TableParser: Automatic Table Parsing with Weak Supervision from Spreadsheets

Susie Xi Rao[1,2,3], Johannes Rausch[1,3], Peter Egger[2] and Ce Zhang[1]

[1]*Systems Group, Department of Computer Science (ETH Zurich), Stampfenbachstrasse 114, 8092 Zurich, Switzerland*

[2]*Chair of Applied Economics, Department of Management, Technology, and Economics (ETH Zurich), Leonhardstrasse 21, 8092 Zurich, Switzerland*

[3]*These authors contributed equally to this work.*

Abstract

Tables have been an ever-existing structure to store data. There exist now different approaches to store tabular data physically. PDFs, images, spreadsheets, and CSVs are leading examples. Being able to parse table structures and extract content bounded by these structures is of high importance in many applications. In this paper, we devise TableParser, a system capable of parsing tables in both native PDFs and scanned images with high precision. We have conducted extensive experiments to show the efficacy of domain adaptation in developing such a tool. Moreover, we create TableAnnotator and ExcelAnnotator, which constitute a spreadsheet-based weak supervision mechanism and a pipeline to enable table parsing. We share these resources with the research community to facilitate further research in this interesting direction.

Keywords

table structure parsing, table annotation, Mask R-CNN, weak supervision, domain adaptation

1. Introduction

Automated processing of electronic documents is a common task in industry and research. However, the lack of structures in formats such as native PDF files or scanned documents remains a major obstacle, even for state-of-the-art OCR systems. In practice, extensive engineering and ad-hoc code are required to recover the document structures, e.g., for headings, tables, or nested figures. Sometimes this is required even for text, e.g., in case of PDFs built on the basis of scans, especially, low-quality scans. These structures are hierarchically organized, which many existing systems often fail to recognize.

With the advance of machine learning (ML) and deep learning (DL) techniques, parsing documents can be done more efficiently than ever. As the first end-to-end system for parsing renderings into hierarchical document structures, DocParser [1] was recently introduced. It presents a robust way to parse complete document structures from rendered PDFs. Such learning-based systems require large amounts of labeled training data. This problem is alleviated through a novel weak supervision approach that automatically generates training data from structured LaTeX source files in readily available scientific articles. DocParser demonstrates a significant reduction of the labeling complexity through this weak supervision in their experiments.

As a special document type, tables are one of the most natural ways to organize structured contents. Tabular data are ubiquitous and come in different formats, e.g., CSV (plain and unformatted) and Microsoft Excel (annotated and formatted), depending on the file creation. Many data processing tasks require tables to be represented in a structured format. However, structured information is not always available in rendered file formats such as PDF. Especially when PDFs are generated from image scans, such information is missing. Luckily, the existing matrix-type organization of spreadsheets can assist to automatically generate document annotations to PDFs. **With spreadsheets as weak supervision, this paper proposes a pipeline to provide an automated process of reading tables from PDFs and utilize them as a weak supervision source for DL systems.**

There are three different types of tasks discussed in the literature about table processing in PDFs, namely, table detection, table structure parsing/recognition [2].[1]

SDU@AAAI'22: The AAAI-22 Workshop on Scientific Document Understanding at the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22), virtual.

✉ srao@ethz.ch (S. X. Rao); johannes.rausch@inf.ethz.ch (J. Rausch); pegger@ethz.ch (P. Egger); ce.zhang@inf.ethz.ch (C. Zhang)
🌐 https://susierao.github.io/ (S. X. Rao); https://github.com/j-rausch (J. Rausch); https://https://cae.ethz.ch/ (P. Egger); https://ds3lab.inf.ethz.ch/ (C. Zhang)
iD 0000-0003-2379-1506 (S. X. Rao); 0000-0002-9409-4401 (J. Rausch); 0000-0002-0546-1207 (P. Egger)
🐦 https://twitter.com/ds3lab (C. Zhang)

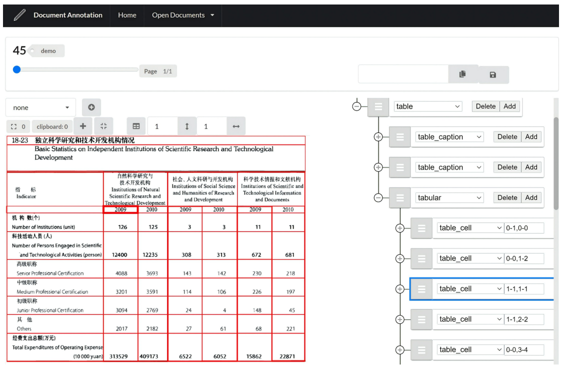
[1] Table detection is a task to draw the bounding boxes of tables in documents; table structure recognition/parsing refers to the (additional) identification of the structural (row and column layout) information of tables. We distinguish between bottom-up and top-down approaches in table structure detection. Bottom-up typically refers to structure detection by recognizing formatting cues such as text, lines, and spacing, while top-down entails table cell detection (see [3, 4, 2]).

Table detection is a popular task with a large body of literature, table structure parsing and table recognition were revisited[2] after the pioneering work of [7] using state-of-the-art deep neural networks. Before DL started to gain success in object detection, table structure parsing was done by bottom-up approaches, using heuristics or ML-based methods like [8, 9]. See [4, 10] for comprehensive reviews on ML methods. The purposes of table structure detection are either layout detection [3] or information retrieval [4] from tabular structures, usually with the former as a preprocessing step for the latter.

The DL-based methods in [7, 11] are among the first to apply neural networks designed for object detection to table parsing. Typically, taking pretrained object detection models e.g., Faster RCNN [12, 13] on benchmarking datasets like ImageNet [14], Pascal VOC [15], and Microsoft COCO [16], they fine-tune the pretrained models with in-domain images for table detection and table structure parsing (domain adaption and transfer learning). In some best performing frameworks [17, 18, 19], they all jointly optimize the structure detection and entity relations in the structure, as in DocParser.

However, a key problem in training DL-based systems is the labeling complexity of generating high-quality in-domain annotations. More generally, an essential limiting factor is the lack of large amounts of training data. Efforts have been put into generating datasets to enable tasks with weak supervision. TableBank [20] is built upon a data set of Word and LaTeX files and extracts annotations directly from the sources. They use 4-gram BLEU score to evaluate the cell content alignments. However, the table layout structure is not of particular focus in TableBank. PubTabNet [2] enables table detection and table cell content detection. arXivdocs-target and arXivdocs-weak by DocParser [1] enables an end-to-end document parsing system of the hierarchical document structure.

In this paper, we devise TableParser with inspiration from DocParser, due to its flexibility in processing both tables and more general documents. We demonstrate that **TableParser** is an effective tool for recognizing table structures and content. The application of TableParser to a new target domain requires newly generated training data. Depending on the target domain, we specify two TableParsers: **ModernTableParser** fine-tuned with native PDFs and **HistoricalTableParser** fine-tuned with scan images. TableParser works in conjunction with **TableAnnotator** (Figure 1) which efficiently assists developers in visualizing the output, as well as help users to generate high-quality human annotations.[3] To generate training instances, we develop **ExcelAnnotator** to interact with spreadsheets and produce annotations for weak supervision.



18-23 独立科学研究和技术开发机构情况
Basic Statistics on Independent Institutions of Scientific Research and Technological Development

指　标 Indicator	自然科学研究与技术开发机构 Institutions of Natural Scientific Research and Technological Development		社会、人文科研与开发机构 Institutions of Social Science and Humanities of Research and Development		科学技术情报和文献机构 Institutions of Scientific and Technological Information and Documents	
	2009	2010	2009	2010	2009	2010
机 构 数(个) Number of Institutions (unit)	126	125	3	3	11	11
科技活动人员 (人) Number of Persons Engaged in Scientific and Technological Activities (person)	12400	12235	308	313	672	681
高级职称 Senior Professional Certification	4088	3693	143	142	230	218
中级职称 Medium Professional Certification	3201	3591	114	106	226	197
初级职称 Junior Professional Certification	3094	2769	24	4	148	45
其 他 Others	2017	2182	27	61	68	221
经费支出总额(万元) Total Expenditures of Operating Expense (10 000 yuan)	313529	409173	6522	6052	15862	22871

table
table_caption
table_caption
tabular
table_cell 0-1,0-0
table_cell 0-0,1-2
table_cell 1-1,1-1
table_cell 1-1,2-2
table_cell 0-0,3-4

Figure 1: TableAnnotator.

With **ExcelAnnotator**, we have compiled a spreadsheet dataset ZHYearbooks-Excel, which is processed via a Python library on Excel (PyWin32[4]) to leverage the structured information stored in the spreadsheets. TableParser is trained with 16’041 Excel-rendered tables using detectron2 ([21, 22]) and fine-tuned with 17 high-quality manual annotations in each domain. We have conducted extensive experiments of domain adaptation. Finally, we evaluate different TableParsers in two domains and make the following observations:

1. In general, domain adaptation works well with fine-tuning the pretrained model (M_{WS} in Figure 2) with high-quality in-domain data.
2. On the test set of 20 tables rendered by Excel, with ModernTableParser we are able to achieve an average precision score (IoU ≥ 0.5) of 83.53% and 73.28% on table rows and columns, respectively.
3. We have tested our HistoricalTableParser on scanned tables in both historical (medium-quality, scan-based) and modern tables. Overall, HistoricalTableParser works better than ModernTableParser on tables stored in image scans.
4. Interestingly, we find that ModernTableParser built on top of DocParser [1] is very robust in adapting to new domains, such as scanned historical tables.

We are willing to open source the ZHYearbook-Excel dataset, TableAnnotator, TableParser system, and its pipeline to the research communities.[5] Moreover, we

[2] Some recent works on Cascade R-CNN [5, 6] manage to push the frontier of table detection. See [1] for a general review on table detection and [2] for a general review on table recognition.

[3] For a live demo of table annotations using our annotation tool, refer to the video under https://github.com/DS3Lab/TableParser/blob/main/demo/2021-06-15%2002-05-58.gif.

[4] https://pypi.org/project/pywin32/ (last accessed: Sep. 30, 2021).

[5] The source code, data, and/or other artifacts for the complete TableParser pipeline have been made available at https://github.com/DS3Lab/TableParser.

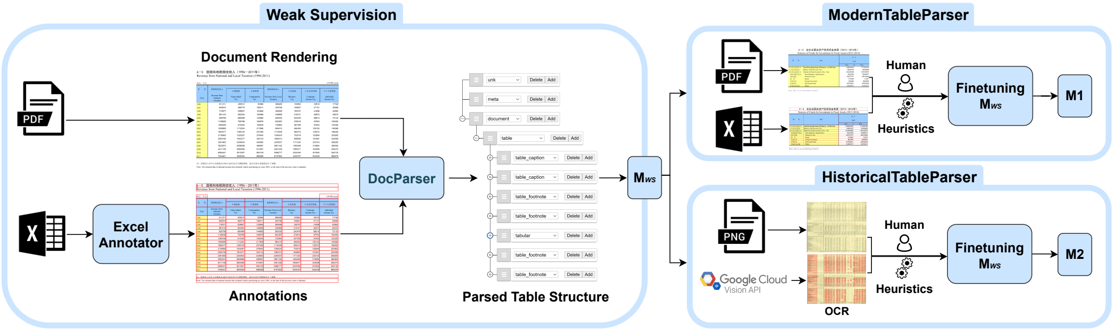


Figure 2: System design of TableParser: Weak supervision, ExcelAnnotator, ModernTableParser and HistoricalTableParser. M_{WS}: the pretrained model with the dataset ZHYearbooks-Excel-WS. M1: for ModernTableParser, fine-tuned on Excel-rendered images; M2: for HistoricalTableParser, fine-tuned on scan images.

welcome future contributions to the project to further increase the usability of TableParser in various domains.

To summarize, our key contributions in this paper are:

1. We present **TableParser** which is a robust tool for parsing modern and historical tables stored in native PDFs or image scans.
2. We conduct experiments to show the efficacy of domain adaptation in TableParser.
3. We contribute a new pipeline (using **ExcelAnnotator** as the main component) to automatically generate weakly labeled data for DL-based table parsing.
4. We contribute **TableAnnotator** as a graphical interface to assist table structure understanding and manual labeling.
5. We open-source the spreadsheet weak supervision dataset and the pipeline of TableParser to encourage further research in this direction.

2. TableParser System

2.1. Problem Description

Following the hierarchical document parsing in DocParser, our objective is to generate a hierarchical structure for a table which consists of the entities (*table, tabular, table_caption, table_row, table_column, table_footnote*) and their relations in the document tree.

Our ultimate goal of table structure parsing is (1) to establish row/column relationships between the table cells, and (2) post-process the established structure and cell content (e.g., with PDFMiner[6] or OCR engines) to enable a CSV export function. In this paper, we emphasize (1) and are still in development to enable (2). Our work will enable a user to parse a table stored in a PDF format and obtain (i) the location of a certain cell (specified by its row range and column range) and (ii) the cell content mapped to the cell location.

2.2. System Components

We introduce the main system components in TableParser, incl. TableAnnotator, ExcelAnnotator, ModernTableParser, and HistoricalTableParser.

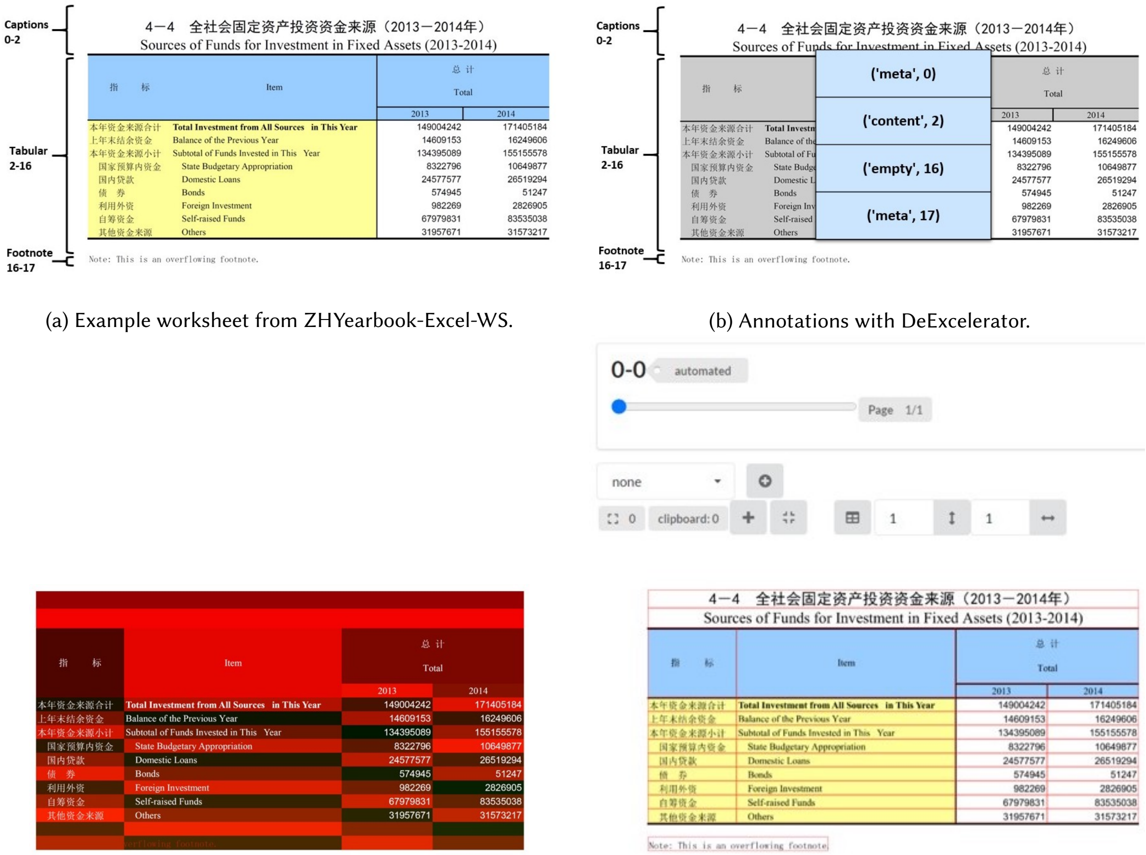
2.2.1. TableAnnotator.

In Figure 1 we show TableAnnotator, which is mainly composed of two parts: image panel (left) and document tree (right). In the code repository[7], there is a manual describing its functionalities in details. In a nutshell, annotators can draw bounding boxes on the left panel and create their entities and relationships on the right. In Figure 1, the highlighted bounding box (the red thick contour on the left) corresponds to the *table_cell* on the second row and second column, indexed by 1-1, 1-1 (the blue highlight on the right). Note that TableAnnotator is versatile and can be used to annotate not only tables, but also generic documents. The output of the tree is in JSON format, as shown in the following code snippet.

```
[{"id": 28,
"category": "table_cell",
"properties": "1-1,1-1",
"row_range": [1,1],
"col_range": [1,1],
"parent": 9},
{"id": 29,
"category": "box",
"page": 0,
"bbox": [365,332,299,27],
"parent": 28}]
```

[6] https://pypi.org/project/pdfminer/ (last accessed: Nov. 11, 2021).

[7] TableAnnotator repo: https://anonymous.4open.science/r/doc_annotation-SDU-AAAI22.



4—4 全社会固定资产投资资金来源（2013—2014年）
Sources of Funds for Investment in Fixed Assets (2013-2014)

指　　标	Item	总 计 Total	
		2013	2014
本年资金来源合计	**Total Investment from All Sources in This Year**	149004242	171405184
上年末结余资金	Balance of the Previous Year	14609153	16249606
本年资金来源小计	Subtotal of Funds Invested in This Year	134395089	155155578
国家预算内资金	State Budgetary Appropriation	8322796	10649877
国内贷款	Domestic Loans	24577577	26519294
债　券	Bonds	574945	51247
利用外资	Foreign Investment	982269	2826905
自筹资金	Self-raised Funds	67979831	83535038
其他资金来源	Others	31957671	31573217

Note: This is an overflowing footnote.

(a) Example worksheet from ZHYearbook-Excel-WS.

(b) Annotations with DeExcelerator.

(c) Representing bounding boxes in Excel.

(d) Visualization of bounding boxes with TableAnnotator.

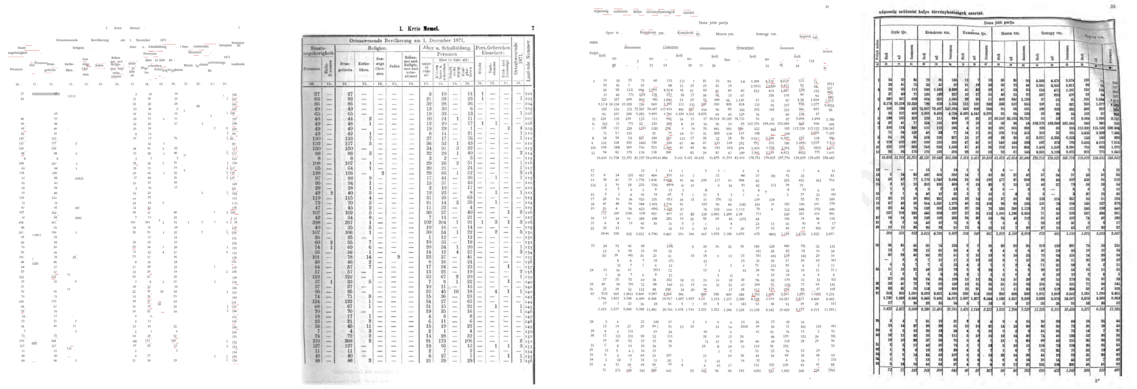
Figure 3: Working example in ExcelAnnotator.

2.2.2. ModernTableParser.

We train ModernTableParser using the data generated by weak supervision signals from Excel sheets and fine-tuned by high-quality manual annotations in this domain. In Figure 2, we show the system design following the underlying components of DocParser.[8] We denote the model that produces ModernTableParser as **M1**.

Weak Supervision with ExcelAnnotator. Now we present the crucial steps in generating weak supervision (the model M_{WS} in Figure 2) for TableParser. These steps are mainly conducted by ExcelAnnotator in Figure 2 (left). Take a worksheet-like Figure 3 (a) from our ZHYearbook-Excel-WS dataset (cf. Section 3), where we see caption, tabular, and footnote areas. We subsequently use DeExcelerator [23] to extract relations from the spreadsheets. We utilize DeExcelerator to categorize the content, such that we can differentiate among table captions, table footnotes and tabular data and create a correct auxiliary file to each PDF containing the structural information of the represented table(s). Illustrated in Figure 3 (b), in this case we annotate the table caption and footnote as 'meta', and mark the range of content with 'content' and 'empty'. We use PyWin32 in Python to interact with Excel, so that intermediate representations like Figure 3 (c) can be created to retrieve entity locations in the PDF rendering. Concretely, we mark neighboring cells with distinct colors, remove all borders, and set the font color to white. To summarize, ExcelAnnotator detects spreadsheet metadata and cell types, as well as retrieves entity locations via intermediate representations. Finally, we are able to load the annotations into TableAnnotator to inspect the quality of weak supervision (Figure 3 (d)).

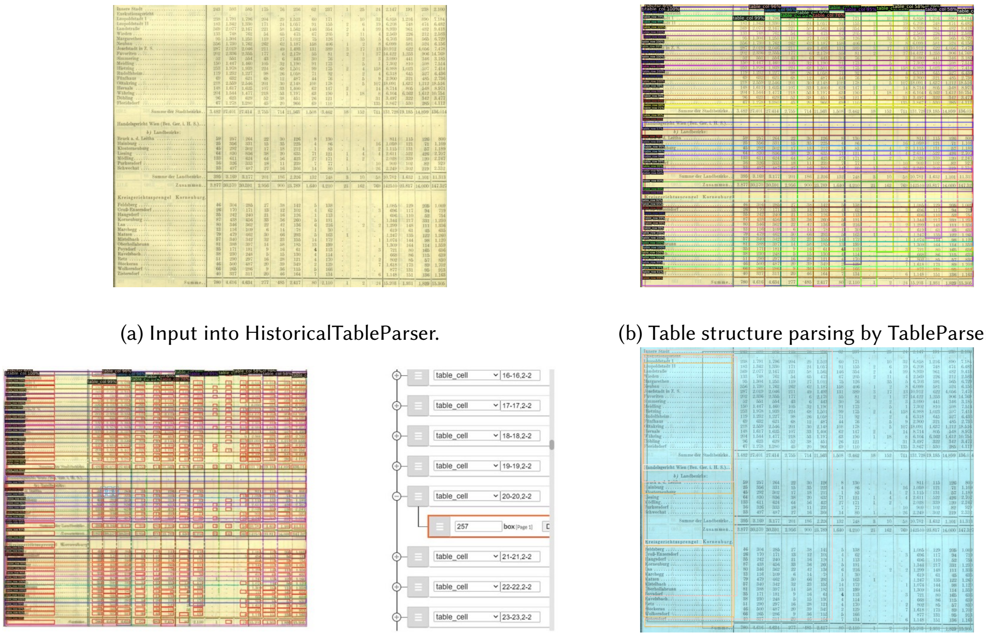
[8] The model structure of DocParser is sketched in Figure 11 of the DocParser paper [1], see https://arxiv.org/pdf/1911.01702.pdf. The model structure (Mask R-CNN) can also be found here.



(a) Bad quality of OCR (left).

(b) Good quality of OCR (left).

Figure 4: Google Vision OCR API output (left) of scan image (right).



(a) Input into HistoricalTableParser.

(b) Table structure parsing by TableParser.

(c) Merging the layout by TableParser and the OCR bounding boxes.

(d) Run LayoutParser [24] on tables.

Figure 5: Working example in HistoricalTableParser.

2.2.3. HistoricalTableParser.

We use the OCR engine from Google Vision API to recognize the text bounding boxes. Then we convert bounding boxes into the input format TableParser requires. Now we are able to manually adjust the bounding boxes in TableAnnotator to produce high-quality annotations. Note that the quality of OCR highly depends on the table layout (see (a) vs. (b) in Figure 4), we often need to adjust the locations of bounding boxes and redraw the bounding boxes of individual cells.

In Figure 2 (lower right), we show the system design by adding an OCR component and a fine-tuning component for domain adaptation. We denote the model that produces HistoricalTableParser as **M2**. Take Figure 5 (a) as input, TableParser can produce a parsed layout-like Figure 5 (b) which can be combined with the OCR bounding boxes in the subsequent steps and export as a CSV file (Figure 5 (c)).[9]

For domain adaptation, we assume that an out-of-domain model performs worse than an in-domain model in one domain. Namely, we would expect ModernTableParser to work better on Excel-rendered PDFs or tables created similarly; on the contrary, we would expect HistoricalTableParser to perform better on older table scans.

3. Datasets

We have compiled various datasets to train, fine-tune, test, and evaluate TableParser.

ZHYearbooks-Excel. We create three datasets from this source: ZHYearbooks-Excel-WS, ZHYearbooks-Excel-FT, and ZHYearbooks-Excel-Test, with 16'041, 17, and 20 tables in each set. On average, it takes 3 minutes 30 seconds for an annotator to produce high-quality annotations of a table. The manual annotations are done with automatically generated bounding boxes and document tree as aid.

ZHYearbooks-OCR. We create the dataset ZHYearbook-OCR-Test, with 20 tables. On average, it takes 2 minutes and 45 seconds to annotate a table with the similar annotation aids mentioned above.

EUYearbooks-OCR. We create two datasets from this source: EUYearbook-OCR-FT and EUYearbook-OCR-Test, with 17 and 10 tables, respectively. Note that these datasets contain various languages like Hungarian and German, with various formats depending on the language. On average, it takes 8 minutes and 15 seconds to annotate a table with the similar annotation aids mentioned above.

Miscellaneous historical yearbooks. We ran ModernTableParser and HistoricalTableParser on Chinese (in Simplified Chinese) and South Korean historical yearbooks (in Classical Chinese) and inspect their outputs qualitatively (see Section 5.2).

Human labeling efforts. We observe a large variance in labeling intensity across the datasets. The EUYearbooks-OCR datasets require more corrections per table compared to the datasets of modern tables. Moreover, they also require more iterations of human annotations with heuristics as aid.

4. Computational Setup

4.1. Mask R-CNN

In line with DocParser, we use the same model but with an updated backend implementation. Namely, we utilize Detectron2 to apply an updated version of Mask R-CNN [25]. For technical details of Mask R-CNN, we refer to DocParser [1]. In Appendix A, we illustrate the architecture of Mask R-CNN used in this paper.

4.1.1. Training Procedure: Weak Supervision + Fine-Tuning.

All neural models are initialized with weights trained on the MS COCO dataset. We first pretrain on the weak supervision data ZHYearbook-Excel-WS for 540k iterations, then fine-tune on our target datasets ZHYearbook-Excel-FT and EUYearbook-OCR-FT for M1 and M2, respectively. We then fine-tune each model across three phrases for a total of 30k iterations. This is split into 22k, 4k, 4k iterations, respectively. The performance is measured every 500 iterations via the IoU with a threshold of 0.5. We train all models in a multi-GPU setting, using 8 GPUs with a vRAM of 12 GB. Each GPU was fed with one image per training iteration. Accordingly, the batch size per training iteration is set to 8. Furthermore, we use stochastic gradient descent with a learning rate of 0.005 and learning momentum of 0.9.

4.1.2. Parameter Settings.

During training, we sampled randomly 100 entities from the ground truth per document image (i.e., up to 100 entities, as some document images might have less). In Mask R-CNN, the maximum number of entity predictions per image is set to 100. During prediction, we only keep entities with a confidence score of 0.5 or higher.

5. Results and Discussion

Here, we evaluate the performance of TableParser in two domains quantitatively and qualitatively.

5.1. Quantitative assessment

Metric. We first introduce the evaluation metric for the object detection/classification tasks. The metric we report is Average Precision (AP), which corresponds to an Intersection over Union rate of IoU=.50:.05:.95.[10] IoU ranges from 0 to 1 and specifies the amount of overlap between the predicted and ground truth bounding box. It is a common metric used when calculating AP.

[9] c.f. The performance of LayoutParser is quite poor on the tabular data in Figure 5 (d) using the best model from its model zoo (PubLayNet/faster_rcnn_R_50_FPN_3x). Input and annotated figures of original size can be found under https://github.com/DS3Lab/TableParser/tree/main/figures.

[10] We refer readers to https://cocodataset.org/#detection-eval for more details on the evaluation metrics (last accessed: Nov. 1, 2021).

Table 1
Fine-tuning results of M1 and M2. M1: for ModernTableParser, fine-tuned on Excel-rendered images; M2: for HistoricalTableParser, fine-tuned on scan images; FT: fine-tune.

ZHYearbook-Excel-FT				EUYearbook-OCR-FT			
Category	# instances	Average Precision		Category	# instances	Average Precision	
		M1 (FT)	**M2** (Test)			**M1** (Test)	**M2** (FT)
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
table	17	90.973	38.034	*table*	17	67.467	93.011
tabular	17	100.000	57.897	*tabular*	17	76.423	100.000
table_column	134	96.730	15.253	*table_column*	260	24.930	81.376
table_row	548	79.228	39.485	*table_row*	1180	19.256	60.899

Table 2
Test results of M1 and M2 on various data sets.

ZHYearbook-Excel-Test				ZHYearbook-OCR-Test				EUYearbook-OCR-Test			
Category	# instances	Average Precision		Category	# instances	Average Precision		Category	# instances	Average Precision	
		M1	M2			M1	M2			M1	M2
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
table	20	85.407	32.821	*table*	10	56.942	53.356	table	10	57.151	81.907
tabular	21	80.193	43.801	*tabular*	10	64.175	52.563	*tabular*	10	85.956	91.429
table_column	176	73.277	14.927	*table_column*	74	43.094	21.997	*table_column*	136	36.616	40.509
table_row	513	83.528	48.912	*table_row*	226	50.055	36.619	*table_row*	665	25.645	40.229

Performances in various domains. As we discussed in Section 2, we have developed ModernTableParser to parse tables with input images rendered by Excel (M1). Then, to work with historical tables in scans, we adapt the pretrained TableParser by fine-tuning it on scanned documents (M2). Now, we present the performances of M1 and M2 in two different domains in the following aspects:

1. **(P1)** the performances on fine-tuning sets on M1 and M2 in Table 1;
2. **(P2)** the performances on fine-tuning sets as test sets on M1 and M2 in Table 1;[11]
3. **(P3)** the performances on three test sets from two domains on M1 and M2 in Table 2.

(P1) & (P2). We want to study the impact of fine-tuning of a pretrained model (using a large body of tables generated by weak supervision signals). The instances used to fine-tune must be high-quality in-domain data. Concretely, we create in-domain annotations for modern tables (rendered by Excel) and historical tables (from scans) with high human efforts assisted by automatic preprocessing: ZHYearbook-Excel-FT and EUYearbook-OCR-FT, each with 17 tables. Note that the latter has much denser rows and columns than the former (see the tables in Figures 3 (a) vs. 5 (a) for an illustration). It is apparent from Table 1 that the AP performance of models on the fine-tuning sets is highly optimized (columns (3) and (8) in Table 1), and it should be better than using those datasets as test sets. This means, if we run M1 (fine-tuned by modern tables) on EUYearbook-OCR-FT (column (7) in Table 1), its performance is worse than fine-tuning; and if we run M2 (fine-tuned by historical tables) on ZHYearbook-Excel-FT (column (4) in Table 1), it performs worse than fine-tuning. Interestingly, if we compare the performance of M2 on modern tables (column (4) in Table 1) with the performance of M1 on historical tables (column (7) in Table 1), we clearly see that the latter has a better performance in all other categories than the class of *table_row*. This can be explained by the fact that the model trained on modern tables is robust in annotating historical tables, at least on the column level. We see this in Figures 9 and 10, where ModernTableParser clearly performs better. **However, the algorithm has problems in delineating narrow and less clearly separated rows.** This could be due to the setting of the maximum number of entities being 100 when predicting per table (Section 4.1).

(P3). In Table 2, we show the performances of three test sets from two domains (Excel-rendered PDFs and historical scans), namely, ZHYearbook-Excel-Test, ZHYearbook-OCR-Test, and EUYearbook-OCR-Test. We see that M2 which is fine-tuned by historical scans performs worse than M1 on both ZHYearbook-Excel-Test and ZHYearbook-OCR-Test. Vice versa, M1 that is fine-tuned by Excel-rendered PDFs performs worse than M2 on EUYearbook-OCR-Test. This suggests that domain adaptation by fine-tuning the pretrained TableParser with in-domain high-quality data works well.

Additionally, if we compare the $\Delta AP|(M1-M2)|$ under each test set (e.g., the differences of columns (3) and (4), of (7) and (8), of (11) and (12) in Table 2), the ΔAP on

[11] This means we evaluate the performance of M1 on the fine-tuned set for M2 (as a test set for M1) and vice versa.

*-OCR-Test in all categories is smaller than ZHYearbook-Excel-Test, with M1 already achieving medium-high performance on the test set. Although M1 is not fine-tuned by in-domain historical images, ModernTableParser is still able to parse historical scans with moderate performance. This suggests that TableParser trained on modern table structures can be used to parse the layout of tabular historical scans. Because the cost is often too high in generating a large amount of training data of historical scans (see Section 3 for the discussion of labeling efforts), our approach shows a promising direction in first developing TableParser that works well for modern tables, and then adapting TableParser to the historical domain by fine-tuning on only a few manually annotated historical scans of good quality.

5.2. Qualitative Assessment

In Figures 7, 8, 9, and 10 in Appendix C, we show the qualitative outputs of ModernTableParser and HistoricalTableParser on various types of inputs.[12] The quality of structure parsing varies across inputs, but overall, the quality is high. Even if we simply use ModernTableParser to parse old scans, it achieves a moderate performance, sometimes better than HistoricalTableParser (see Figures 9 and 10). This substantiates our claim that knowing the table structure (caption, tabular, row, column, multi-cell, etc.) is of foremost importance for parsing tables. We see that the performance of LayoutParser is quite poor on the tabular data in Figure 5 (d) using the best model from its model zoo (PubLayNet/faster_rcnn_R_50_FPN_3x).

6. Related Work

Table Annotation. TableLab [26] provides an active learning based annotation GUI for users to jointly optimize the model performance under the hood. LayoutParser [24] has also promoted an interactive document annotation tool[13], but the tool is not optimized for table annotations.

Table Structure Parsing. As pioneering works in table structure parsing, [3] and [7] have both included a review of works in table structure recognition prior to DL. Prior methods typically required high human efforts in creating the feature extraction. After [7], researchers have started to revisit table structure parsing with DL methods, which turned out highly promising compared to the rule-based (e.g., [3, 4]) and ML-based methods (e.g., [8, 10, 9]).

The success of DL has marked the revisiting of table structure parsing by [7], which inspired follow-up research [27, 1, 6, 2, 28, 29, 30, 19, 31, 18, 32, 17]. To highlight a few, [2] proposed EDD (encoder-dual-decoder) to covert table images into HTML code, and they evaluate table recognition (parsing both table structures and cell contents) using a newly devised metric, TEDS (Tree-Edit-Distance-based Similarity). [29] proposed TGRNet as an effective end-to-end trainable table graph construction network, which encodes a table by combining the cell location detection and cell relation prediction. [28] used bi-LSTM on table cell detection by encoding rows/columns in neural networks before the softmax layer. Researchers also started discussing effectively parsing tables in the wild [30], which is relevant to the perturbation tests we want to conduct for historical tables. TabCellNet by [19] adopts a Hybrid Task Cascade network, interweaving object detection and instance segmentation tasks to progressively improve model performance. We see from the previous works, the most effective methods [17, 18, 19] always jointly optimize the cell locations and cell relationships. In our work, we consider these two aspects by learning the row and column alignments in a hierarchical structure, where we know the relationship of entities in the table (row, column, cell, caption, footnote).

7. Discussion and Conclusion

7.1. Efficiency

PyWin32 uses the component object model (COM), which only supports single-thread processing and only runs under Windows. But with 20 VMs, we managed to process a large amount of files. This is a one-time development cost. On average – on the fastest machine used (with 16 GB memory, 6 cores, each of 4.8GHz max (2.9 base)) – it took 15.25 seconds to process one document (a worksheet in this case). To fine-tune a pretrained TableParser with 17 images, it takes 3-4 hours to fine-tune the model with 30k iterations.

7.2. Future Work

Based on our findings, we will further improve the parsing performance on table row/column/cell. Besides, we plan to enable a CSV-export functionality in TableParser, which allows users to export a CSV file that attends to both bounding boxes generated by the OCR'ed and the hierarchical table structure. We will also benchmark this functionality against human efforts. Another practical functionality we add to facilitate users' assessment of table parsing quality, is that we enable TableParser to compute row and column sums when exporting to the CSV format. Because tables sometimes come with row/column sums in the rendered format, this functionality can

[12] Input and annotated figures of original size can be found under https://github.com/DS3Lab/TableParser/tree/main/figures.

[13] See https://github.com/Layout-Parser/annotation-service (last accessed: Nov. 1, 2021).

help users to assess their manual efforts in post-editing the CSV output. We also plan to conduct perturbation tests of table structures and quantify the robustness of our models in those scenarios. These exercises will be highly valuable because, as we see in Figure 7, we often encounter scan images of tables where the rectangle structures cannot be maintained (the upper right corner). This brings us to another interesting research direction: how to efficiently annotate the non-rectangle elements in a table, e.g., [30] have provided the benchmarking dataset and method for parsing tables in the wild. Finally, we would like to benchmark TableParser using the popular benchmarking datasets such as ICDAR-2013, ICDAR-2019, TableBank, and PubTabNet. Note that since we develop TableParser on top of the DocParser [1], where the reported F1 score has shown superior performance of our method on ICDAR-2013.

7.3. Conclusion

We present in this work our DL-based pipeline to parse table structures and its components: TableAnnotator, TableParser (Modern and Historical), and ExcelAnnotator. We also demonstrate that pre-training TableParser on weakly annotated data allows highly accurate parsing of structured data in real-world table-form data documents. Fine-tuning the pretrained TableParser in various domains has shown large improvements in detection accuracy. We have observed that the state-of-the-art for table extraction is shifting towards DL-based approaches. However, devising suitable tools to facilitate training of such DL approaches for the research community is still lacking. Hence, we provide a pipeline and open-source code and data to invite the active contribution of the community.

Acknowledgments

Peter Egger acknowledges Swiss National Science Foundation (Project Number 100018_204647) for supporting this research project. Ce Zhang and the DS3Lab gratefully acknowledge the support from Swiss National Science Foundation (Project Number 200021_184628, and 197485), Innosuisse/SNF BRIDGE Discovery (Project Number 40B2-0_187132), European Union Horizon 2020 Research and Innovation Programme (DAPHNE, 957407), Botnar Research Centre for Child Health, Swiss Data Science Center, Alibaba, Cisco, eBay, Google Focused Research Awards, Kuaishou Inc., Oracle Labs, Zurich Insurance, and the Department of Computer Science at ETH Zurich. Besides, this work would not be possible without our student assistants: We thank Ms. Ada Langenfeld for assisting us in finding the Hungarian scans and annotating the tables; we thank Mr. Livio Kaiser for building an ExcelAnnotator prototype during his master thesis. We also appreciate the users' insights on LayoutParser [24] shared by Mr. Cheongyeon Won. Moreover, the comments and feedback from Sascha Becker and his colleagues at SoDa Labs, Monash University, are valuable in producing the current version of TableParser. We also thank Sascha and Won for providing us with various South Korean/European table scans. Finally, we thank the reviewers at SDU@AAAI22 for carefully evaluating our manuscripts and their constructive comments.

References

[1] J. Rausch, J. O. Martinez Bermudez, F. Bissig, C. Zhang, S. Feuerriegel, Docparser: Hierarchical document structure parsing from renderings, in: 35th AAAI Conference on Artificial Intelligence (AAAI-21)(virtual), 2021.

[2] X. Zhong, E. ShafieiBavani, A. Jimeno Yepes, Image-based table recognition: data, model, and evaluation, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16, Springer, 2020, pp. 564–580.

[3] T. Kieninger, A. Dengel, The t-recs table recognition and analysis system, in: International Workshop on Document Analysis Systems, Springer, 1998, pp. 255–270.

[4] A. Pivk, P. Cimiano, Y. Sure, M. Gams, V. Rajkovič, R. Studer, Transforming arbitrary tables into logical form with tartar, Data & Knowledge Engineering 60 (2007) 567–595.

[5] J. Fernandes, M. Simsek, B. Kantarci, S. Khan, Tabledet: An end-to-end deep learning approach for table detection and table image classification in data sheet images, Neurocomputing 468 (2022) 317–334.

[6] D. Prasad, A. Gadpal, K. Kapadni, M. Visave, K. Sultanpure, Cascadetabnet: An approach for end to end table detection and structure recognition from image-based documents, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 572–573.

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

[8] D. Pinto, A. McCallum, X. Wei, W. B. Croft, Table extraction using conditional random fields, in: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, 2003, pp. 235–242.

[9] W. Farrukh, A. Foncubierta-Rodriguez, A.-N. Ciubotaru, G. Jaume, C. Bejas, O. Goksel, M. Gabrani, Interpreting data from scanned tables, in: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), volume 2, IEEE, 2017, pp. 5–6.

[10] Y. Wang, I. T. Phillips, R. M. Haralick, Table structure understanding and its performance evaluation, Pattern recognition 37 (2004) 1479–1497.

[11] S. R. Qasim, H. Mahmood, F. Shafait, Rethinking table recognition using graph neural networks, in: 2019 International Conference on Document Analysis and Recognition (ICDAR), IEEE, 2019, pp. 142–147.

[12] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3431–3440.

[13] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, Advances in neural information processing systems 28 (2015) 91–99.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, International journal of computer vision 115 (2015) 211–252.

[15] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, International journal of computer vision 88 (2010) 303–338.

[16] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, P. Dollár, Microsoft coco: Common objects in context, 2015. arXiv:1405.0312.

[17] S. Raja, A. Mondal, C. Jawahar, Table structure recognition using top-down and bottom-up cues, in: European Conference on Computer Vision, Springer, 2020, pp. 70–86.

[18] X. Zheng, D. Burdick, L. Popa, X. Zhong, N. X. R. Wang, Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 697–706.

[19] J. Jiang, M. Simsek, B. Kantarci, S. Khan, Tabcellnet: Deep learning-based tabular cell structure detection, Neurocomputing 440 (2021) 12–23.

[20] M. Li, L. Cui, S. Huang, F. Wei, M. Zhou, Z. Li, Tablebank: A benchmark dataset for table detection and recognition, 2019. arXiv:1903.01949.

[21] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980–2988. doi:10.1109/ICCV.2017.322.

[22] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, R. Girshick, Detectron2, https://github.com/facebookresearch/detectron2, 2019.

[23] J. Eberius, C. Werner, M. Thiele, K. Braunschweig, L. Dannecker, W. Lehner, Deexcelerator: a framework for extracting relational data from partially structured documents, in: 22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013, He, Qi; Iyengar, Arun; Nejdl, Wolfgang; Pei, Jian & Rastogi, Rajeev, 2013, pp. 2477–2480. URL: http://doi.acm.org/10.1145/2505515.2508210.

[24] Z. Shen, R. Zhang, M. Dell, B. C. G. Lee, J. Carlson, W. Li, Layoutparser: A unified toolkit for deep learning based document image analysis, arXiv preprint arXiv:2103.15348 (2021).

[25] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[26] N. X. R. Wang, D. Burdick, Y. Li, Tablelab: An interactive table extraction system with adaptive deep learning, in: 26th International Conference on Intelligent User Interfaces, 2021, pp. 87–89.

[27] Z. Chi, H. Huang, H.-D. Xu, H. Yu, W. Yin, X.-L. Mao, Complicated table structure recognition, arXiv preprint arXiv:1908.04729 (2019).

[28] Y. Li, Y. Huang, Z. Zhu, L. Pan, Y. Huang, L. Du, Z. Tang, L. Gao, Rethinking table structure recognition using sequence labeling methods, in: International Conference on Document Analysis and Recognition, Springer, 2021, pp. 541–553.

[29] W. Xue, B. Yu, W. Wang, D. Tao, Q. Li, Tgrnet: A table graph reconstruction network for table structure recognition, arXiv preprint arXiv:2106.10598 (2021).

[30] R. Long, W. Wang, N. Xue, F. Gao, Z. Yang, Y. Wang, G.-S. Xia, Parsing table structures in the wild, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 944–952.

[31] D. Nazir, K. A. Hashmi, A. Pagani, M. Liwicki, D. Stricker, M. Z. Afzal, Hybridtabnet: Towards better table detection in scanned document images, Applied Sciences 11 (2021) 8396.

[32] S. Luo, M. Wu, Y. Gong, W. Zhou, J. Poon, Deep structured feature networks for table detection and tabular data extraction from scanned financial document images, arXiv preprint arXiv:2102.10287 (2021).

A. DocParser Mask R-CNN

For technical details of Mask R-CNN, we refer to DocParser [1]. In Figure 6, we illustrate the Mask R-CNN model used.

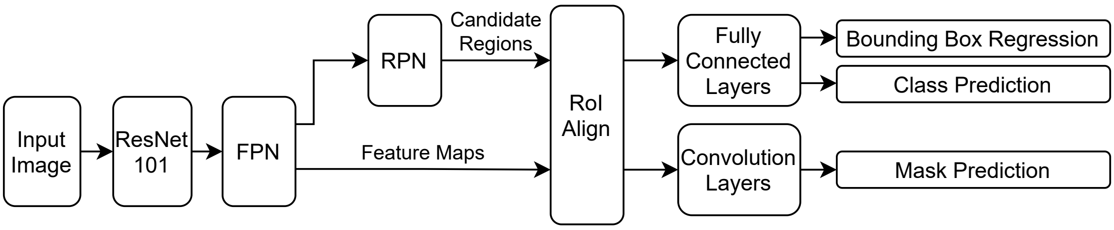


Figure 6: Mask R-CNN overview.

B. Online Resources

The source code, data, and/or other artifacts for the complete TableParser pipeline have been made available at https://github.com/DS3Lab/TableParser.

The 10-minute lightning presentation at SDU@AAAI-22 to the paper could be found under this recording, starting at 1:42:35.

C. Images for Qualitative Assessment

As we have discussed in Section 5.2, we show the qualitative outputs of ModernTableParser and HistoricalTableParser on various types of inputs in Figures 7, 8, 9, and 10.[14] The quality of structure parsing varies across inputs, but overall, the quality is high.

[14] Input and annotated figures of original size can be found under https://github.com/DS3Lab/TableParser/tree/main/figures.

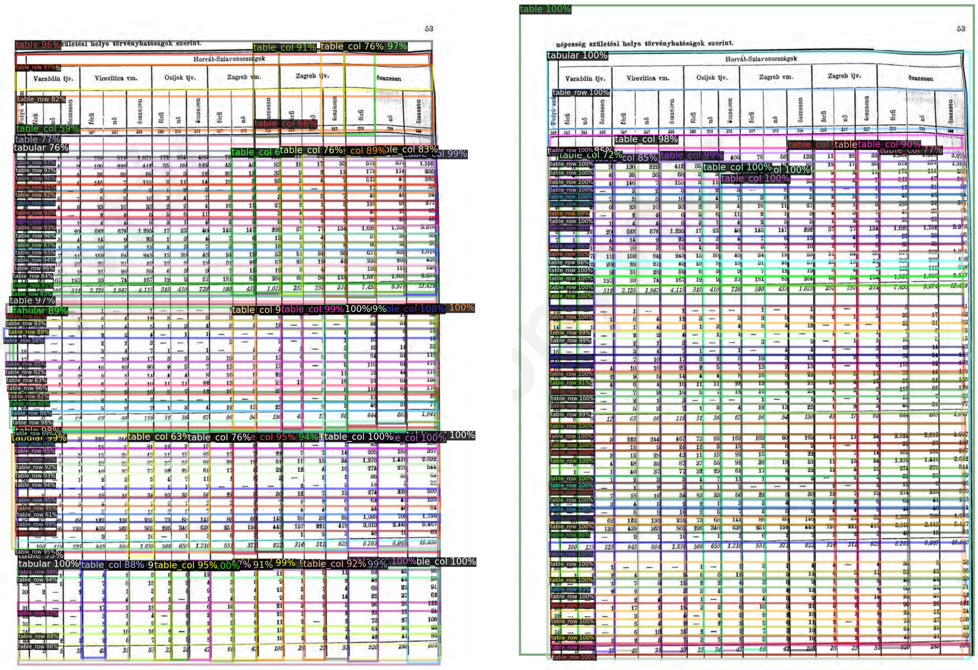


Figure 7: Hungarian table parsed by ModernTableParser (left) and HistoricalTableParser (right).

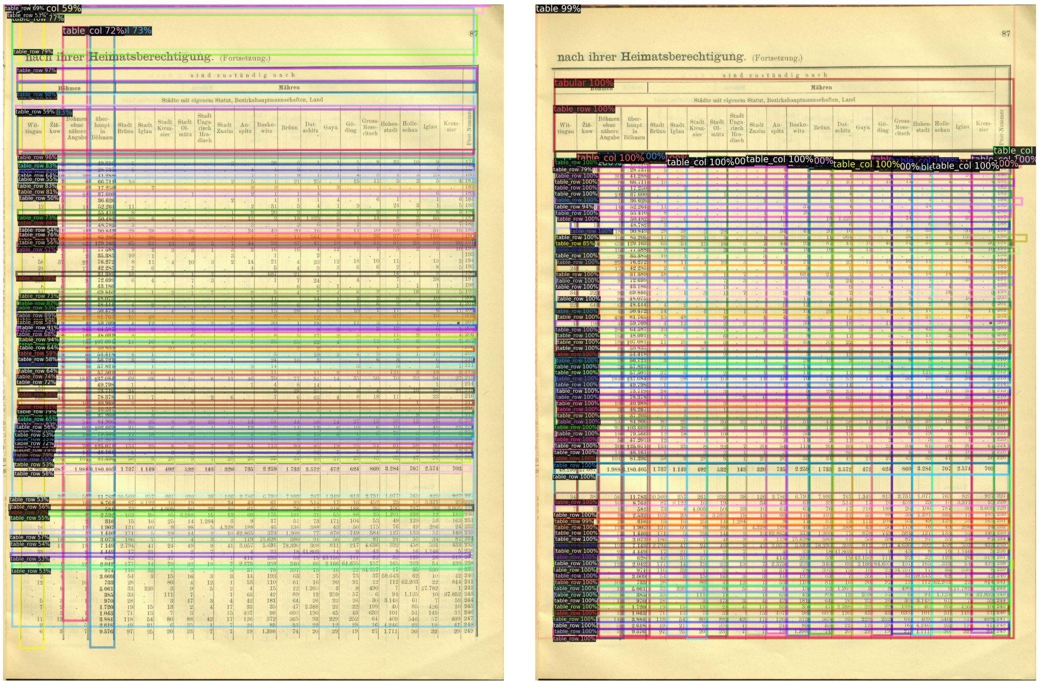


Figure 8: German table parsed by ModernTableParser (left) and HistoricalTableParser (right).

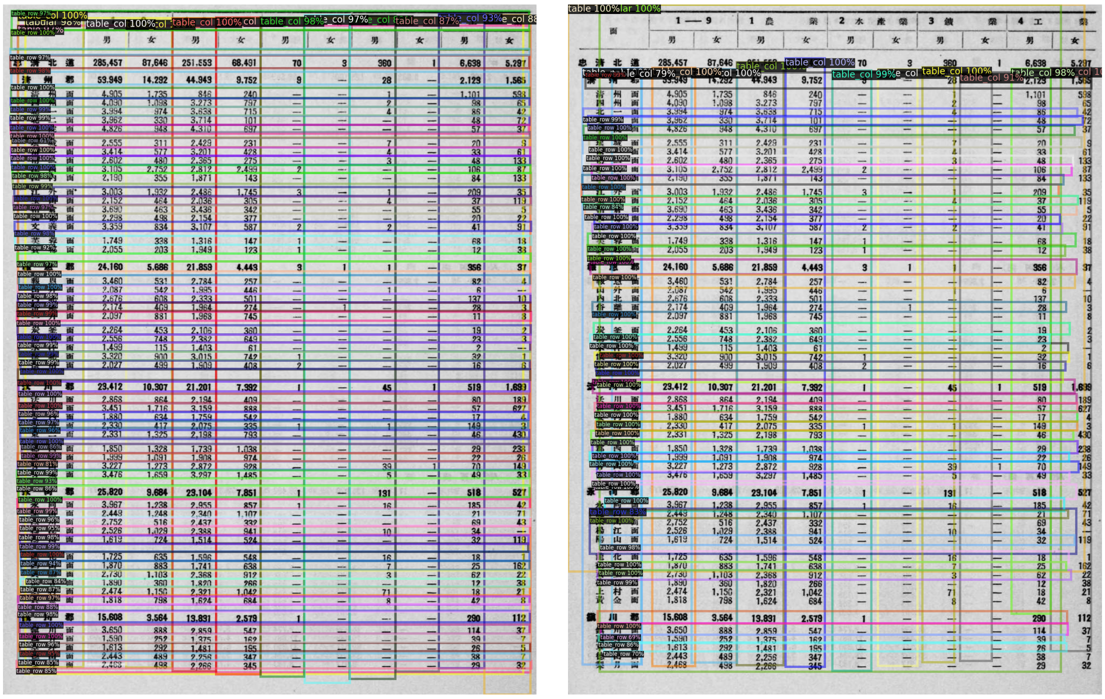


Figure 9: South Korean table parsed by ModernTableParser (left) and HistoricalTableParser (right).

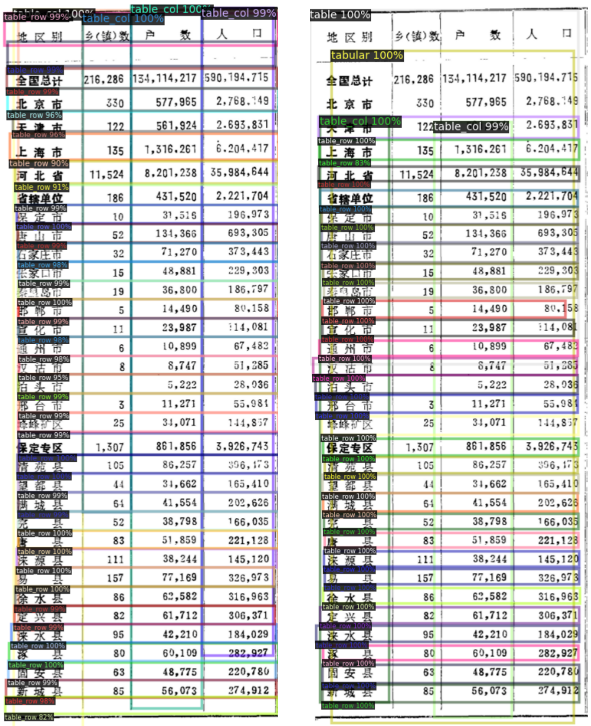


Figure 10: Chinese table parsed by ModernTableParser (left) and HistoricalTableParser (right).