IoU $\in \{0.6, 0.7, 0.8, 0.9\}$

[a]Uses a domain-specific table detection model from [27]

[b]Uses a general-purpose table detection model from [20], i.e., the X152 (ResNeXt-512) Latex+Word model

Table Interpretation Experiment

Evaluation Data Set

For table interpretation, a common, publicly available benchmark can hardly be found, neither for general data nor for our use case [18], which motivated us to prepare the data for experiments ourselves. To this end, we annotated 13 documents with tables from our internal biomedical data collection.[24]

In our evaluation scenario, the annotated, ground-truth data for a table consists of a list of tuples, each representing an intersection of a data row and the columns that correspond to the defined meanings. The annotations are stored in JSON files (Fig. 17c) with the following name pattern:

```
<FILE_ID>_<PAGE_NR>_<TABLE_IDX>.json
```

where `<FILE_ID>` is the file identifier, `<PAGE_NR>` is the page number in the corresponding PDF file, and `<TABLE_IDX>` is the index of a table on a page.

In total, 113 tuples from 17 tables were annotated and used as ground-truth test data in our experiment. Moreover, a separate, representative development set of four documents was also prepared and used to fine-tune the rules employed by our table interpretation method. Figure 18 presents an example of a ground-truth file from our data collection.

[24] We selected documents in the context of the histone deacetylase (HDAC) gene (https://en.wikipedia.org/wiki/Histone_deacetylase), focusing on the HDAC1 and HDAC6 target genes.

It is worth noting that only a subset of tables present in the employed data collection contains information relevant to our scenario. Even if it is the case, a table may contain superfluous columns that do not contain the target information. Therefore, we carefully designed the rules employed by our table interpretation method using the documents in the development set (Fig. 19).

Evaluation Setup

To evaluate the end-to-end table extraction process, we execute the complete pipeline presented in Fig. 1. We first detect and recognize all tables in the test data set using the baseline method [22], as it performs reliably in the scenario where the tabular layout is well defined, i.e., it follows the fully or partially bordered format, and the table labels are present.

Subsequently, the proposed table interpretation method is employed to extract the relevant tuples from the tables recognized in the previous step. To facilitate evaluation, the extracted tuples for each table are stored in a separate JSON file (Fig. 17c) using the same file name pattern as in the case of the ground-truth files.

We feed two sets of JSON files, each corresponding to the ground-truth and the recognized tables, respectively, as input to the evaluation script. For every page, the script creates

```
[
    {
        "compound": "9b (IC50;nM)",
        "hdac1_ic50": "84.9 \u00b1 25.1",
        "hdac6_ic50": "95.9 \u00b1 0.78"
    },
    {
        "compound": "SAHA (IC50;nM)",
        "hdac1_ic50": "102.7 \u00b1 5.9",
        "hdac6_ic50": "198.5 \u00b1 103.0"
    }
]
```

Fig. 18 An example of a ground-truth file from our collection used in our table interpretation experiment (*11_page07_table0.json*). The character '`\u00b1`' corresponds to the Unicode symbol '±'. Reprinted from Namysl et al. [22]

a bipartite graph with two sets of nodes corresponding to the ground-truth and the recognized tables, respectively (Fig. 20) and, subsequently, it performs maximum weight matching, as proposed by Edmonds [5], to find the correspondence between these two sets of tables.

To compute cumulative scores, the results from all pages are collected and the exact precision, recall, and F_1 score are calculated. The tuples from the missed reference tables and incorrectly extracted relations are also included in the reported results. Therefore, the obtained scores reflect the performance of the complete table extraction process.

Evaluation Results

Table 5 reports the results of the complete table extraction system studied in this work. Our system extracted 74 tuples from 10 out of 28 tables present in the test data set, achieving a solid F_1 score of 0.7380. Moreover, after decoupling the errors that result from the missed reference tables, our table interpretation method exhibits a high F_1 score of 0.9388 and proves its utility.

As we expect that lower recall of the complete system resulted from the errors made by its upstream components, we qualitatively analyzed the results and discovered that only one false-positive and one false-negative error was directly related to the designed interpretation rules. The remaining errors were caused by table structure recognition issues like incorrectly merged cells.

Discussion

Robustness to Preprocessing Errors

In the preprocessing stage ("Preprocessing"), some solid separators might be missed or false-positively recognized. The proposed TSR heuristics are designed to mitigate the errors caused by preprocessing artifacts and imperfect table formatting. In

```
[
  {
    "id": "compound",
    "keywords": ["Compound", "compd", "Comp.", "cpd"],
    "datatype": "string",
    "weightTitle": 1.0,
    "weightContent": 0.0,
    "minAffinityScore": 0.5
  },
  {
    "id": "hdac1_gene",
    "keywords": ["HDAC1"],
    "titleRegex": "^HDAC[-]{0,1}1[^\\d].*$",
    "datatype": ["double", "range", "integer"],
    "weightTitle": 1.0,
    "weightContent": 0.0,
    "minAffinityScore": 0.85
  },
  {
    "id": "hdac6_gene",
    "keywords": ["HDAC6"],
    "titleRegex": "^HDAC[-]{0,1}6[^\\d].*$",
    "datatype": ["double", "range", "integer"],
    "weightTitle": 1.0,
    "weightContent": 0.0,
    "minAffinityScore": 0.85
  }
]
```

Fig. 19 A JSON file defining the meanings and rules for matching columns to these meanings used in our table interpretation experiment. Reprinted from Namysl et al. [22]

particular, the heuristic for partially bordered and borderless tables adds virtual ruling lines to facilitate the recognition of tables that do not strictly follow the rules of the booktabs format (see "Partially Bordered and Borderless Tables"). Moreover, the proposed heuristic for fully bordered tables expands the separator boxes to increase the chance of intersection with the neighboring solid separators in the separator merging stage ("Fully Bordered Tables") to facilitate recognition of tabular grids that contain boxes that are not fully enclosed.

ICDAR 2019 Evaluation

As it was shown in "ICDAR 2013 Evaluation", the comparison of the results reported on the ICDAR 2019 benchmark could be difficult. We advocate that the setup that employs the revised annotations and the corrected script should be used to evaluate approaches that report the results of this data set in the future. On the other hand, the scenario where the old annotations and the evaluation script before the correction are employed could be additionally presented to be used as a reference for comparison with the methods that reported the results on this data set in the past.

Proposed vs. Baseline TSR Method

In this section, we present all major differences between the TSR approach proposed in this work and the baseline

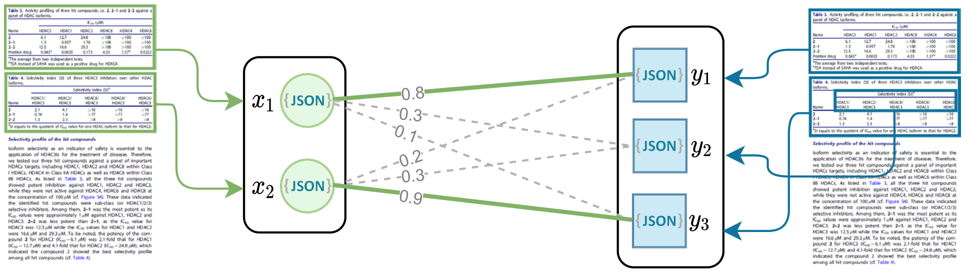


Fig. 20 An example of a weighted bipartite interpretation graph that contains two ground-truth and three recognized tables, represented by green circles and blue squares, respectively. Each vertex in a graph corresponds to a set of tuples extracted from a table and stored in a separate JSON file. The edges are weighted by the F_1 scores of the matching between the corresponding sets of tuples. The matching with the maximum sum of weights is marked with green solid lines. Note that the y_2 vertex corresponds to a false-positive result, which is not included in the final matching. Adapted from Namysl et al. [22]

Table 5 Results of information extraction from tabular data

Method	TP	FP	FN	Precision	Recall	F_1
This work: end-to-end method	69	4	45	0.9452	0.6053	0.7380
This work: interpretation-only	69	4	5	0.9452	0.9324	0.9388

We include the scores obtained both through the end-to-end table extraction process (This work: end-to-end method) and solely from the correctly recognized tables (This work: interpretation-only). We report the precision, recall, and F_1 score. TP, FN, and FP refer to the number of tuples that were perfectly matched (true-positive), missed (false-negative), or incorrectly recognized (false-positive), respectively. Adapted from Namysl et al. [22]

method from Namysl et al. [22]. In our previous work, we noticed a precision-recall gap in the results of our method. To overcome this issue, in this work, we proposed to integrate a deep learning-based table detection module, which indeed balanced and improved the results of our method, as it was empirically validated in "Table Recognition Experiments".

Note that, in this work, the input to the TSR method is supposed to contain a single table object. The approach from our previous work takes the image of a whole page as input to perform table detection and structure recognition at once. To adapt the method presented in Namysl et al. [22] to this new scenario, we have implemented the following changes:

(1) We do not use table labels for filtering table candidates in this work as the table detection module already delivers fairly accurate detection results.
(2) We filter all fully bordered tables based on their row height ratio (see Fig. 7) to discard all candidates that contain many rows that are not separated by a solid separator. These candidates are preferably handled by the subsequent heuristic for partially bordered and borderless tables.
(3) Our heuristic for partially bordered and borderless tables adds virtual ruling lines to facilitate the recognition of tables that do not strictly follow the booktabs format, as illustrated in Fig. 10. We also merge vertically overlapping candidates, as shown in Fig. 11.
(4) We estimate the threshold for column separation using the average character size within the table instead of the median unit distance between the words on a page because, in the new scenario, the full page content cannot be exploited by the TSR method. Nevertheless, the strategy used for threshold calculation employed in this work is still effective and less complicated than before.

Conclusions

In this article, a flexible, hybrid table extraction system was presented. It combines a deep learning-based table detection module with heuristic-based TSR method to infer the exact structure of tables in unstructured documents. Moreover, to extract semantic information from tables, the basic formulation of the table recognition task is complemented by including a graph-based table interpretation method. The proposed system works with both image-based inputs and born-digital PDF files. Our approach is modular and configurable, allowing us to adapt particular processing steps to a specific scenario.

We conducted extensive experiments on two challenging table recognition benchmarks, outperforming the baseline approach from our previous work and achieving results on par with the state-of-the-art methods on the respective data sets. Moreover, we evaluated our system in a scenario, where the target information is extracted directly from raw documents, and achieved a solid F_1 score that confirmed the utility of our holistic table extraction system.

We make the corrected evaluation script used in ICDAR 2019 Table Competition, the evaluation script employed in the experiment performed on the ICDAR 2013 benchmark, and the output XML files as well as detailed log files produced by our method publicly available and hope that our contribution will foster fair and reproducible future research on information extraction in the document processing domain.

Future work could investigate different choices for the table detection module, preferably trained using a large, representative data set containing tables with various layouts, originating from different sources. Perspectively, we intend to process various documents, including but not limited to invoices or balance sheets as well as camera-captured documents [42]. The most promising direction for future improvements is the incorporation of recent advances in the field of multimodal, pretrained models that exploit both visual and text information, such as the work presented recently by Huang et al. [14].

Author Contributions Conceptualization and methodology: Marcin Namysl; Formal analysis and investigation: Marcin Namysl; Software: Marcin Namysl; Supervision and validation: Sven Behnke; Writing—original draft preparation: Marcin Namysl, Alexander Esser; Writing—review and editing: Marcin Namysl, Sven Behnke, Alexander Esser, Joachim Köhler.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was supported in part by the Fraunhofer Internal Programs under Grant No. 836 885.

Data availability statement The resources that support the findings of this study are available on GitHub: https://github.com/mnamysl/tabrecsncs and https://github.com/mnamysl/table-interpretation. The datasets analyzed in Section "Table Recognition Experiments" are available in the following repositories: ICDAR 2013 Table Competition (https://roundtrippdf.com/en/data-extraction/pdf-table-recognition-dataset) and ICDAR 2019 cTDaR (https://github.com/cndplab-founder/ICDAR2019_cTDaR).

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Adams T, Namysl M, Kodamullil AT, Behnke S, Jacobs M. Benchmarking table recognition performance on biomedical literature on neurological disorders. Bioinformatics. 2021;38(6):1624–30. https://doi.org/10.1093/bioinformatics/btab843.
2. Auer S, Bizer C, Kobilarov G, Lehmann J, Cyganiak R, Ives Z. DBpedia: a nucleus for a web of open data. In: The semantic web. Berlin, Heidelberg: Springer; 2007. p. 722–35. https://doi.org/10.1007/978-3-540-76298-0_52.
3. Debnath S, Debnath T, Bhaumik S, Majumdar S, Kalle AM, Aparna V. Discovery of novel potential selective HDAC8 inhibitors by combine ligand-based, structure-based virtual screening and in-vitro biological evaluation. Sci Rep. 2019;9(1):17174. https://doi.org/10.1038/s41598-019-53376-y.
4. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009; pp 248–255 (2009) https://doi.org/10.1109/CVPR.2009.5206848.
5. Edmonds J. Maximum matching and a polyhedron with 0, 1 vertices. J Res Natl Bur Stand. 1965;69B:125–30.
6. Fischer, P., Smajic, A., Abrami, G., Mehler, A.: Multi-Type-TD-TSR - Extracting tables from document images using a multi-stage pipeline for table detection and table structure recognition: From OCR to structured table representations. In: KI 2021: Advances in Artificial Intelligence. Lecture Notes in Computer Science, vol. 12873. Springer, Cham, 2021; pp. 95–108. https://doi.org/10.1007/978-3-030-87626-5_8
7. Gao L, Huang Y, Déjean H, Meunier J-L, Yan Q, Fang Y, Kleber F, Lang E. ICDAR 2019 competition on table detection and recognition (cTDaR). In: International Conference on Document Analysis and Recognition (ICDAR), 2019; pp. 1510–1515. https://doi.org/10.1109/ICDAR.2019.00243.
8. Gatos B, Danatsas D, Pratikakis I, Perantonis SJ. Automatic table detection in document images. In: Pattern recognition and data mining. Berlin, Heidelberg: Springer; 2005. p. 609–18.
9. Göbel M, Hassan T, Oro E, Orsi, G. ICDAR 2013 Table Competition. In: International Conference on Document Analysis and Recognition (ICDAR), 2013; pp. 1449–1453. https://doi.org/10.1109/ICDAR.2013.292.
10. Haralick RM, Shapiro LG. Image segmentation techniques. In: Applications of Artificial Intelligence II, vol. 0548, 1985; pp. 2–9. https://doi.org/10.1117/12.948400. International Society for Optics and Photonics.
11. Hassan T, Baumgartner R. Table recognition and understanding from PDF files. In: International Conference on Document Analysis and Recognition (ICDAR), vol. 2, pp. 1143–1147 (2007). https://doi.org/10.1109/ICDAR.2007.4377094.
12. Herzig J, Nowak PK, Müller T, Piccinno F, Eisenschlos J. TaPas: Weakly supervised table parsing via pre-training. In: Annual Meeting of the Association for Computational Linguistics (ACL), pp. 4320–4333. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.acl-main.398.
13. Hoshen J, Kopelman R. Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm. Phys Rev B. 1976;14:3438–45. https://doi.org/10.1103/PhysRevB.14.3438.
14. Huang Y, Lv T, Cui L, Lu Y, Wei F. LayoutLMv3: Pre-training for document AI with unified text and image masking. In: ACM International Conference on Multimedia. Association for Computing Machinery, New York, NY, USA 2022; pp. pp. 4083–4091. https://doi.org/10.1145/3503161.3548112.
15. Hulsebos M, Hu K, Bakker M, Zgraggen E, Satyanarayan A, Kraska T, Demiralp C, Hidalgo C. Sherlock: a deep learning approach to semantic data type detection. In: ACM SIGKDD

International Conference on Knowledge Discovery and Data Mining (KDD). Association for Computing Machinery, New York, NY, USA 2019; pp. 1500–1508. https://doi.org/10.1145/3292500.3330993.

16. Kleene SC. Representation of events in nerve nets and finite automata. Technical report, Rand Project Air Force Santa Monica, CA, Santa Monica, CA 1951. https://apps.dtic.mil/sti/pdfs/ADA596138.pdf.
17. Konya I. Adaptive methods for robust document image understanding. PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn (April 2013).
18. Lage-Rupprecht V, Schultz B, Dick J, Namysl M, Zaliani A, Gebel S, Pless O, Reinshagen J, Ellinger B, Ebeling C, Esser A, Jacobs M, Claussen C, Hofmann-Apitius M. A hybrid approach unveils drug repurposing candidates targeting an Alzheimer pathophysiology mechanism. Patterns. 2022;3(3): 100433. https://doi.org/10.1016/j.patter.2021.100433.
19. Levenshtein VI. Binary codes capable of correcting deletions, insertions, and reversals. Sov Phys Dokl 1966;10(8).
20. Li M, Cui L, Huang S, Wei F, Zhou M, Li Z. TableBank: table benchmark for image-based table detection and recognition. In: Language Resources and Evaluation Conference (LREC), pp. 1918–1925. European Language Resources Association, Marseille, France 2020. https://aclanthology.org/2020.lrec-1.236.
21. Macdonald E, Barbosa D. Neural relation extraction on Wikipedia tables for augmenting knowledge graphs. In: ACM International Conference on Information and Knowledge Management. CIKM '20, pp. 2133–2136. Association for Computing Machinery, New York, NY, USA 2020. https://doi.org/10.1145/3340531.3412164.
22. Namysl M, Esser A, Behnke S, Köhler J. Flexible table recognition and semantic interpretation system. In: International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, vol. 4: VISAPP, pp. 27–37. SciTePress, Setúbal, Portugal 2022. https://doi.org/10.5220/0010767600003124. INSTICC.
23. Nurminen A. Algorithmic extraction of data in tables in PDF documents. Master's thesis, Tampere University of Technology 2013.
24. Oro E, Ruffolo M. PDF-TREX: an approach for recognizing and extracting tables from PDF documents. In: International Conference on Document Analysis and Recognition (ICDAR), 2009; pp. 906–910. https://doi.org/10.1109/ICDAR.2009.12. IEEE.
25. Otsu N. A threshold selection method from gray-level histograms. IEEE Trans Syst Man Cybern. 1979;9(1):62–6. https://doi.org/10.1109/TSMC.1979.4310076.
26. Paliwal SS, D, V, Rahul R, Sharma M, Vig L. TableNet: deep learning model for end-to-end table detection and tabular data extraction from scanned document images. In: International Conference on Document Analysis and Recognition (ICDAR), 2019; pp. 128–133. https://doi.org/10.1109/ICDAR.2019.00029.
27. Prasad D, Gadpal A, Kapadni K, Visave M, Sultanpure K. CascadeTabNet: an approach for end to end table detection and structure recognition from image-based documents. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2020; pp. 2439–2447. https://doi.org/10.1109/CVPRW50498.2020.00294.
28. Rastan R, Paik H-Y, Shepherd J. TEXUS: a task-based approach for table extraction and understanding. In: ACM Symposium on Document Engineering (DocEng), 2015; pp. 25–34. https://doi.org/10.1145/2682571.2797069.
29. Ren S, He K, Girshick R, Sun J. Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 28. Curran Associates, Inc., Red Hook, NY, USA; 2015.
30. Reza MM, Bukhari SS, Jenckel M, Dengel A.: Table localization and segmentation using GAN and CNN. In: International Conference on Document Analysis and Recognition Workshops (ICDARW), vol. 5, 2019; pp. 152–157. https://doi.org/10.1109/ICDARW.2019.40097.
31. Schreiber S, Agne S, Wolf I, Dengel A, Ahmed S. DeepDeSRT: deep learning for detection and structure recognition of tables in document images. In: International Conference on Document Analysis and Recognition (ICDAR), vol. 01, 2017; pp. 1162–1167. https://doi.org/10.1109/ICDAR.2017.192.
32. Shigarov A, Altaev A, Mikhailov A, Paramonov V, Cherkashin E. TabbyPDF: Web-based system for PDF table extraction. In: Information and Software Technologies, 2018; pp. 257–269. Springer, Cham. https://doi.org/10.1007/978-3-319-99972-2_20.
33. Silva ACE, Jorge AM, Torgo L. Design of an end-to-end method to extract information from tables. Int J Doc Anal Recognit. 2005;8:144–71. https://doi.org/10.1007/s10032-005-0001-x.
34. Smith, R.: An overview of the Tesseract OCR engine. In: International Conference on Document Analysis and Recognition (ICDAR), vol. 2, 2007; pp. 629–633. https://doi.org/10.1109/ICDAR.2007.4376991.
35. Wainer H. Improving tabular displays, with NAEP tables as examples and inspirations. J Educ Behav Stat. 1997;22(1):1–30. https://doi.org/10.3102/10769986022001001.
36. Wang J, Sun K, Cheng T, Jiang B, Deng C, Zhao Y, Liu D, Mu Y, Tan M, Wang X, Liu W, Xiao B. Deep high-resolution representation learning for visual recognition. IEEE Trans Pattern Anal Mach Intell. 2021;43(10):3349–64. https://doi.org/10.1109/TPAMI.2020.2983686.
37. Xie S, Girshick R, Dollar P, Tu Z, He K. Aggregated residual transformations for deep neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017.
38. Yan C, He Y. Synthesizing type-detection logic for rich semantic data types using open-source code. In: International Conference on Management of Data (SIGMOD). Association for Computing Machinery, New York, NY, USA 2018; pp. 35–50. https://doi.org/10.1145/3183713.3196888.
39. Yu T, Wu C-S, Lin XV, Wang B, Tan YC, Yang X, Radev D, Socher R, Xiong C. GraPPa: Grammar-augmented pre-training for table semantic parsing. In: International Conference on Learning Representations (ICLR) 2021. https://openreview.net/forum?id=kyaIeYj4zZ.
40. Zhang D, Hulsebos M, Suhara Y, Demiralp C, Li J, Tan W-C. Sato: Contextual semantic type detection in tables. VLDB Endow. 2020; 13(12), 1835–1848. https://doi.org/10.14778/3407790.3407793.
41. Zheng Y, Liu C, Ding X, Pan S. Form frame line detection with directional single-connected chain. In: International Conference on Document Analysis and Recognition (ICDAR), 2001; pp. 699–703. https://doi.org/10.1109/ICDAR.2001.953880.
42. Zhu Z, Gao L, Li Y, Huang Y, Du L, Lu N, Wang X. NTable: a dataset for camera-based table detection. In: Document Analysis and Recognition (ICDAR), 2021; pp. 117–129. Springer, Cham. https://doi.org/10.1007/978-3-030-86331-9_8.
43. Namysł M. Robust Information Extraction From Unstructured Documents. Ph.D. Dissertation, Rheinische Friedrich-Wilhelms-Universität Bonn (January 2023). https://hdl.handle.net/20.500.11811/10560.