Benchmarking Table Extraction from Heterogeneous Scientific Documents [Experiments, Analysis & Benchmark]

Marijan Soric
DI ENS, ENS, PSL University, CNRS, Inria
Paris, France
marijan.soric@inria.fr

Cécile Gracianne
BRGM
Orléans, France
c.gracianne@brgm.fr

Ioana Manolescu
Inria, Institut Polytechnique de Paris
Palaiseau, France
ioana.manolescu@inria.fr

Pierre Senellart
DI ENS, ENS, PSL University, CNRS, Inria
Paris, France
pierre@senellart.com

ABSTRACT

Table Extraction (TE) consists in extracting tables from PDF documents, in a structured format which can be automatically processed. While numerous TE tools exist, the variety of methods and techniques makes it difficult for users to choose an appropriate one. We propose a novel benchmark for assessing end-to-end TE methods (from PDF to the final table). We contribute an analysis of TE evaluation metrics, and the design of a rigorous evaluation process, which allows scoring each TE sub-task as well as end-to-end TE, and captures model uncertainty. Along with a prior dataset, our benchmark comprises two new heterogeneous datasets of 37k samples. We run our benchmark on diverse models, including off-the-shelf libraries, software tools, large vision language models, and approaches based on computer vision. The results demonstrate that TE remains challenging: current methods suffer from a lack of generalizability when facing heterogeneous data, and from limitations in robustness and interpretability.

PVLDB Reference Format:
Marijan Soric, Cécile Gracianne, Ioana Manolescu, and Pierre Senellart. Benchmarking Table Extraction from Heterogeneous Scientific Documents [Experiments, Analysis & Benchmark]. PVLDB, 19(1): XXX-XXX, 2026.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://gitlab.inria.fr/msoric/table-extraction-benchmark.

1 INTRODUCTION

In PDF documents, such as scientific publications, business or administrative reports, etc., tables are often used to represent data in a way that is easy to read by humans. As the number of documents increases, some containing tables with unique information, which cannot be easily found in other sources, retrieving the set of all tables from these documents is a crucial task to enable their automated analysis. This is the goal of **table extraction** (**TE**). TE is often a first step of data analysis pipelines, e.g., to answer questions over tables [3, 20], combine them within a data lake [29, 31, 48], etc.

In this paper, we propose carefully justified metrics for comparing TE methods, and evaluate them on numerous TE methods in an *end-to-end* fashion, on scientific PDF documents. Prior research has studied two core TE sub-tasks: **Table Detection** (**TD**) and **Table Structure Recognition** (**TSR**), often studied and evaluated independently [8, 27, 35, 58]. In this work, we review existing methods on heterogeneous datasets, ranging from simple rule-based libraries to complex software, including a Large Vision Language Model (LVLM) and object detection systems based on transformers.

Our main contributions are as follows: (1) We propose a new framework for TE evaluation, with formally justified, new end-to-end metrics; (2) We created and made publicly available two novel datasets for TD, TSR, and TE, with high-quality ground-truth data; (3) We compare academic methods to those widely used in practice, on three benchmark datasets. While prior work assessed TE subtasks in isolation, this paper enables end-to-end quality evaluation of TE methods.

We define the TE, TD, and TSR tasks in Section 2, also discussing the relevant literature. We describe our datasets and the methods we compare in Sections 3 and 4, respectively. We present evaluation metrics in Section 5 and our evaluation methodology in Section 6. Experimental results are in Section 7. Our benchmark is available at https://gitlab.inria.fr/msoric/table-extraction-benchmark, with additional material and further details also provided.

2 TABLE EXTRACTION

We consider a **table** extracted from a document to be a list of *rows*, each of which consists of a list of *cells*. Further, each cell has a *content*, that may be textual, numerical, mixed, etc. Note that different rows may have different numbers of cells; this is due to *merged cells*, i.e., cells spanning several rows and/or columns.

We use the following formal representation of the structure of an extracted table (see Figure 1 for illustration). Given two natural numbers $k < n$, we denote by $[k:n]$ the set $\{k, k+1, \ldots, n\}$, and write simply $[n]$ for $[1:n]$. Let $(n, m) \in \mathbb{N}^*$ be the shape of a table, i.e., its number of rows and columns. Each *cell* is defined by the tuple $(i, j, r, c) \in \mathbb{N}^4$, where (i, j) is the location of the cell, and $(r+1, c+1)$ are the number of rows and columns spanned by the cell. We require $i \in [n]$, $i + r \in [n]$, $j \in [m]$, $j + c \in [m]$. Any two

Proceedings of the VLDB Endowment, Vol. 19, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

<table>
<tr><td>(1, 1)</td><td>(1, 2)</td><td>(1, 3)</td><td>(1, 4)</td></tr>
<tr><td>(2, 1)</td><td rowspan="2">(2, 2), $r = 1$</td><td>(2, 3)</td><td>(2, 4)</td></tr>
<tr><td>(3, 1)</td><td colspan="2">(3, 3), $c = 1$</td></tr>
</table>

Figure 1: Sample table where (i, j) pairs indicate the location of each cell. When omitted, default values for r and c are 0.

distinct cells defined by (i, j, r, c) and (i', j', r', c') are disjoint, i.e., $[i : i + r] \cap [i' : i' + r'] = \emptyset$ and $[j : j + c] \cap [j' : j' + c'] = \emptyset$. Since $r, c \geqslant 0$, the location of a cell that spans multiple rows or columns is defined by the location (i, j), of their top-left corner cells. We call a cell *simple* if $r = c = 0$, i.e., the cell does not result from fusion. A table has at most be $n \times m$ cells, but there might be fewer, if some cells are merged (non-simple).

2.1 Table Extraction Tasks

Table Extraction (TE) involves detecting and recognizing a table's logical structure and content from its unstructured presentation in a document. TE can be decomposed in subtasks, as follows.

Table Detection (TD). This consists in *detecting* tables in an input document (PDF document, or page given as a bitmap image). We assume tables are *aligned with the axes* of the pages, and might have been *rotated* by ±90 degrees. Typically, a table is detected as a **rectangular bounding box** (x_0, y_0), (x_1, y_1), where x_0, y_0, x_1, and y_1 are the coordinates of the top left corner and bottom right corner of the bounding box, respectively. Note that some TE models do not output bounding boxes, but only **HTML tables**.

Table Structure and Content Recognition. Table Structure Recognition (TSR) consists in *recognizing the structure of a table in terms of rows, columns, and cells.* The input can be a cropped table image or a page image with detected table bounding box. Table Content Recognition (TCR) *identifies the content (characters) within each cell.* Clearly, TCR depends on TSR for the identification of cells; the two tasks are often performed together. Thus, by a small abuse of language, and for coherence with prior work, we will denote *both tasks* by the TSR acronym. At the end of TSR, the table is fully captured in the structured format introduced earlier, and directly rendered in HTML with its rows, cells, spans, and content.

2.2 Prior datasets and benchmarks

The literature comprises several benchmarks for TD and TSR, the latter with or without TCR. In 2013, a TD competition was introduced based on a small dataset (128 images) [27]. TableBank [35] proposed a larger dataset for TD and TSR, but does not include TCR. SciTSR [8] and FinbTabNet [58] focused on TSR only with respectively scientific papers and financial reports. The largest dataset available is PubTables-1M [57] (2021) that counts one million tables for TD and TSR. Most recent datasets like TabRecSet [64] include images from real-world scanned documents.

Existing benchmarks tackle *TD and TSR as two independent problems* and *do not evaluate the quality of end-to-end TE methods.* Consider a two-step TE method involving two models where the input of the TSR model is the output of the TD model. While TSR datasets provide the same cropped table images to different models for evaluation, in downstream tasks, the performance of the TSR model may be artificially degraded by the output of the TD model; this phenomenon is disregarded evaluating TD and TSR separately. However, downstream TE applications are impacted by the *end-to-end* TE result quality. This is why we need an **end-to-end benchmark**, and thus **new metrics** for evaluating the end-to-end model synergy.

Another limitation of existing benchmarks is that *they only consist of positive instances*, i.e., each sample includes at least one table. As we will show, this *inflates precision scores* and reduces their relevance for real-world performance

Existing TE methods. An end-to-end TE model processes a document (or a page) to extract tables. Early heuristics methods relying on hard-coded patterns were limited to specific table types [7, 17, 25, 26, 52, 61]. Machine Learning (ML) later improved generalization [14, 23, 28, 30] but required heavy feature engineering, which was eventually surpassed by Deep Learning (DL) models [4, 53]. Thanks to larger training datasets, deeper networks and more efficient architectures, these now achieve superior accuracy and versatility. Notably, [22] used CNN for TD, and DeepDeSRT [54] was the first to use Faster R-CNN-based models for TD and TSR. Some models are focused on certain data types, for example scans of paper documents with distortion [42], others on robustness. We present in detail state-of-the-art methods in Section 4.

3 OUR BENCHMARK

We have designed a benchmark for evaluating the quality of end-to-end TE methods. Our benchmark is based on three datasets of scientific documents typeset in PDF. Each of which contains: **as input**, document pages (both as PDF files and as individual rasterized images) with coordinates of each word appearing within the page (such coordinates can be obtained by standard PDF or image analysis libraries such as PDFAlto and PDFMiner[1]); **as output**, every table identified within each page with bounding box coordinates, as well as content and structure formatted as HTML markup.

First, we used a subset of the largest prior TD/TSR dataset (PubTables-1M), enriched with additional information (see below). However, this dataset **lacks diversity**, as most tables share similar templates and appearance. To address this, we generated a **heterogeneous dataset of equivalent size** ("Table-arXiv"), and manually built a **domain-specific** one ("Table-BRGM") from geological scientific reports. Its interest is that *its layout significantly differs from the others*, mainly because these were mostly produced with a Word processing software rather than a typsetting system such as LaTeX. Table 1 shows dataset statistics; we detail them next.

PubTables-Test. PubTables-1M is a dataset of 1M tables from scientific articles in PubMed Central Open Access [24]. We used the raw test dataset of PubTables for evaluation, and additionally retrieved PDF files from PubMed from which images from the dataset were taken. Since not all PDFs were available in PubMed, we used a subset of the initial dataset composed of papers with valid PDF files and that could be matched to the individual images. As the original dataset only contains table structure coordinates but not structured table content, we generated ground truth tables as HTML markup. All documents come from the biomedical domain and tend to be fairly homogeneous in the way tables are typeset; based on PDF metadata, most documents were produced using professional publishing systems such as Adobe Indesign.

[1] https://github.com/kermitt2/pdfalto, https://github.com/pdfminer/pdfminer.six

Table 1: Summary of datasets used in our benchmark. (L: LaTeX, W: Word, P: Professional publishing system, O: Other)

Dataset	Topic	Language	% Type				Count			
			L	W	P	O	# PDF	# Pages	# Positives	# Tables
PubTables	Biomedical, life sciences	English	0	1	85	14	23 175	46 942	46 942	55 990
Table-arXiv	Math, physics, astrophysics	English	100	0	0	0	2 443	36 869	5 214	6 308
Table-BRGM	Geological reports	French, English	33	67	0	0	6	499	91	124

Table-arXiv. To ensure heterogeneity within our data, we have also automatically generated "Table-arXiv", a new dataset of 36k-samples dataset built from 2 443 LaTeX source files from arXiv[2] preprints for ground truth generation. These sources were sampled from *the last 20 years* and *across all arXiv domains* to ensure **heterogeneity** in the data. Using the LaTeX source code, we automatically generated TD and TSR (with TCR) annotations: for TD, by instrumenting the commands starting and ending a `tabular` environment to add anchors that allow tracking their locations within the generated PDF; for TSR, we used LaTeXML[3] to generate HTML markup tables from the source file.

Table-BRGM. We have constructed a last dataset, denoted as "Table-BRGM", that is domain-specific and comprises French and English geological reports sourced from a selection of BRGM documents – BRGM is France's reference public institution in Earth sciences. The documents contain geological survey reports and records. We manually annotated this dataset for TD and TSR (for TSR, by first employing an end-to-end model and then manually correcting its output). Reports mainly come from documents produced by Word processing software, but some also originate from LaTeX-compiled documents. The dataset also contains **heterogeneous** tables: large tables, tables with or without borders, empty cells, merged cells, etc., in English and French.

4 END-TO-END METHODS

Table 2: End-to-end methods for table extraction

Method	Input type	TCR	Output			Steps
			Location	Conf.	HTML	
Camelot	PDF	PDFMiner	✓		✓	1
PyMuPDF	PDF	PyMuPDF	✓		✓	1
PDFPlumber	PDF	PDFMiner	✓		✓	1
Grobid	PDF	PDFAlto	✓		✓	1
LVLM	Image	LVLM			✓	1
Docling	PDF	EasyOCR	✓		✓	2
TATR-extract	Image	PDFAlto	✓	✓	✓	2
XY+TATR-extract	Image	PDFAlto	✓	✓	✓	2
VGT+TATR-structure	Image	PDFAlto	✓	✓	✓	2

We tested various methods, ranging from simple Python libraries, to well-established extraction platforms, a large vision language model (LVLM) and computer vision-based methods. Some (but not all) methods output a confidence score along with their predictions. Table 2 summarizes the methods; Conf. stands for confidence.

4.1 Baseline Models (without confidence scores)

Our first four models, referred to as "baseline models", output table predictions but no uncertainty measure.

4.1.1 *Commonly used Python libraries.*

We selected popular TE Python libraries on Github: PDFPlumber, PyMuPDF and Camelot, with respectively 8.1k, 7.8k and 3.4k stars.

PDFPlumber [55] provides detailed information about each text object in a PDF file; it extract tables through rule-based heuristics. Inspired by Tabula[4] and [12], it leverages the *lines* that are explicitly drawn in the PDF, or implied by the word alignment on the page. While widely used, PDFPlumber only handles simple layouts. It lacks robust support for complex, multi-column, or merged-cell table structures, commonly found in real-world documents.

PyMuPDF[5] allows to analyze, extract and convert PDF (and other) documents. For TE, it leverages the presence or absence of lines, rectangles or other supporting vector graphics.

Camelot[6] also leverages rules for TE. It is widely adopted for its simplicity and ease of integration. However, it may struggle with complex layouts featuring merged cells or inconsistent formatting.

Another similar library is PDFTables[7] but it is no longer maintained, thus we do not include it. We also experimented with commercial tools, such as Docsumo[8]. Extraction quality was very uneven, thus we considered that it did not justify its cost, e.g., Docsumo charges 89 USD for analyzing 1 000 pages.

4.1.2 *Grobid.*

GeneRation Of BIbliographic Data [40] (Grobid[9]) is a machine learning library for extracting, parsing, and re-structuring raw documents such as PDF into structured XML/TEI-encoded documents, with a particular focus on technical and scientific publications; it is used in production, e.g., on France's preprint server HAL. Grobid structures the document's text body into paragraph, section titles, reference, figures, *tables*, etc., which it represents as XML, e.g., tables appear as: `<table>`, `<row>`, and `<cell>` elements; table coordinates are provided as attributes.

4.1.3 *LVLM.*

Next, we consider a multimodal (text and image) Large Vision Language Model (LVLM). We chose OpenAI's "GPT-4o mini" [46] since it is recent, popular, reported to perform very well on multimodal tasks, and cost-efficient (0.29 USD for 1 000 pages). We will denote it simply by LVLM below, and use it in zero-shot mode for TE. Given an image and a prompt, the LVLM outputs an HTML table; it does not provide pixel coordinates. As we will discuss in Section 6.1, this raises a challenge when evaluating the method (the gold standard does include coordinates). Also, we cannot guarantee the absence of contamination: the LVLM may have been trained on some datasets from the benchmark.

[2] https://arxiv.org/
[3] https://github.com/brucemiller/LaTeXML
[4] https://tabula.technology/
[5] https://github.com/pymupdf/PyMuPDF
[6] https://github.com/camelot-dev/camelot
[7] https://github.com/pdftables/python-pdftables-api
[8] https://www.docsumo.com/
[9] https://github.com/kermitt2/grobid

4.1.4 Docling. Docling [2] is an advanced Python library for PDF document processing and understanding (40.1k stars on Github). Docling extracts tables in two steps: it leverages RT-DETR [66] for TD, and TableFormer [44] for TSR, while relying on EasyOCR[10], a third-party OCR library, for TCR. Even though Docling uses an object detection model for TD, it does not provide table confidence scores for this task.

RT-DETR finds different layout components (table, text, title, picture, caption, footnote, formula, etc.) for each page. RT-DETR is based on DETR, which will be explained in more detail in Section 4.2.1. RT-DETR adds an efficient hybrid encoder that processes multi-scale features (inspired by Deformable-DETR [70]); and the uncertainty-minimal query selection (inspired by DINO [65]) that improves the quality of initial object queries. This model has been re-trained on the DocLayNet dataset [51], gathering diverse data sources to represent a wide variability in layouts.

TableFormer [44] is a vision-transformer model that identifies the structure of a table by using a custom structure token language [41], starting from an image of the table. The model architecture is composed of a CNN backbone network that encodes the image which is then provided to the encoder producing features that represent the input image. Then, the features are then passed to both the transformer decoders: Structure Decoder and Cell BBox Decoder. The former generates the logical table structure as a sequence of tokens (HTML tags), while the latter predicts simultaneously bounding boxes of table cells. This TSR model handles many characteristics of tables, such as partial or no borderlines, empty cells, rows or columns, cell spans, tables with inconsistent indentation or alignment and other complexities.

4.2 TATR-extract

Next, we use a model that output confidence. We refer to this type of model as "probabilistic models" (even though confidence scores are not always probabilities).

4.2.1 DETR. TAble TRansformer (TATR) [57] is composed of two models: one for TD, and the other for TSR. Both models use the DEtection TRansformer (DETR) [6] architecture.

DETR is an encoder–decoder transformer model for object detection tasks. The architecture is divided into two parts: a backbone model (CNN) which extracts image features, fed into a transformer. Its neural architecture enables DETR to predict all objects at once (in parallel). In contrast with prior approaches, e.g. [39, 60, 69], DETR does not need to be given "anchors" (regions where to look for candidate objects to extract). Instead, DETR learns a small, fixed number of position encodings, so-called *object queries*, and uses them for extraction.

DETR outputs a fixed number of predictions that include bounding boxes, object labels and confidence score distributions. *TATR-detect* is the TD model, abridged to *TATR-d* in Figure 2; its output feeds the *TATR-structure (TATR-s)* model which performs TSR.

4.2.2 TATR-extract pipeline. Given a page as an image, TATR-detect performs object detection on the image with 3 classes: "table", "table rotated" and "no object". It outputs 15 predictions, each with bounding box coordinates and confidence scores for each class. In the original inference implementation, only the label with the highest confidence score was retained, dropping predictions classified as "no object" even if their confidence for "table" or "table rotated" was relatively high (while lower than that of "no object"). In our experiments, this sometimes lead to poor performance. To mitigate this loss of information, we modify TATR-extract to retain all predictions where the confidence score for either "table" or "table rotated" exceeds a threshold of 0.05. This adjustment ensures that potential table detections are preserved, resulting in a refined set of predicted tables. We then apply *Non-Maximum Suppression (NMS)* [15] to eliminate overlapping table predictions[11]. We further filter out predicted tables that *lack any tokens within their bounding boxes*, excluding empty detections. We use PDFAlto to extract the tokens in a document. However, for tables embedded as bitmap image (tables as images) in a PDF, PDFAlto is unable to extract the tokens present because images have no extractable text/lines in PDFs. Such tables are not retained if detected.

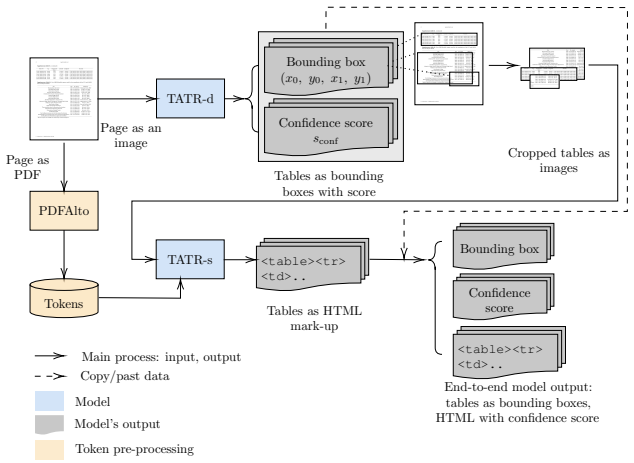

Figure 2: TATR-extract pipeline for table extraction

For each table , TATR-structure then processes the cropped image and associated tokens. The image is padded with white pixels to align with the training data format used for DETR. In each resulting image, TATR-structure performs object detection across seven table structure classes ("table", "table column", "table row", "table column header", "table projected row header", "table spanning cell" and "no object"). Several post-processing steps are applied on the output of TATR-structure to obtain an actual table, also complete with textual cell content. First, TATR-structure outputs overlapping bounding boxes (columns and rows), which are modified to obtain aligned cells. Further, to identify the tokens in each table cell, we compare the tokens' bounding boxes with those of the cells. Finally, tables are exported into HTML mark-up.

4.3 XY-cut+TATR-extract

Next, we introduce a new probabilistic model, XY-cut+TATR-extract, based on TATR-extract.

We noted that TATR-detect was sometimes wrong on pages containing multiple tables: either some were missed, or, when recognized, their confidence scores were low. As explained in Section 4.2.1, the bounding boxes used for prediction come from the

[10] https://github.com/JaidedAI/EasyOCR

[11] Vanilla TATR did not require NMS because it kept only the most likely label (a table or "no object"). Our choice to prioritize "table" predictions makes NMS necessary.

transformer decoder output. Object queries (decoder input) are positional embeddings learned during training, which contain abstract information about spatial locations, or "where should the model look?" Thus, each output, which is directly associated to an object query, is roughly specialized in an area. To mitigate this, we pre-process each page input to TATR-extract, by isolating layout components that are isolated by white space; for this, we rely on the algorithm XY-cut [21].

XY-cut works as follows. By counting the number of black pixels[12] along each axis, X and Y, we obtain profiles of pixel distributions, which are used to isolate rectangle components to form sub-images. This process is recursively applied until stop criteria are met (a minimum area is found). Each sub-image is then fed as input to TATR-detect. This allows to help the model focus on individual areas where many tables are unlikely (these are usually separated by white areas, which XY-cut identifies).

Within XY-cut+TATR-detect, instead of keeping 15 predictions (as in for TATR-extract), we keep only **the two most confident predictions**, as XY-cut should already have split the tables into separate images. Indeed, image chunks issued by XY-cut contain almost no white space gap; they are titles, paragraphs, tables, figures, etc. As in Section 4.2, we apply NMS, and eliminate empty tables (those containing no symbols) before running TATR-structure.

4.4 VGT+TATR-structure

Our final probabilistic method follows a pipeline similar to that of TATR-extract, where TATR-detect is replaced with a different component, namely VGT (discussed below).

Document Layout Analysis (DLA) consists of transforming documents into structured representations, on which information extraction can be performed. Some DLA approaches are based on visual information (like TATR-detect, that only performs TD and not the whole layout segmentation), but more recently, models also use textual information[13] from documents. Vision Grid Transformer [9] (VGT) is a DLA tool. In contrast with TATR, which only uses visual information, VGT also relies on textual information, which it views as a 2D grid. This model achieves state-of-the-art results with the best Average Precision score on TD over the PubLayNet [68] dataset. VGT has two parts:

(1) A Vision Transformer (ViT), which models visual information, inspired by ViT [11] and DiT [34]. The input image is split into non-overlapping patches that are flattened and linearly projected to get a sequence of tokens embeddings that is then fed into ViT.

(2) A Grid Transformer (GiT), that models 2D language information. The input image is seen as a 2D token-level grid which feeds GiT (architecture similar to ViT). GiT has been pre-trained with Masked Grid Language Modeling (MGLM), inspired by BERT's Masked Language Modeling [10] in order to learn better token-level semantics for grid features.

Overall, the framework aims to generate better embeddings from the input image page, in order to feed detection framework that performs object detections using 6 classes ("title", "text", "figure", "table", "list" and "no object"). The detection framework is the Cascade R-CNN [5] detector, which is implemented based on the detection library Detectron2 [62].

The VGT+TATR-structure pipeline combines the VGT table detector, with the TATR-structure recognition model. For each table prediction (bounding box with a confidence score) output by VGT, we do not know whether it is *rotated* or not, yet that matters in order to use TATR-structure, that takes as input only non-rotated tables. To find out whether a table is rotated, we simply look at the average shape[14] (length and width) of the tokens within the table. Then we use the same post-processing as for TATR-extract (rotate tables, remove empty tables) as discussed in Section 4.2.2.

5 METRICS

In this section, we present metrics used for TD, TSR and TE.

We first introduce some notations. From now on, *we highlight the first occurrence of each notation encasing it in a box, for quick reference.* Let $\boxed{\mathcal{P}}$ be the set of TE model outputs: each of its element is a tuple that includes a bounding box, an HTML table representation, and if available, a confidence score c_{table}. When the models lack a confidence score, we consider $c_{\text{table}} = 1$ in all $\mathcal{P}$ entries.

We define $\boxed{\mathcal{P}^+} \subseteq \mathcal{P}$ as the set of positive predictions: those that we consider to be actual tables. For the simple baselines where $c_{\text{table}} = 1$, we have $\mathcal{P}^+ = \mathcal{P}$. For probabilistic models, where confidence scores can be interpreted as table probabilities, we use a threshold $\boxed{\theta_c}$ to define positive predictions. This threshold draws a decision boundary: only predictions with scores above it are counted as positive. In this case, we set $\boxed{\mathcal{P}_{\theta_c}}$ to be $\{(\hat{y}, c) \in \mathcal{P} \mid c_{\text{table}} > \theta_c\}$, with confidence score threshold $\theta_c \in [0, 1]$.

5.1 Table detection (TD)

TD performance can be evaluated using traditional metrics inspired from Information Retrieval. Specifically, a True Positive (TP) is a table correctly recognized, a False Positive (FP) is a table found by the model where a human user would not consider a table exists, and a False Negative (FN) is a table recognized by human users but missed by the model. Given a table predicted by the model, and a table in the gold standard, to identify TPs, we rely on the Intersection-over-Union (IoU) metric between the areas of the prediction, and the gold standard tables. If IoU is above a given threshold $\boxed{\theta_J} \in [0, 1]$, the prediction is counted as a TP. For a given TD method, $\boxed{\mathcal{P}^{++}} \subseteq \mathcal{P}^+$ denotes the set of true positive predictions.

5.1.1 Probabilistic metrics. The value $\theta_J = 0.5$ is a natural choice, because if we consider non-overlapping predictions, we can get only one prediction with IoU greater than 50% (TP) per ground truth. However, different θ_J values are also sometimes used. For instance, the ICDAR 2019 competition cTDaR [16] computes F_1-scores with $\theta_J \in \{0.6, 0.7, 0.8, 0.9\}$; then, they compute $\text{WAvg}(F_1)$, which is a weighted average of F_1-scores for these four θ_J values, with weights proportional to their thresholds. Thus, the F_1-score based on higher thresholds are given more importance.

Let us discuss the mathematical basis for choosing such a set of θ_J values. A general, principled way to compute a threshold-dependent metric X_θ, such as Precision, Recall or F_1-score, is an *integral* over the interval comprising all possible θ_J values. $\text{WAvg}(X)$ can be

[12] Assuming PDF reports are written in black on a white background.

[13] TATR-structure uses token positions only in post-processing, not during the actual TSR task (row, columns, header, etc.)

[14] Most of the time, tokens are multi-character words that are longer than they are wide (counting pixels).

seen as a *discrete approximation* of such an integral. Further, the weighted average can be interpreted as the *expected value of the random variable (RV)* X_θ according to a *piecewise linear probability density function* (*pdf*, in short) f_s, where s is a parameter between 0 and 1 and $f_s : x \mapsto \alpha_s \cdot x \cdot \mathbf{1}_{[s,1]}(x)$. Here, $\mathbf{1}_{[s,1]}$ is the *indicator* function whose value is 1 on $[s, 1]$ and 0 everywhere else, and $\alpha_s > 0$ is a normalization factor to ensure f_s is a *pdf*.

Now, let us view θ_J as a real RV with values in $[0, 1]$. As *pdf*, we chose two different functions: f_0 the simplest and most naive version, and $f_{0.5}$ that generalizes WAvg. After some normalization, we get $f_0(\theta) = 2 \cdot \theta \cdot \mathbf{1}_{[0,1]}(\theta)$ and $f_{0.5}(\theta) = \frac{8}{3} \cdot \theta \cdot \mathbf{1}_{[0.5,1]}(\theta)$. For a random variable θ_J distributed according to *pdf* f, we write $\theta_J \sim f$.

Definition 1 (Expected Precision and Recall)**.** Let $\boxed{P_{\theta_J}, R_{\theta_J}, F_{1,\theta_J}}$ be the Precision, Recall and F_1-score obtained with the random variable threshold θ_J. Under the assumption that θ_J is sampled according to the probability distribution function f, we have:

$$\mathbb{E}_{\theta_J \sim f}[P_{\theta_J}] = \frac{1}{|\mathcal{P}^+|} \sum_{i \in \mathcal{P}^+} \mathbb{E}_{\theta_J \sim f}[\mathbf{1}_{[J_i > \theta_J]}]$$

$$\mathbb{E}_{\theta_J \sim f}[R_{\theta_J}] = \frac{1}{|\mathcal{G}|} \sum_{i \in \mathcal{P}^+} \mathbb{E}_{\theta_J \sim f}[\mathbf{1}_{[J_i > \theta_J]}]$$

where $|\mathcal{G}|$ is the number of ground truth tables and J_i the IoU for the positive sample i.

From now on, we use $\boxed{\mathbb{E}_s[X]}$ as shorthand for $\mathbb{E}_{\theta_J \sim f_s}[X_{\theta_J}]$, for some $s \in [0, 1]$ and metric $X_{\theta_J} \in \{P_{\theta_J}, R_{\theta_J}, F_{1,\theta_J}\}$. Note that $\mathbb{E}_s[X]$ becomes stricter as s increases.

5.1.2 Average precision. For probabilistic model, the set of positive predictions $\mathcal{P}^+ \in \left\{\mathcal{P}_{\theta_c}\right\}_{\theta_c}$ depends on the choice of a confidence score threshold θ_c, thus we obtain a set of corresponding scores $\{X_{\theta_c}\}_{\theta_c}$. To aggregate over several θ_c values, in the object detection literature, the Average Precision metric is used. Authors of [38] introduced performance metrics for object detection: *Average Precision (AP)* at multiple IoU threshold values (0.50, 0.75, or a range), and with AP_{θ_J} being the main metric.

For probabilistic models that output a confidence score c_k on each prediction k (that is, bounding box plus label), we can rank predictions in the increasing order of their confidence score. Let $\left(c_{(k)}\right)_k$ be the ordered sequence. Thus, we can compute Precision and Recall, denoted as $\boxed{P^{\theta_c}_{\theta_J}}$ and $\boxed{R^{\theta_c}_{\theta_J}}$, at confidence score θ_c (where the positive predictions are elements from $\mathcal{P}^+_{\theta_c}$) and at matching threshold θ_J. We obtain a list of pairs $\left(\left(P^{c_{(k)}}_{\theta_J}, R^{c_{(k)}}_{\theta_J}\right)\right)_k$ that defines a *Precision–Recall curve* where the horizontal coordinate is given by R and whose y values form a *"staircase"* dictated by the values $P_{\theta_J}(R)$ (see Section 7). We use the Scikit-learn library [50] which computes AP_θ instead of the Pascal VOC metric [13] to account for the entire recall range, rather than focusing on specific recall threshold.

5.1.3 Estimating model confidence via its (mis)calibration. The metrics used above (X_{θ_J}, $\mathbb{E}_s[X]$ and AP_{θ_J}) rely on binary decisions, where each prediction is either positive or negative. However, a more finer-grain evaluation should account not only for these binary choices but also for the associated confidence scores.

A model is said to be **calibrated** when its probabilities (confidence scores) are aligned with the real-world outcomes [19]. A calibrated model is desirable because its predictions can be seen as more trustworthy. Formally, model calibration is defined by:

$$\mathbb{P}[\widehat{Y} = Y \mid \widehat{P} = p] = p$$

where $[\widehat{Y} = Y]$ is the event "the model prediction is right" and $[\widehat{P} = p]$ is the event "the confidence score (seen as a RV) is equal to p", for $p \in [0, 1]$. If this is the case, the random variable $\widehat{P}$ perfectly reflects the prediction trustworthiness. Based on this definition, the miscalibration of a classifier can be measured by the *expected calibration error (ECE)* [43].

However, in the case of object detection, we cannot enumerate $[\widehat{Y} = Y]$, but only $[\widehat{Y} = 1 \wedge Y = 1]$, because there is a multitude of negative locations (possible bounding boxes that do not contain a table). Thus, for an object detection model, calibration is measured with respect to the precision [32], leading to *the object detection version of ECE*, or $\boxed{\text{D-ECE}}$, in short:

$$\text{D-ECE} = \mathbb{E}_{\widehat{P}}\left[\left|\mathbb{P}\left[\widehat{Y} = Y = 1 \,\middle|\, \widehat{P} = p\right] - p\right|\right] \tag{1}$$

To compute D-ECE efficiently, (2) approximates (discretizes) (1) by partitioning predictions into M equal bins, and taking a weighted average of the bins' (Precision − Confidence) gap. n is the number of predictions, and B_m is the set of indices of predictions whose confidence falls between $\frac{m-1}{M}$ and $\frac{m}{M}$. The prec and the conf functions compute the average precision and confidence, respectively:

$$\text{D-ECE} \approx \sum_{m=1}^{M} \frac{|B_m|}{m} \left|\text{prec}(B_m) - \text{conf}(B_m)\right| \tag{2}$$

We compute this score over $\mathcal{P}$ to obtain an overall calibration error.

5.2 Table structure recognition

To evaluate TSR result quality, we need to check that the predicted table has the correct table shape (topology, structure), and that cells have the expected content.

A naive approach for TSR evaluation could be to tackle it as an object detection task by comparing predicted bounding box cells with Ground Truth (GT) cells by their absolute positions. Such an approach is used to evaluate TATR-structure in [57]. This approach is vulnerable to *shifts*: a slight shift along one or both coordinates of a predicted bounding box cell would disproportionately penalize the models. However, essential information describing the table structure is in the relative structure/position of cells, rather than their absolute coordinates. Therefore, we prefer metrics that rely on the structure, not on absolute coordinates.

The 4-gram BLEU score [47] (BLEU-4) metric can be applied by comparing HTML tables as strings. It allows comparing at the same time, markup structure (tags) and content between predictions and GT. However, BLEU-4 evaluates exact (not partial) match cell correctness, and does not capture the accuracy of the table's spatial representation (cells under/above one another, etc.) The DAR metric [18] (Directed Adjacency Relation) is based on immediate neighbors: the relative positions of non-empty cells (which cells are adjacent to a given cell). However, it does not capture the global 2D structure: a table can be locally consistent but globally wrong, notably regarding the overall placement of cells relative to the whole table and table rotation.

Thus, we chose to use two metrics: GriTS [59], which evaluates tables directly in their matrix form and computes a similarity between these matrices, and TEDS [67] which compares tables seen as trees of HTML tags. We describe these metrics below and show that they satisfy our requirements.

5.2.1 TEDS. The metric Tree-Edit-Distance-Based Similarity (TEDS) measure views each HTML table as follows. There is a root `<table>` with children `<tr>` (nodes are table rows). The leaves of the tree are cells `<td>` with attributes `colspan`, `rowspan` and `content`. Alternatively, if a model is capable of *functional analysis*, i.e., distinguishing *header* cells from the non-header ones, we could use `<thead>`, respectively `<tbody>` to describe parts of a table.

TEDS builds upon the previous Tree-Edit Distance [49], in turn inspired the Levenshtein Distance [33]. The latter counts the minimum number of insertions, deletions, and substitutions to transform a string into another string. Similarly, TEDS counts the node deletions, insertions, relabeling and content modifications needed to transform a tree (HTML table) into another, as follows: the cost to *insert or delete* and *relabel* a node is 1; the cost of *relabeling a cell content* is the normalized Levenshtein similarity between the two strings. Formally, for trees T_P, T_{GT}: $\text{TEDS}(T_P, T_{GT}) = 1 - \frac{\text{EditDist}(T_P, T_{GT})}{\max(|T_P|, |T_{GT}|)}$ where $|\cdot|$ is the number of nodes.

5.2.2 GriTS. The Grid Table Similarity (GriTS) family of metrics distinguishes three levels at which table structure can be identified: *Topology*, describing the rows and columns each cell spans over, in a two-dimensional grid; *Content*, which refers to the textual content within each cell; and *Location*, denoting the rectangular span in a pixel matrix that is occupied by each cell. This leads to *three GriTS similarity metrics* for Topology, Content, and Location respectively. We ignore the one based on Location (coordinates) since it suffers from the shift problem described above. Each metric takes as input two matrices of cells, the prediction and the GT, and computes their *2-dimensional Longest Common Subsequence* (2D-LCS, in short). A GriTS metric is computed in three steps:

(1) Generate the matrices P, G (prediction, GT) from tables. When computing a Topology metric, each entry (matrix cell) is a value in $\mathbb{Z}^4$; for the Content metric, an entry belongs to $V^{\mathbb{N}}$ (strings). We use the notation E to refer in a general way to the domain of entries.

(2) Identify *the two substructure matrices (see below)* $\widetilde{P}$ of P and $\widetilde{G}$ of G (of the same shape between themselves) *that are the most similar according to* f, the penalty function between the grid cells' properties. Formally, we call *the set of substructures of a matrix* M with rows R and columns C, denoted $\mathfrak{M}$, the set of matrices M' that have rows R' and columns C' respectively included in R and C.

$$\left(\widetilde{P}, \widetilde{G}\right) \in \underset{\substack{(P',G')\in\mathfrak{P}\times\mathfrak{G} \\ P',G' \text{have same dimensions}}}{\arg\max} \sum_{i,j} f(P'_{i,j}, G'_{i,j})$$

where $\mathfrak{P}, \mathfrak{G}$ are the set of substructures of P, respectively, G. We call $\widetilde{P}, \widetilde{G}$ the *two-dimensional most similar substructures* (2D-MSS, in short) of the matrices P and G.

(3) We compute a similarity score between $\widetilde{P}$ and $\widetilde{G}$, denoted $\text{GriTS}_f(P, G)$, using a *penalty function* f which depends on the subtask (Topology, Content), as explained below.

The *entry-wise penalty function* $f\colon E \times E \to [0, 1]$ measures the *partial correctness* between two entries with the same coordinates;

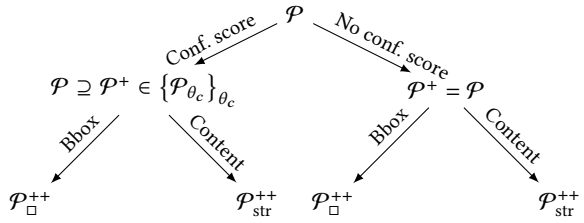

Figure 3: TSR inputs depending on the TD model type (with or without a confidence score), and evaluation type (*bbox*-based or *content*-based).

It replaces the 0 or 1 value used for the vanilla 2D-LCS computation [1]. Concretely, f is IoU for Topology and LCS for Content.

The overall GriTS score is defined as:

$$\text{GriTS}_f(P, G) = 2\frac{\sum_{i,j} f(\widetilde{P}_{i,j}, \widetilde{G}_{i,j})}{|P| + |G|}$$

where $|\cdot|$ is the matrix size (number of grid cells). Recall that f is a penalty function specific to the distance being computed.

We compute the GriTS Topology and Content metrics from the HTML markup table representations.

5.2.3 HTML markup vocabulary to be used in our evaluation. One last aspect needs to be settled with respect to the metrics based on HTML markup (TEDS and GriTS). Different models used slightly different sets of HTML tags to represent the tables they understood. For instance, some models distinguish header and body (using tags such as `<thead>`, `<tbody>`), while others do not. To avoid being penalized by such inconsistencies, we normalize each output preserving only three tags: `<table>`, `<tr>`, `<td>`. From now on, we implicitly consider that each TSR metric applies over such normalized outputs (and normalized GT) only.

5.3 Table extraction

We aim at evaluating TE to compare directly end-to-end methods as a whole. To this end, we propose a metric similar in spirit with the well-known Precision and Recall metrics, because these are easily interpretable. Recall how, in Definition 1, we replaced *binary* values (1 for TP and 0 for FP) by *continuous* scores, parameterized by the Jaccard index score. We use a similar idea, as follows.

Recall that θ_J is the Jaccard threshold above which table detection is considered to have found a table. Then:

Definition 2 (TSR Precision and Recall)**.** Denoting for each detected (positive) table i its Jaccard index J_i, given a TSR metric which associates to each positive i, the score s_i^{TSR}, we define the TSR precision and recall as:

$$P^{\text{TSR}} = \frac{1}{|\mathcal{P}^+|} \sum_{i\in\mathcal{P}^+} s_i^{\text{TSR}} \cdot \mathbf{1}_{[J_i>\theta_J]} \qquad R^{\text{TSR}} = \frac{1}{|\mathcal{G}|} \sum_{i\in\mathcal{P}^+} s_i^{\text{TSR}} \cdot \mathbf{1}_{[J_i>\theta_J]}$$

where $\mathbf{1}_{[J_i>\theta_J]}$ is the indicator function whose value is 0 for tables whose detection score is below θ_J and 1 for the others.

Intuitively, in the above metrics, we propagate the strength (or confidence) of the table detection, in order to also reflect it in the "downstream" task of TSR.

6 EVALUATION METHODOLOGY

We now explain how we apply these metrics to ensure a fair comparison of all models (Section 6.1) and to evaluate end-to-end TE methods (Section 6.2).

6.1 Content Table Detection

For **methods that output table locations** (bounding boxes), we can use the metrics presented in Section 5.1, based on IoU. We denote the output of such a method by $\boxed{\mathcal{P}^{++}_{\square}}$, whose elements are tuples that include bounding boxes, HTML table representation and, if available, confidence scores. For models which **do not output bounding boxes** (recall Table 2), based on the *content*, we obtain $\boxed{\mathcal{P}^{++}_{\text{str}}}$ whose elements are tuples that include HTML table representation and, if available, confidence scores. Accordingly, we call "*bbox* TD" and "*content* TD" the TD evaluations based on bounding boxes, respectively, table content. After the TD evaluation, TSR evaluation inputs elements from either $\mathcal{P}^{++}_{\square}$ or $\mathcal{P}^{++}_{\text{str}}$. Figure 3 depicts an overview of possible paths for evaluation. Probabilistic models offer different ways to evaluate their performance.

As the **LVLM** (Section 4.1.3) outputs only HTML tables without bounding box coordinates, we need a way to determine whether a table has been detected, based only on the predicted and GT tables HTML markup. This requires correctly identifying "corresponding" tables across predicted and GT HTML tables, which is not straightforward. Recall that here we want to know if the table has been *found*, not if its structure is correct (the latter is the job of TSR). For this purpose, metrics such as ROUGE [37] or BLEU [47], which are Recall and Precision-based metrics, are sometimes applied, e.g., BLEU-4 was used as TSR metric [36] (see Section 5.1). However, these metrics focus on one aspect only (either Precision or Recall), and do not measure similarity.

To evaluate LVLM TD results, we characterize tables by their *content* (ignoring the table structure tags), by building from each table a *multisets of 2-grams* from content present cells. We ignore table structure tags here because they simply reflect HTML syntax thus their similarity is not informative; we want to measure how much cell *content* is shared by the prediction and GT. Given the multisets obtained from the prediction (S_P), respectively, from the GT (S_{GT}), we define: $\text{Jaccard}(S_P, S_{GT}) = \frac{|S_P \Cap S_{GT}|}{|S_P \Cup S_{GT}|}$ where $\Cup$ is the multiset union (retaining each multiset element with its maximum multiplicity), and $\Cap$ the multiset intersection (based on minimum multiplicity). We use *multisets* rather than regular sets to better characterize cell content, which may contain some repetitions. We use *character-level n*-grams instead of the more common word-level *n*-grams since the former better reflect character-level errors, which happen frequently with LVLM, due to hallucination or token extraction issues. Because of frequent alignment issues for token extraction (through PDFAlto), especially within mathematical formulas, we do not treat a whitespace as a character.

From now on, we call **content-Jaccard** the *Jaccard index over 2-gram-2-character multisets*. Thus, for all models, we can define TP and FP using *content-Jaccard*, which leads to $\mathcal{P}^{++}_{\text{str}}$.

6.2 Evaluating TSR Impacted by TD

Traditionally, TD and TSR are evaluated independently:

$$\mathcal{X}_{\text{TD}} \xmapsto{\text{TD Model}} \widehat{\mathcal{Y}}_{\text{TD}} \qquad \mathcal{X}_{\text{TSR}} \xmapsto{\text{TSR Model}} \widehat{\mathcal{Y}}_{\text{TSR}}$$

The TD test dataset consists of a set of page images $\mathcal{X}_{\text{TD}}$ and GT table location annotations $\mathcal{Y}_{\text{TD}}$. The TSR test dataset consists of a set of cropped table images $\mathcal{X}_{\text{TSR}}$ and GT table structures $\mathcal{Y}_{\text{TSR}}$.

However, in downstream tasks, models are used sequentially to perform end-to-end TE, thus TSR evaluation depends on TD:

$$\mathcal{X}_{\text{TD}} \overbrace{\xmapsto{\text{TD Model}} \mathcal{P}|_{\text{TD}} \xmapsto{\text{TSR Model}}}^{\text{TE Model}} \mathcal{P}$$

where $\mathcal{P}$ contains the HTML tables, and optionally, the bounding box coordinates and the confidence score, as shown in Table 2. Note that for one-step TE methods, we may not be able to separate (discern) the two sub-models.

Aiming for an end-to-end evaluation, we choose not to use a predefined dataset $\mathcal{X}_{\text{TSR}}$ as input for TSR. Instead, we use the set of true positives $\mathcal{P}^{++}|_{\text{TSR}} \subseteq \mathcal{P}^{+}$ obtained from the TD inference part; we use $\theta_J = 0.5$ to distinguish TP from FP. In this way, the TSR score better reflects what can be observed in the use of the end-to-end model. In other words, the evaluation pairs are: $(\mathcal{P}^{+}|_{\text{TD}}, \mathcal{Y}_{\text{TD}})$ and $(\mathcal{P}^{++}|_{\text{TSR}}, \mathcal{Y}_{\text{TSR}})$. This has two consequences: (*i*) *TSR evaluation is not independent of TD*; we denote those scores $\boxed{\text{TSR}|_{\text{TD}}}$; (*ii*) We can *no longer directly compare* $\text{TSR}|_{\text{TD}}$ *results between models*, since the input dataset $\mathcal{P}^{++}|_{\text{TSR}}$ is different for each model (determined by the model's performance on TD).

Consequently, if TD fails to correctly[15] detect tables, the structure model will be penalized. While this may seem "unfair" to TSR models, it reflects the reality of end-to-end TE result quality, which are measured on the final results. Moreover, when using one-step (black-box) models, we cannot separate the detection part from the structure part.

Given a TSR metric, for each TP i, we compute a score s_i^{TSR}. The final $\text{TSR}|_{\text{TD}}$ score for each model is then obtained by averaging the scores over the set of TP $\mathcal{P}^{++}$: $\text{TSR}|_{\text{TD}} = \frac{1}{|\mathcal{P}^{++}|} \sum_{i \in \mathcal{P}^{++}} s_i^{\text{TSR}}$.

7 EXPERIMENTAL EVALUATION

We describe the settings used for the models under test, then the result of our evaluation on each task. The datasets used are described in Section 3. Our code is available on GitLab; we implemented our pipelines using [56] as starting point. Our benchmark measures method performance using the previously described metrics in Section 5.

7.1 Models settings

Among the Python baselines (Section 4.1.1), we used PDFPlumber and PyMuPDF with the default configuration. We used Camelot with the `Lattice` method, which relies on demarcated lines between cells and outperformed other methods. However, this method is limited, as not all tables include visible lines.

We used Grobid's default configuration (Section 4.1.2), which is a feature-engineered ML approach with a cascading of linear chains of Conditional Random Fields (CRF). The command `processFullText` extracts and structures in TEI (an XML dialect) the full text of PDF files. From the TEI output, we retrieve table's locations (page numbers) and coordinates (in points) and convert them into pixels with respect to GT image sizes for comparison.

For LVLM, we used zero-shot learning, with a prompt, to perform table extraction. The image size parameter (through API) is set to

[15] For instance, a TP with IoU at 51% may miss part of the table's content.

`high`: LVLM looks at the input as a low-resolution 512×512 image, and then creates detailed crops for each tile of 512×512 pixels. The HTML output is then formatted for *content*-based evaluation.

For Docling, we use the default `convert` method to retrieve extracted tables (location and HTML). We calculate the location using the minimum bounding box of the table tokens' convex hull.

For object detection-based methods, we use TATR-e, XY+TATR-e and VGT+TATR-s, respectively now denoted as TATR, XY and VGT in figures. Their pipelines are respectively described in Sections 4.2.2, 4.3 and 4.4. All rely on TATR-s for TSR; we use PDFAlto to retrieve text content (token) from the initial PDF. Regarding the parameters, we set $p_{\text{px,s}} = 100$ pixels, which determines how much padding in pixels will be added around a detected table before outputting a cropped image of the table. For XY+TATR-e, cropped images, which are XY-cut outputs, are padded with $p_{\text{px,d}} = 10$ pixels before being fed to TATR-d. These hyperparameters have been adjusted based on TATR-s. Finally, we use the VGT fine-tuned on the PubLayNet benchmark, DiT-base weights for ViT, BERT tokenizer with LayoutLM [63] for word embeddings grid model to obtain the same pre-trained model from the VGT repository [9]. We used the pipeline of [9], including the PDF parser, to obtain a grid input (tokens' position and content on pages).

For *bbox* TD evaluation, we use the pixel coordinates of images.

7.2 Table detection performance

Figure 4 plots Precision–Recall curves from *bbox* TD and Figure 5 from *content* TD over our three datasets. Models are distinguished by colors (same as in Table 2). We plot probabilistic model scores with curves, and baseline scores with various-shape dots. Different line styles denote different metrics: plain curves (or round, colored dots •) show Precision and Recall with Jaccard index threshold at $\theta_J = 0.5$ (denoted as $\boxed{X_{0.5}}$); dotted curves (or triangular dots ▲) show expected Precision and expected Recall, corresponding to $\boxed{\mathbb{E}_{0.5}[X]}$. Moreover, we draw iso-$F_1$ curves in gray (F_1 values: 0.2, 0.4, 0.6, 0.8). We omit $\mathbb{E}_0[X]$ to avoid overloading the figures. *The closer a curve is to the top-right corner, the better its trade-off between Precision and Recall.* Figure 5 with *content* TD is used only to compare the LVLM with other models. All curves start at (0,1), which is the hypothetical case of $\theta_c = 1$ with a perfect Precision (because only the most trustworthy predictions are considered), and end at (1,0), which is the hypothetical case of $\theta_c = 0$ with a perfect Recall (because all predictions are considered positive). Curves are decreasing and their tail (constant Precision over a Recall range until 1) is due to non-vanishing minimum θ_c, thus models reach their higher Recall with $\widetilde{\theta_c} > 0$. Finally, we compute Average Precision scores with $AP_{0.5}$, denoted as $\boxed{AP}$, which is the area under the Precision–Recall curve (see Table 3) to easily compare curves. Table 4 gathers F_1 scores for other models.

As expected, results obtained through *bbox* TD (Figure 4) are better than those with *content* TD (Figure 5): detecting a table by its bounding box is easier than by its content. Performance varies when changing the datasets. For example, Grobid, trained on scientific documents, has low F_1-score on Table-BRGM but significantly higher on PubTables. XY achieves better performance on Table-BRGM but does not maintain its lead over the other two datasets. On our PubTables dataset, TATR obtains an almost perfect score, on the raw Precision–Recall curve (at $\theta_J = 0.5$). This is because it has been pre-trained on PubTables, and our test set has similar inputs (in terms of layout or table style).

PyMuPDF and Camelot rank among the best on Table-BRGM, competing with computer vision-based approaches, but they are worse than Grobid on Table-arXiv, due to the different table types. PDFPlumber achieves results similar to Camelot but with lower precision.

Based on Figure 5, we can rank baselines on each dataset. We write $\boxed{x \preccurlyeq y}$ to denote that Model x has worse scores than Model y, i.e., its performance metrics (precision, recall, F_1-score, expected or not) are lower. For Table-BRGM: Grobid $\preccurlyeq$ LVLM $\preccurlyeq$ PDFPlumber $\preccurlyeq$ Camelot $\preccurlyeq$ PyMuPDF $\preccurlyeq$ Docling, while for PubTables and Table-arXiv: PDFPlumber $\preccurlyeq$ Camelot $\preccurlyeq$ LVLM $\preccurlyeq$ PyMuPDF $\preccurlyeq$ Grobid $\preccurlyeq$ Docling. As noted in [46], LVLM struggles with small text and tasks requiring precise spatial positioning, leading to lower performance. Models not based on object detection (PDFPlumber, PyMuPDF, Camelot) are strongly outperformed by the best (Precision, Recall) pairs obtained for TATR, XY and VGT. Docling, with the advanced RL-DETR architecture (recall Section 4.1.4), significantly outperforms the other methods, particularly on heterogeneous scientific documents (PubTables and Table-arXiv).

The expected metric used, $\mathbb{E}_{0.5}[X]$, follows the $X_{0.5}$ trend, but is stricter (lower scores) and also adds information about the Jaccard index. On one hand, on Table-BRGM and Table-arXiv, considering dotted curves, VGT outperforms TATR-detect-based models. Since these expected scores include a varying threshold, it means that VGT predicts bounding boxes that are closer (pixel-wise) to the GT annotations used. This underlines the fact that VGT delimits a table right up to its borders, as we did during annotation, contrary to TATR-detect that stops at the written elements. Thus, the IoU score is higher which induce higher expected Precision and Recall. On the other hand, on PubTables, TATR-detect-based models (i.e. TATR and XY) output bounding boxes which are similar to the ones in GT (automatically generated), and thus achieve better scores.

Figure 6 illustrates miscalibration for probabilistic models on PubTables and Table-arXiv (Table-BRGM is similar to Table-arXiv). The yellow curves show the *distribution of confidence scores* for each model. Reliability diagrams [45] are histograms showing *precision as a function of confidence* by intervals of width 0.1. As positive, we consider the set of almost all predictions $\mathcal{P}^+ = \mathcal{P}_{0.05}$, and then compute $P_{0.5}$. For perfect calibration, the histogram should match the identity function (the hatched histogram). Thus, the gap between the colored and identity histograms denotes a model's miscalibration (the lower, the better); D-ECE (Section 5.1.3) measures this.

On *PubTables*, TATR and VGT are well calibrated (low D-ECE). Note that TATR outputs almost binary confidence score (according to the confidence score distribution). There is no uncertainty over this dataset, which is similar to its training set. XY-cut preprocessing impacted the model performance: in the reliability diagrams, only highest confidence scores predictions should be considered as positive. On *Table-arXiv*, more heterogeneous, TATR-detect-based models are poorly calibrated compared to VGT, and, as for XY-cut, confidence scores become meaningless. For example, among predictions with confidence scores between 80% and 90%, on average, less than 2% are actual TP.

From these experiments, we draw the following lessons.

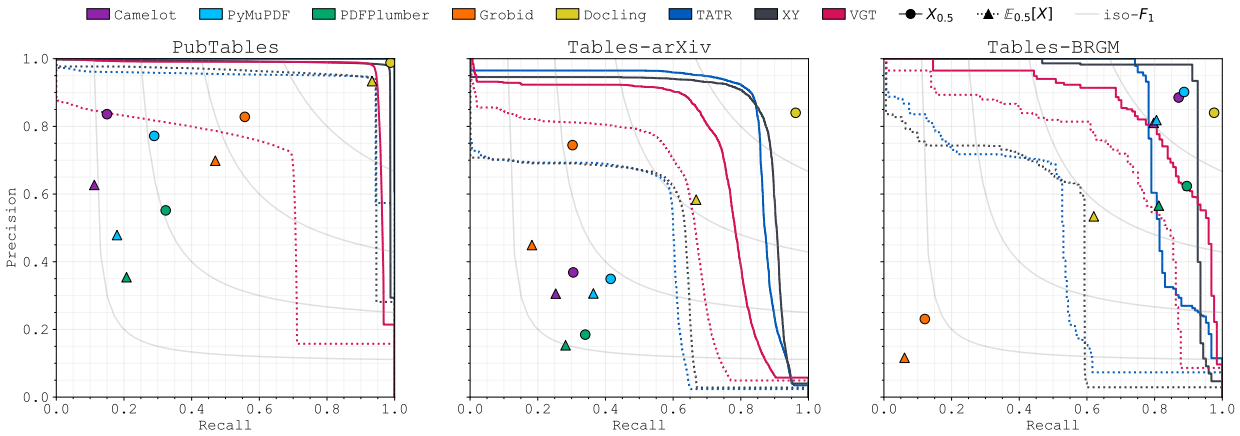

Figure 4: Precision–Recall curves, depending on IoU (50 % or expected metric) for (left to right): PubTables, Table-arXiv and Table-BRGM dataset (*bbox* TD).

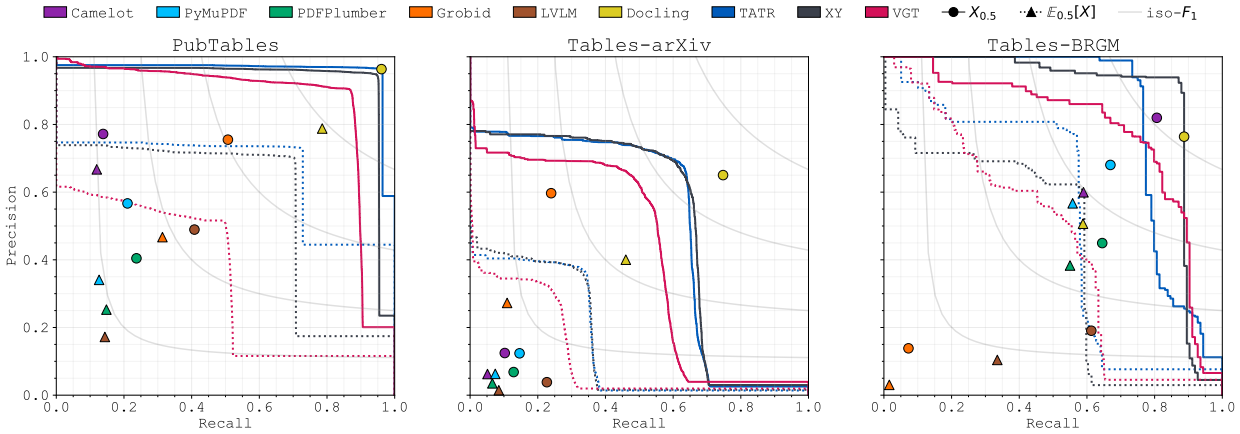

Figure 5: Precision–Recall curves, depending on content-Jaccard (50% or expected metric) for (left to right): PubTables, Table-arXiv and Table-BRGM dataset (*content* TD).

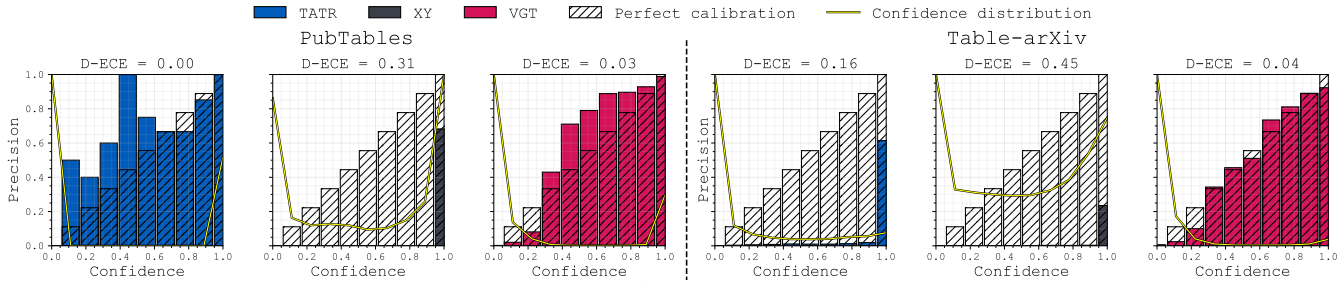

Figure 6: Reliability diagrams with confidence distribution (in yellow) for TATR, XY and VGT on PubTables and Table-arXiv.

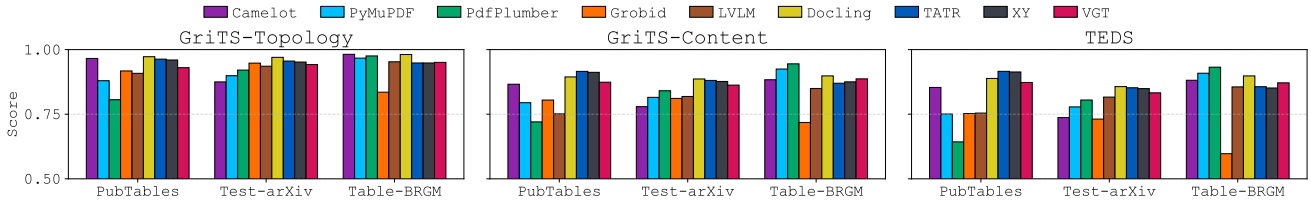

Figure 7: $\text{TSR}|_{\text{TD}}$ for GriTS Topology, GriTS Content and TEDS (*content* evaluation). Scores are between 0.5 and 1.

Recall that there are *two paradigms for TE models usage* in downstream tasks. First, one can consider only positive predictions $\mathcal{P}^+$, which requires setting a confidence threshold. Second, we can consider the whole set of detected tables $\mathcal{P}$, with their attached confidence scores. In the first case, TATR and XY models fit perfectly, and the θ_c choice is easy according to Figure 6. Docling also fits this case, without choosing any threshold. In the second case, VGT ensures interpretable and reliable confidence scores.

The threshold choice θ_c impacts VGT and TATR differently. TATR with XY achieves higher Precision for high θ_c. XY+TATR-detect's Precision is lower than TATR-detect's (for low thresholds) because there are more predictions with XY-cut; without the 2-top post-processing (recall Section 4.3), there would be even more.

7.3 Table structure recognition performance

Figure 7 plots $\text{TSR}|_{\text{TD}}$ scores (recall Section 6.2) from *content* TD; all scores are between 0.5 and 1. For each model, we use the color assigned to it in Table 2. Each bar height averages the scores of all TP predictions, with $\theta_J = 0.5$. Thus, scores cannot be directly compared because the input dataset is different for each model (not the same size and not the same images). For example, TATR, XY, and VGT share the same TSR model (TATR-structure), and their average TSR scores are similar but not equal, because the TD model impacts the subsequent TSR model's performance. GriTS Topology scores tend to be higher in general than for Content and TEDS. Although datasets contain various cells shapes, most are "simple". Thus, if a model correctly identifies the number of rows and columns, it achieves a perfect structural score. However, for higher Content and TEDS scores, token extraction must align with GT annotations. But every tokenization method introduces biases, making it harder to closely match the GT. For instance, PDFAlto does not retrieve all text from PDFs, especially stylized text like equations, which implies partial TCR for TATR-structure-based models.

7.4 Table extraction performance

Figures 8 and 9 illustrate Precision–Recall weighted by TSR scores curves (see Section 5.3) from *bbox* TD. Dashed curves (- -) represent scores for probabilistic models (with varying thresholds), while baseline models are represented with inverted triangle dots (▼). We compute $P^{\text{TSR}} - R^{\text{TSR}}$ with $\theta_J = 0.5$, denoted as $\boxed{X^{\text{TSR}}_{0.5}}$. Difference areas with raw P–R curves are shown shaded in the model's colors, and with transparent dots for baseline models.

The scores decrease compared to vanilla Precision and Recall (Figures 4 and 5), regardless of the TSR score used. GriTS Topology scores are closer to vanilla P–R, which means that when a table is detected (with Jaccard index $\theta_J = 0.5$), most of the time, the table topology (rows, columns, spanning cells) is correctly found. However, Precision and Recall with TEDS are lower, because they take into account partial token match. We obtain similar results using GriTS Content.

In Table 3 the "TE metrics" columns summarize AP metrics for different TSR scores: *Top*ology GriTS (denoted $\mathbf{AP}^{\text{Top}}$) and *Con*tent GriTS (denoted $\mathbf{AP}^{\text{Con}}$, recall Section 5.2.2), as well as TEDS (denoted $\mathbf{AP}^{\text{TEDS}}$, introduced in Section 5.2.1). Table 4 does the same for F_1-scores instead of AP. Though scores decreased for metrics with TSR, the ranking remains the same across different methods. AP^{TSR} and F_1^{TSR} fall for all models, revising the initial positive assessment of task TD (e.g. TATR and Docling on PubTables).

Further, we measured the CPU time used by inference on 1 000 samples, using 4 CPUs per task and 2 concurrent tasks. Camelot, PyMuPDF, PDFPlumber and Grobid are 10 times faster (~ 1 s/image) than LVLM (API calls), Docling, TATR, XY and VGT. Docling is significantly slower than Grobid and other Python libraries due to its use of computer vision models.

7.5 Experiment conclusions

The performance of the methods vary significantly depending on dataset style, a dependency captured by our diverse benchmark datasets. This variability arises from intrinsic design choices: either raw hard-coded features (for heuristic-based methods) or pre-training data (for probabilistic models). Docling achieves the best overall performance across datasets for TD, followed by probabilistic models. Meanwhile, other methods occasionally yield good results on specific document layouts. TATR (TD-specialized model) produces meaningless confidence scores, unlike VGT (DLA-specialized model), which offers interpretable results.

For TSR, the global table topology is generally well-understood in the detected tables, but TCR remains challenging in terms of content accuracy.

We considered a specific input type: PDF reports. Nevertheless, Grobid was the only model that leveraged the PDF format as input; other models handle images, as they rely on image object detection. Overall, while Docling (the most promising TD model) demonstrates strong performance, it has yet to achieve perfect TE, in terms of F_1^{TSR}.

8 CONCLUSION

We compared different methods for end-to-end table extraction from scientific PDF documents. Our main conclusion is that table extraction is not a solved problem (neither table detection nor table structure recognition). Various DL approaches have improved state-of-the-art results, however, there is no perfect solution. We manage to achieve good performance on both subtasks with: TATR-extract by modifying its pipeline; VGT+TATR-structure, to create a new end-to-end and better-calibrated model that provides more confidence in prediction; XY+TATR-extract, by adding additional pre-processing to use sub-images as input. In addition, Docling also performs well overall, thanks to its versatile pre-training, competing with probabilistic approaches.

Dataset characteristics have significant impact on model performance. Object detection-based models achieve the best overall results on TE. On downstream tasks, the choice of θ_c matters: either we set a high threshold and expect to get positive predictions, or we set low threshold and confidence scores and rely on confidence scores as probabilities. As models based on TATR-detect are not well-calibrated, possible future work includes using recalibration techniques to obtain more interpretable scores.

Our new framework for TE allows better end-to-end evaluation, encompassing TD and TSR, over a wide variety of model types on new datasets (Table-arXiv and Table-BRGM) that provide novel resources for advancing TE research and development.

Acknowledgments This work was supported by Inria and BRGM via the GéolAug project.

Table 3: Average Precision scores for TD and TE across datasets for probabilistic models.

Method	PubTables				Table-arXiv				Table-BRGM			
	TD	TE metrics			TD	TE metrics			TD	TE metrics		
	AP	AP^{Top}	AP^{Con}	AP^{TEDS}	AP	AP^{Top}	AP^{Con}	AP^{TEDS}	AP	AP^{Top}	AP^{Con}	AP^{TEDS}
TATR	**1.00**	**0.91**	**0.80**	**0.80**	0.84	**0.73**	**0.56**	**0.52**	0.84	0.77	0.69	0.66
XY	0.97	0.88	0.78	0.78	**0.85**	**0.73**	**0.56**	0.51	**0.92**	**0.83**	**0.71**	**0.67**
VGT	0.96	0.81	0.69	0.70	0.73	0.60	0.44	0.40	0.86	0.76	0.67	0.65

Table 4: F_1-scores for TD and TE across datasets for baseline models.

Method	PubTables				Table-arXiv				Table-BRGM			
	TD	TE metrics			TD	TE metrics			TD	TE metrics		
	F_1	F_1^{Top}	F_1^{Con}	F_1^{TEDS}	F_1	F_1^{Top}	F_1^{Con}	F_1^{TEDS}	F_1	F_1^{Top}	F_1^{Con}	F_1^{TEDS}
Camelot	0.25	0.23	0.20	0.20	0.33	0.20	0.15	0.13	0.88	0.82	0.73	0.73
PyMuPDF	0.42	0.33	0.29	0.27	0.38	0.25	0.19	0.17	0.89	0.76	0.69	0.68
PDFPlumber	0.41	0.29	0.25	0.22	0.24	0.17	0.13	0.12	0.73	0.63	0.56	0.56
Grobid	0.67	0.59	0.51	0.47	0.43	0.39	0.31	0.28	0.16	0.11	0.09	0.07
Docling	**0.99**	**0.95**	**0.86**	**0.86**	**0.89**	**0.83**	**0.72**	**0.67**	**0.90**	**0.88**	**0.79**	**0.79**

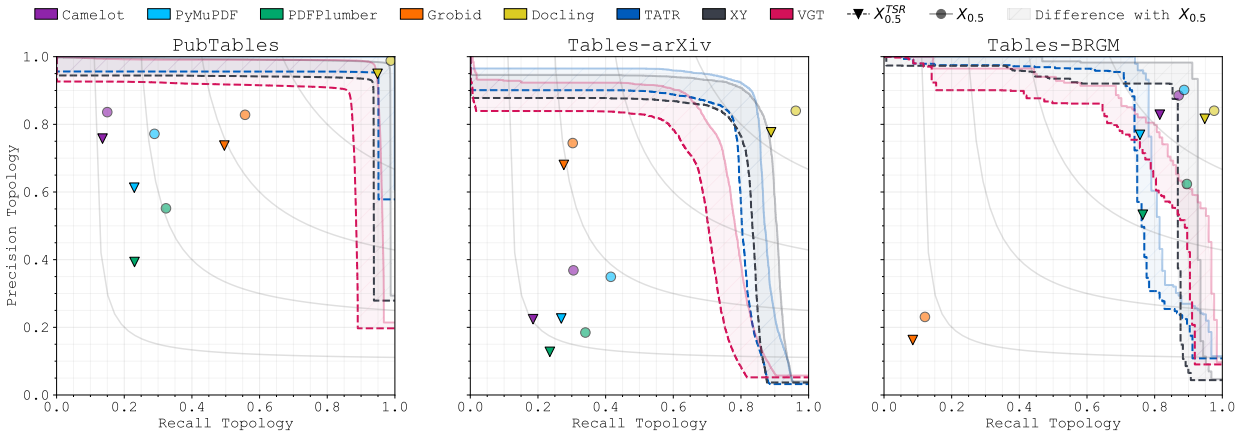

Figure 8: $P^{Top} - R^{Top}$ curves for (left to right): PubTables, Table-arXiv dataset and Table-BRGM (*bbox* TD).

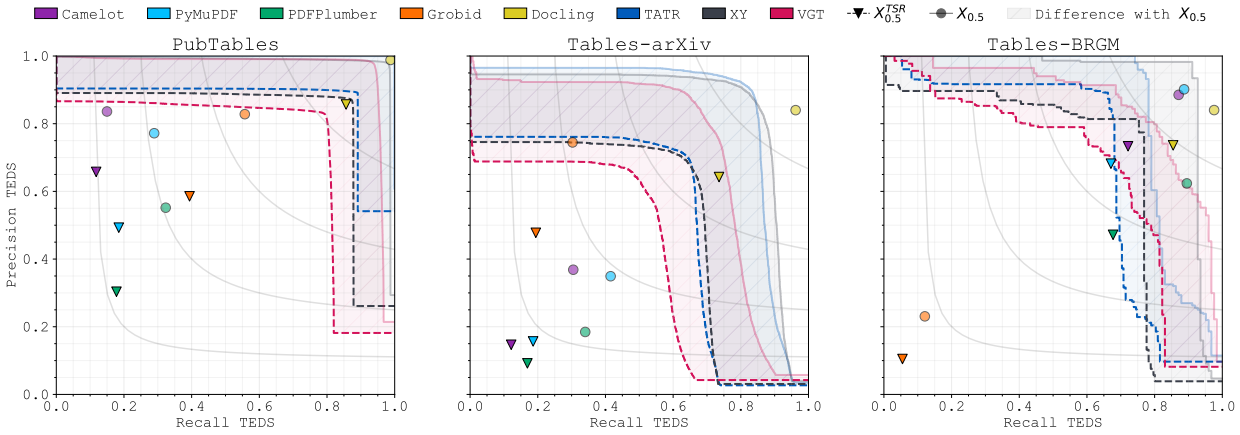

Figure 9: $P^{TEDS} - R^{TEDS}$ curves for (left to right): PubTables, Table-arXiv dataset and Table-BRGM (*bbox* TD).

REFERENCES

[1] Amihood Amir, Tzvika Hartman, Oren Kapah, Braha Shalom, and Dekel Tsur. 2007. Generalized LCS. *Theoretical Computer Science* 409, 50–61. https://doi.org/10.1007/978-3-540-75530-2_5

[2] Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Nikolaos Livathinos, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Fabian Lindlbauer, Kasper Dinkla, Lokesh Mishra, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. 2024. Docling Technical Report. arXiv:2408.09869 [cs.CL] https://arxiv.org/abs/2408.09869

[3] Muhammad Imam Luthfi Balaka, David Alexander, Qiming Wang, Yue Gong, Adila Krisnadhi, and Raul Castro Fernandez. 2025. Pneuma: Leveraging LLMs for Tabular Data Representation and Retrieval. *Proc. ACM Manag. Data* 3, 3 (2025), 200:1–200:28.

[4] Galal M. Binmakhashen and Sabri A. Mahmoud. 2019. Document Layout Analysis: A Comprehensive Survey. *ACM Comput. Surv.* 52, 6, Article 109 (Oct. 2019), 36 pages. https://doi.org/10.1145/3355610

[5] Zhaowei Cai and Nuno Vasconcelos. 2017. Cascade R-CNN: Delving into High Quality Object Detection. arXiv:1712.00726 [cs.CV] https://arxiv.org/abs/1712.00726

[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-End Object Detection with Transformers. arXiv:2005.12872 [cs.CV] https://arxiv.org/abs/2005.12872

[7] Surekha Chandran and Rangachar Kasturi. 1993. Structural recognition of tabulated data. In *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93)*. IEEE, 516–519.

[8] Zewen Chi, Heyan Huang, Heng-Da Xu, Houjin Yu, Wanxuan Yin, and Xian-Ling Mao. 2019. Complicated Table Structure Recognition. *arXiv preprint arXiv:1908.04729* (2019).

[9] Cheng Da, Chuwei Luo, Qi Zheng, and Cong Yao. 2023. Vision Grid Transformer for Document Layout Analysis. arXiv:2308.14978 [cs.CV] https://arxiv.org/abs/2308.14978

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL] https://arxiv.org/abs/1810.04805

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:2010.11929 [cs.CV] https://arxiv.org/abs/2010.11929

[12] Tapio Elomaa. 2013. Anssi Nurminen Algorithmic Extraction of Data in Tables in PDF Documents. https://api.semanticscholar.org/CorpusID:33157514

[13] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision* 88 (2010), 303–338.

[14] Miao Fan and Doo Soon Kim. 2015. Detecting Table Region in PDF Documents Using Distant Supervision. arXiv:1506.08891 [cs.CV]

[15] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. 2010. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 9 (2010), 1627–1645. https://doi.org/10.1109/TPAMI.2009.167

[16] Liangcai Gao, Yilun Huang, Hervé Déjean, Jean-Luc Meunier, Qinqin Yan, Yu Fang, Florian Kleber, and Eva Lang. 2019. ICDAR 2019 Competition on Table Detection and Recognition (cTDaR). In *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 1510–1515. https://doi.org/10.1109/ICDAR.2019.00243

[17] Basilios Gatos, Dimitrios Danatsas, Ioannis Pratikakis, and Stavros J. Perantonis. 2005. Automatic Table Detection in Document Images. In *Pattern Recognition and Data Mining*, Sameer Singh, Maneesha Singh, Chid Apte, and Petra Perner (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 609–618.

[18] Max Göbel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. 2012. A methodology for evaluating algorithms for table understanding in PDF documents. In *Proceedings of the 2012 ACM Symposium on Document Engineering* (Paris, France) *(DocEng '12)*. Association for Computing Machinery, New York, NY, USA, 45–48. https://doi.org/10.1145/2361354.2361365

[19] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. arXiv:1706.04599 [cs.LG] https://arxiv.org/abs/1706.04599

[20] Yuxiang Guo, Zhonghao Hu, Yuren Mao, Baihua Zheng, Yunjun Gao, and Mingwei Zhou. 2025. Birdie: Natural Language-Driven Table Discovery Using Differentiable Search Index. *ArXiv preprint* abs/2504.21282 (2025). https://arxiv.org/abs/2504.21282

[21] Jaekyu Ha, R.M. Haralick, and I.T. Phillips. 1995. Recursive X-Y cut using bounding boxes of connected components. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Vol. 2. 952–955 vol.2. https://doi.org/10.1109/ICDAR.1995.602059

[22] Leipeng Hao, Liangcai Gao, Xiaohan Yi, and Zhi Tang. 2016. A table detection method for pdf documents based on convolutional neural networks. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*. IEEE, 287–292.

[23] Gaurav Harit and Anukriti Bansal. 2012. Table detection in document images using header and trailer patterns. In *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing* (Mumbai, India) *(ICVGIP '12)*. Association for Computing Machinery, New York, NY, USA, Article 62, 8 pages. https://doi.org/10.1145/2425333.2425395

[24] PMC Open Access Subset [Internet]. 2003. . Bethesda (MD): National Library of Medicine. Retrieved June 23, 2025 from https://pmc.ncbi.nlm.nih.gov/tools/openftlist/.

[25] Katsuhiko Itonori. 1993. Table structure recognition based on textblock arrangement and ruled line position. In *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93)*. IEEE, 765–768.

[26] MAC Akmal Jahan and Roshan G Ragel. 2014. Locating tables in scanned documents for reconstructing and republishing. In *7th International Conference on Information and Automation for Sustainability*. IEEE, 1–6.

[27] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazàn Almazàn, and Lluís Pere de las Heras. 2013. ICDAR 2013 Robust Reading Competition. In *2013 12th International Conference on Document Analysis and Recognition*. 1484–1493. https://doi.org/10.1109/ICDAR.2013.221

[28] Thotreingam Kasar, Philippine Barlas, Sebastien Adam, Clément Chatelain, and Thierry Paquet. 2013. Learning to detect tables in scanned document images using line information. In *2013 12th international conference on document analysis and recognition*. IEEE, 1185–1189.

[29] Aamod Khatiwada, Roee Shraga, and Renée J. Miller. 2026. Diverse Unionable Tuple Search: Novelty-Driven Discovery in Data Lakes. In *Proceedings 29th International Conference on Extending Database Technology, EDBT 2026, Tampere, Finland, March 24-27, 2026*, Wolfgang Lehner, Vanessa Braganholo, Kostas Stefanidis, Zheying Zhang, Alexander Krause, and João Felipe Nicolaci Pimentel (Eds.). OpenProceedings.org, 42–55. https://doi.org/10.48786/EDBT.2026.04

[30] Thomas Kieninger and Andreas Dengel. 1998. The T-Recs Table Recognition and Analysis System, Vol. 1655. 255–269. https://doi.org/10.1007/3-540-48172-9_21

[31] Christos Koutras, Jiani Zhang, Xiao Qin, Chuan Lei, Vassilis N. Ioannidis, Christos Faloutsos, George Karypis, and Asterios Katsifodimos. 2025. OmniMatch: Joinability Discovery in Data Products. *Proc. VLDB Endow.* 18, 11 (2025), 4588–4601. https://www.vldb.org/pvldb/vol18/p4588-koutras.pdf

[32] Fabian Kuppers, Jan Kronenberger, Amirhossein Shantia, and Anselm Haselhoff. 2020. Multivariate Confidence Calibration for Object Detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE. https://doi.org/10.1109/cvprw50498.2020.00171

[33] Vladimir I. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady* 10 (1965), 707–710. https://api.semanticscholar.org/CorpusID:60827152

[34] Junlong Li, Yiheng Xu, Tengchao Lv, Lei Cui, Cha Zhang, and Furu Wei. 2022. DiT: Self-supervised Pre-training for Document Image Transformer. arXiv:2203.02378 [cs.CV] https://arxiv.org/abs/2203.02378

[35] Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. 2019. TableBank: A Benchmark Dataset for Table Detection and Recognition. arXiv:1903.01949 [cs.CV]

[36] Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. 2020. TableBank: Table Benchmark for Image-based Table Detection and Recognition. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis (Eds.). European Language Resources Association, Marseille, France, 1918–1925. https://aclanthology.org/2020.lrec-1.236/

[37] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, 74–81. https://aclanthology.org/W04-1013/

[38] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. *CoRR* abs/1405.0312 (2014). arXiv:1405.0312 http://arxiv.org/abs/1405.0312

[39] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. Focal Loss for Dense Object Detection. arXiv:1708.02002 [cs.CV] https://arxiv.org/abs/1708.02002

[40] Patrice Lopez. 2009. GROBID: Combining Automatic Bibliographic Data Recognition and Term Extraction for Scholarship Publications. In *Research and Advanced Technology for Digital Libraries*, Maristella Agosti, José Borbinha, Sarantos Kapidakis, Christos Papatheodorou, and Giannis Tsakonas (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 473–474.

[41] Maksym Lysak, Ahmed Nassar, Nikolaos Livathinos, Christoph Auer, and Peter Staar. 2023. Optimized Table Tokenization for Table Structure Recognition. In *Document Analysis and Recognition - ICDAR 2023*, Gernot A. Fink, Rajiv Jain, Koichi Kise, and Richard Zanibbi (Eds.). Springer Nature Switzerland, Cham,

37–50.
[42] Chixiang Ma, Weihong Lin, Lei Sun, and Qiang Huo. 2023. Robust Table Detection and Structure Recognition from Heterogeneous Document Images. *Pattern Recognition* 133 (Jan. 2023), 109006. https://doi.org/10.1016/j.patcog.2022.109006
[43] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (Austin, Texas) *(AAAI'15)*. AAAI Press, 2901–2907.
[44] Ahmed Nassar, Nikolaos Livathinos, Maksym Lysak, and Peter Staar. 2022. TableFormer: Table Structure Understanding With Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4614–4623. https://doi.org/10.1109/CVPR52688.2022.00457
[45] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning* (Bonn, Germany) *(ICML '05)*. Association for Computing Machinery, New York, NY, USA, 625–632. https://doi.org/10.1145/1102351.1102430
[46] OpenAI. 2024. *GPT-4o mini: advancing cost-efficient intelligence*. Retrieved May 26, 2025 from https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/?utm_source=chatgpt.com
[47] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Annual Meeting of the Association for Computational Linguistics*. https://api.semanticscholar.org/CorpusID:11080756
[48] Norman W. Paton, Jiaoyan Chen, and Zhenyu Wu. 2024. Dataset Discovery and Exploration: A Survey. *ACM Comput. Surv.* 56, 4 (2024), 102:1–102:37. https://doi.org/10.1145/3626521
[49] Mateusz Pawlik and Nikolaus Augsten. 2016. Tree edit distance: Robust and memory-efficient. *Information Systems* 56 (2016), 157–173. https://doi.org/10.1016/j.is.2015.08.004
[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn : Average precision score. *Journal of Machine Learning Research* 12 (2011), 2825–2830. https://scikit-learn.org/1.5/modules/generated/sklearn.metrics.average_precision_score.html
[51] Birgit Pfitzmann, Christoph Auer, Michele Dolfi, Ahmed S. Nassar, and Peter Staar. 2022. DocLayNet: A Large Human-Annotated Dataset for Document-Layout Segmentation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 3743–3751. https://doi.org/10.1145/3534678.3539043
[52] P. Pyreddy and W. B. Croft. 1997. *TINTI: A System for Retrieval in Text Tables TITLE2:*. Technical Report. USA.
[53] Mahmoud Salaheldin Kasem, Abdelrahman Abdallah, Alexander Berendeyev, Ebrahem Elkady, Mohamed Mahmoud, Mahmoud Abdalla, Mohamed Hamada, Sebastiano Vascon, Daniyar Nurseitov, and Islam Taj-Eddin. 2024. Deep Learning for Table Detection and Structure Recognition: A Survey. *ACM Comput. Surv.* 56, 12, Article 305 (Oct. 2024), 41 pages. https://doi.org/10.1145/3657281
[54] Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. 2017. DeepDeSRT: Deep Learning for Detection and Structure Recognition of Tables in Document Images. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Vol. 01. 1162–1167. https://doi.org/10.1109/ICDAR.2017.192
[55] Jeremy Singer-Vine and The pdfplumber contributors. 2025. *pdfplumber*. https://github.com/jsvine/pdfplumber
[56] Brandon Smock and Rohith Pesala. 2021. *Table Transformer*. https://github.com/microsoft/table-transformer
[57] Brandon Smock, Rohith Pesala, and Robin Abraham. 2022. PubTables-1M: Towards Comprehensive Table Extraction From Unstructured Documents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4634–4642.
[58] Brandon Smock, Rohith Pesala, and Robin Abraham. 2023. Aligning Benchmark Datasets for Table Structure Recognition. In *Document Analysis and Recognition - ICDAR 2023*, Gernot A. Fink, Rajiv Jain, Koichi Kise, and Richard Zanibbi (Eds.). Springer Nature Switzerland, Cham, 371–386.
[59] Brandon Smock, Rohith Pesala, and Robin Abraham. 2023. GriTS: Grid table similarity metric for table structure recognition. arXiv:2203.12555 [cs.LG] https://arxiv.org/abs/2203.12555
[60] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. 2019. FCOS: Fully Convolutional One-Stage Object Detection. arXiv:1904.01355 [cs.CV] https://arxiv.org/abs/1904.01355
[61] Yalin Wangt, Ihsin T Phillipst, and Robert Haralick. 2001. Automatic table ground truth generation and a background-analysis-based table structure extraction method. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*. IEEE, 528–532.
[62] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. https://github.com/facebookresearch/detectron2.
[63] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2019. LayoutLM: Pre-training of Text and Layout for Document Image Understanding. arXiv:1912.13318 [cs.CL]
[64] Fan Yang, Lei Hu, Xinwu Liu, Shuangping Huang, and Zhenghui Gu. 2023. A large-scale dataset for end-to-end table recognition in the wild. *Scientific Data* 10, 1 (2023), 110.
[65] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. 2022. DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection. arXiv:2203.03605 [cs.CV] https://arxiv.org/abs/2203.03605
[66] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. 2024. DETRs Beat YOLOs on Real-time Object Detection. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 16965–16974. https://doi.org/10.1109/CVPR52733.2024.01605
[67] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. 2020. Image-based table recognition: data, model, and evaluation. arXiv:1911.10683 [cs.CV] https://arxiv.org/abs/1911.10683
[68] Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. 2019. PubLayNet: largest dataset ever for document layout analysis. arXiv:1908.07836 [cs.CL] https://arxiv.org/abs/1908.07836
[69] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. 2019. Objects as Points. arXiv:1904.07850 [cs.CV] https://arxiv.org/abs/1904.07850
[70] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2021. Deformable DETR: Deformable Transformers for End-to-End Object Detection. arXiv:2010.04159 [cs.CV] https://arxiv.org/abs/2010.04159