Structured Prediction for Scalable Spreadsheet Table Understanding: From Cell Types to Table Ranges

Antoine Gauquier
antoine.gauquier@ens.psl.eu
DI ENS, ENS, CNRS, PSL, Inria
Paris, France

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

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

Abstract

Spreadsheets are a primary medium for publishing tabular data, yet automatically extracting structured content from them remains difficult due to heterogeneous layouts, diverse file formats, and inconsistent organizational conventions. We address two core tasks in spreadsheet understanding: *Cell-Type Classification* (CTC), which assigns roles to cells, and *Table Detection* (TD), which identifies table bounding boxes within sheets. We propose an efficient two-stage pipeline in which a learned CTC model feeds a deterministic TD algorithm. For CTC, we use a LightGBM classifier over 65 structured features together with a pairwise CRF enforcing spatial consistency across the cell grid. Our TD method extracts table ranges from predicted cell types by a deterministic five-stage procedure. For evaluation, we built and share StatSheets, a multilingual benchmark of 737 manually annotated sheets from 14 public data providers across multiple countries and file formats. Under 5-fold cross-validation, our CRF-LightGBM system achieves a Mean File-Macro F_1 score of 0.937 on CTC, within 0.6 percentage points of the GPU-based TUTA Transformer, while requiring substantially fewer computational resources. For TD, our deterministic approach outperforms region-based baselines and remains competitive with recent LLM-based systems such as SpreadsheetLLM. These results demonstrate that combining non-linear structured prediction with deterministic range extraction provides a competitive, scalable, and computationally efficient approach to spreadsheet table understanding.

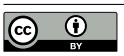
CCS Concepts

• **Applied computing** → **Document analysis**; • **Information systems** → *Data extraction and integration.*

Keywords

Spreadsheets, Table Understanding, Structured Prediction, Cell-Type Classification, Table Detection

ACM Reference Format:
Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. 2026. Structured Prediction for Scalable Spreadsheet Table Understanding: From Cell Types to Table Ranges. In *Proceedings of the 35th ACM International Conference on Information and Knowledge Management (CIKM '26), November 07–11, 2026, Rome, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3799682.3840907



CIKM '26, Rome, Italy

ACM ISBN 979-8-4007-2539-5/2026/11
https://doi.org/10.1145/3799682.3840907

1 Introduction

Spreadsheets are among the most widely used data management artifacts. Governments, international organizations, companies, and journalists routinely publish data in formats such as XLSX, XLS, ODS, CSV, and TSV. Public repositories such as Eurostat alone expose thousands of spreadsheets, many continuously updated and reused in downstream pipelines [11]. Unlike relational databases, however, *spreadsheets are designed for human interpretation rather than machine consumption*: their semantic structure is only partially explicit and instead conveyed through layout, formatting, merged cells, spatial proximity, etc. The conventions for doing so vary across publishers, domains, languages, and software ecosystems.

This flexibility, while central to spreadsheets' success, makes automated extraction and integration difficult. Real-world spreadsheets rarely contain clean rectangular tables. Instead, they mix data regions with hierarchical headers, titles, footnotes, metadata, and other auxiliary elements within the same sheet. Similar layouts may encode different semantics, while equivalent structures may appear in very different forms. Structural irregularities such as sparsity, merged cells, and discontinuous regions further complicate the problem. Consequently, recovering machine-interpretable tables from spreadsheets is substantially harder than extracting structured data from HTML tables or database exports.

Robust, large-scale spreadsheet understanding is increasingly critical for modern data management systems, including data lake construction, open-data indexing, fact-checking, business intelligence, and retrieval-augmented analytics. Errors in the detection of table boundaries or cell roles propagate to downstream tasks, producing corrupted schemas, invalid joins, or unusable datasets: spreadsheet understanding is not merely a preprocessing step, but *a foundational problem for scalable structured-data acquisition.* Two core tasks can be identified: *Cell-Type Classification* (CTC) assigns semantic roles (e.g., HEADER, DATA, TITLE) to cells, while *Table Detection* (TD) recovers table boundaries within sheets. Although CTC and TD have received continuous attention during the last decade, see [3, 7, 15, 16, 23, 30, 34], and [6, 8, 9, 21, 24, 32], respectively, existing approaches still face important limitations. **1.** Recent progress relies on large neural architectures *trained on proprietary corpora* and which are *expensive.* Transformer-based systems such as TUTA [34] and LLM-based approaches such as SpreadsheetLLM [9] achieve strong results through large-scale pre-training, but at substantial cost (Sec. 5.1 and 5.2). Such GPU-intensive systems can be problematic for pipelines processing thousands to millions of spreadsheets, where throughput and operational cost remain primary constraints. **2.** Prior work *only partially captures the heterogeneous signals* governing spreadsheet structure. Some methods emphasize semantic embeddings while under-exploiting formatting and

geometric regularities; others focus on visual or region-based segmentation while insufficiently modeling long-range dependencies. Yet, spreadsheet semantics emerges precisely from the interaction of multiple weak signals, including lexical content, formatting consistency, row- and column-level regularities, neighborhood coherence, and global spatial organization. **3.** *Reproducibility and evaluation remain problematic.* Several influential systems do not release implementations [7, 8, 21, 24, 30], or rely on proprietary datasets [7–9], preventing direct comparison and independent validation. Thus, it remains difficult to isolate the respective contributions of semantic modeling, structural priors, feature engineering, and computational scale. These issues are compounded by the lack of public datasets jointly supporting CTC and TD. Existing benchmarks are either single-task, structurally limited, or derived from outdated corpora. DECO [22], the only public benchmark covering both tasks, is based on the Enron archive [17] and reflects practices from the early 2000s; it excludes large spreadsheets, lacks multilingual coverage, and omits visually hidden content, making it poorly suited for evaluating modern structured-prediction methods.

In this work, we make the following contributions:
(1) We introduce StatSheets, a **publicly available dataset of 737 manually annotated spreadsheet sheets** collected from 14 public organizations publishing statistical data across multiple countries, languages, and spreadsheet formats. Unlike prior datasets, StatSheets jointly supports both CTC and TD under realistic conditions, including large sheets, multilingual content, heterogeneous layouts, and multiple spreadsheet formats (Sec. 4).
(2) We propose an **efficient end-to-end spreadsheet understanding pipeline combining structured cell-type prediction with deterministic table extraction** (Sec. 3), which operates in two sequential stages. First, a non-linear classifier predicts per-cell roles from a rich feature space combining lexical, formatting, positional, and row-/column-aware structural descriptors; these predictions are then refined through a pairwise Conditional Random Field (CRF) enforcing spatial coherence over the spreadsheet grid (Sec. 3.1). Second, table ranges are recovered through a deterministic 5-stage extraction algorithm operating on predicted cell-type grids (Sec. 3.2).
(3) We conduct a **thorough experimental study** of both tasks, comparing against non-DL and DL state-of-the-art approaches (Sec. 5). Our experiments show that: (i) on CTC, our CRF-LightGBM pipeline achieves a Mean File-Macro F_1 of 0.937, within 0.6 pp of the Transformer-based TUTA model, while requiring substantially lower computational resources and avoiding GPU dependence (Sec. 5.1); (ii) on TD, our deterministic extraction algorithm consistently outperforms generic region-based methods and remains competitive with recent LLM-based systems, again with orders of magnitude of computational cost savings (Sec. 5.2). These results show that **carefully designed structured models, combining non-linear feature modeling with structured spatial inference, remain highly competitive for spreadsheet understanding**, particularly in realistic large-scale data-management settings **where scalability, robustness, and interpretability are critical**.

We discuss related work in Sec. 2. The StatSheets dataset, our implementation of SpreadsheetLLM, and the code to reproduce the experiments are available at [13]. An extended version of this paper is available in [14].

2 Related Work

Cell-Type Classification (CTC) in Spreadsheets
Methods in this area have evolved from feature-engineered, to large pre-trained neural architectures. Early work [3] uses an undirected graphical model combining formatting and content cues to recover annotation-to-data mappings for relational integration, with semi-automatic error repair, emphasizing spatial dependencies but requiring human interaction and targeting relational extraction rather than general-purpose CTC. [23] introduces the first dedicated CTC framework using a Random Forest (RF) over formatting, content, and positional features to classify cells independently, later extended with active learning by [16]. Our RF-Koci baseline follows [23] with a similar feature design. [5] introduces Pytheas, a fuzzy-rule-based system classifying CSV rows into roles such as header, data, and context from column coherency and semantic keywords. Its formulation assigns a single label to each spreadsheet row. Thus, it cannot represent multiple tables appearing side by side, making precise 2-dimensional TD impossible. It also precludes cell-level CTC, since all cells in a row share the same label, despite spreadsheets commonly mixing roles within rows (e.g., header then data cells). Recent works use DL approaches with semantic and contextual representations. [15] combines pre-trained cell embeddings, recurrent architectures and formatting features. [7] proposes a multi-task framework jointly addressing TD, structural component recognition, and CTC using BERT embeddings, while [30] combines neural embeddings with Probabilistic Soft Logic constraints encoding structural regularities. Neither released code nor datasets. TUTA [34] extends Transformers with structure-aware self-attention capturing spreadsheet layout and semantics, pre-trained on large spreadsheet and Web table corpora. It substantially outperforms earlier neural methods; we therefore use it as our main DL baseline (Sec. 5.1). ForTaP [4] extends TUTA with formula-aware pre-training objectives, but relies on expert-authored formulas.

Table Detection (TD) in Spreadsheets
We identify two categories: *region-based* methods (extracting generic grid structures) and *table-specific* ones (modeling table semantics and structure to extract table boundaries).

Region-based approaches. [6] groups adjacent non-empty cells into connected components represented by bounding boxes. A purely structural, learning-free approach targeting dense regions rather than tables, it is sensitive to sparsity and cannot distinguish tables from metadata. [32] introduces Mondrian, which represents spreadsheets as binary images segmented into homogeneous rectangular regions through density-based clustering and compares regions across files to identify recurring templates. Although learning-free and unsupervised, both methods detect generic structural regions rather than tables. Their evaluation in *region-anchored mode* overestimates true TD quality; as shown in Sec. 5.2, performance drops sharply at higher IoU thresholds, highlighting the limits of generic region extraction for precise TD.

Table-specific approaches. Other works explicitly focus on predicting table boundaries. [21] proposes the graph-based *Remove and Conquer* (RAC) framework, which infers cell layout roles before constructing graphs over spatially related regions to detect table boundaries through curated rules. As one of the earliest coupled

CTC+TD pipelines, it is also the prior approach closest to ours. [24] extends RAC with a genetic optimization procedure maximizing a fitness function over predicted cell-type grids. Although effective with accurate CTC predictions, it is sensitive to classification errors and noise [32]. Neither [21] nor [24] releases code, datasets, or trained models, preventing direct comparison. [8] introduces TableSense, a CNN-based end-to-end TD framework formulating detection as object detection over featurized cell grids. Trained on the proprietary WebSheet10K dataset, it is reported as a strong specialized TD system, though no code, models, or datasets were released; later, SpreadsheetLLM, from the same Microsoft research group, reports substantially outperforming it. SpreadsheetLLM [9] encodes spreadsheets into compact token sequences, and fine-tunes LLMs for TD and related tasks (details in Sec. 5.2). We include it as a direct competitor using our own implementation, derived from the supplementary materials of [9].

End-to-end Spreadsheet Understanding

Few works address spreadsheet understanding as unified CTC+TD pipelines. [21], later extended by [24], couples the tasks through rule-based extraction but releases neither code nor trained models. [20] demonstrates XLIndy, an interactive Excel add-in for layout inference and table recognition, without an evaluation or sharing the code. [7] jointly addresses TD, structural component recognition, and CTC via a multi-task FCNN with strictly coarse-to-fine dependencies: TD feeds component recognition, which then feeds CTC, without reverse interaction. In contrast, our pipeline performs CTC based solely on the sheet and derives table boundaries deterministically and exclusively from the predicted cell-type grid. Moreover, [7] relies on a proprietary dataset and releases no implementation. Overall, prior end-to-end systems rely on rule-based extraction, proprietary implementations, or coarse-to-fine designs where TD is not influenced by downstream CTC. In contrast, we derive table boundaries solely from CTC predictions while jointly evaluating both tasks under cross-validation.

Datasets for Spreadsheet Understanding

The only publicly available dataset covering both CTC and TD is DECO [22], built from the Enron email archive [17]. DECO is unsuitable in our setting for several reasons. First, it excludes files larger than 5 MB, while large tables are common in public data portals, e.g., Eurostat tables regularly exceed hundreds of MB [11]. Second, it excludes non-Western languages such as Arabic and Japanese; this introduces semantic and structural biases (e.g., right-to-left layout in Arabic spreadsheets). Third, based on Enron emails (2000–2001), it reflects outdated formatting practices. Finally, annotations only cover visible content: hidden rows/columns (via zero height/width or explicit hide/unhide) are unlabeled, leaving large parts of sheets unannotated and making the dataset incompatible with spatial and sequential methods that require complete grid annotations. Several smaller CTC-only benchmarks also exist. DeEx [10] provides 444 sheets with a six-type label scheme, while SAUS [3] and CIUS [15] follow the same scheme on narrow domains. However, all three provide only CTC labels, lack TD annotations, and have limited linguistic, formatting, and structural diversity; DeEx and SAUS are no longer accessible. The proprietary WebSheet10K dataset used in [8, 9, 34] is also unavailable publicly. A subset of related annotations (VEnron2, VEUSES, VFUSE) released by [8] contains only TD table ranges (no CTC labels) and is restricted to English XLS/XLSX files derived from Enron/EUSES. To facilitate extensive evaluation at scale, we therefore introduce StatSheets (Sec. 4), a publicly available dataset addressing these limitations: 737 sheets in six languages across five file formats, with complete cell-level CTC and table-range TD annotations collected from contemporary statistical spreadsheets published by international organizations.

3 Our Approach

We decompose spreadsheet understanding into two stages (Figure 1). CTC (Sec. 3.1) assigns each cell a label using a non-linear classifier combined with a pairwise CRF enforcing spatial coherence. TD (Sec. 3.2) takes the resulting label grid and recovers axis-aligned table extents through a deterministic, learning-free algorithm.

3.1 Cell-Type Classification

Let $C = \{\text{EMPTY}, \text{HEADER}, \text{DATA}, \text{TITLE}, \text{OTHER}\}$ be the set of five cell-type classes. We formulate CTC as a structured prediction problem on a 2-dimensional grid, where each cell is a node, and edges connect horizontal and vertical neighbors. Each label $\hat{y}_i \in C$ should be assigned considering both local features and neighborhood consistency. This motivates a two-stage pipeline in which a non-linear classifier produces per-cell class probabilities that serve as unary potentials for a pairwise CRF which enforces spatial coherence.

Problem 1 (Cell-Type Classification). *A spreadsheet region is represented as a 2-dimensional grid $G = (V, E)$, where each node $i \in V$ corresponds to a cell and $E \subseteq V \times V$ is the set of pairs of horizontally or vertically adjacent cells. Given G and some features describing its elements, the CTC problem consists of predicting a class for each cell, or, equivalently, predicting for all cells the label vector $\hat{\mathbf{y}} \in C^{|V|}$.*

Features

First, we rely on *65 unary features* (65-dimensional vector) for each cell, organized into three groups (see Table 1; we detail them in the extended version [14]). *Group A* contains content and positional features describing intrinsic cell properties and local context, *Group B* contains formatting and layout features extracted from rich spreadsheet formats (XLSX, XLS, ODS), with default values for plain-text formats (CSV, TSV). *Group C* contains row-/column-aware statistical features, capturing global structural context.

Pairwise features (30 features). For each edge (i, j) in the 2D grid, we compute a feature vector describing the relationships between adjacent cells. These features fall in four groups. *Orientation features* encode the relative horizontal and vertical positioning between cells. *Type features* encode agreement and transitions, including whether cells share the same type (empty, numbers, strings, dates, or formula) and type transitions (empty–non-empty, number–string, number–date). *Content difference features* capture absolute differences in text statistics (character and digit counts, jointly starting with a letter, etc.). *Formatting features* measure visual agreement: same font size and name, consistent bold/italic, same font and background colors, shared-border presence, horizontal/vertical alignment differences, and whether both cells are part of merged regions.

Classification Pipeline

The classification pipeline addresses two complementary limitations of cell-wise prediction. First, most Group A and B features

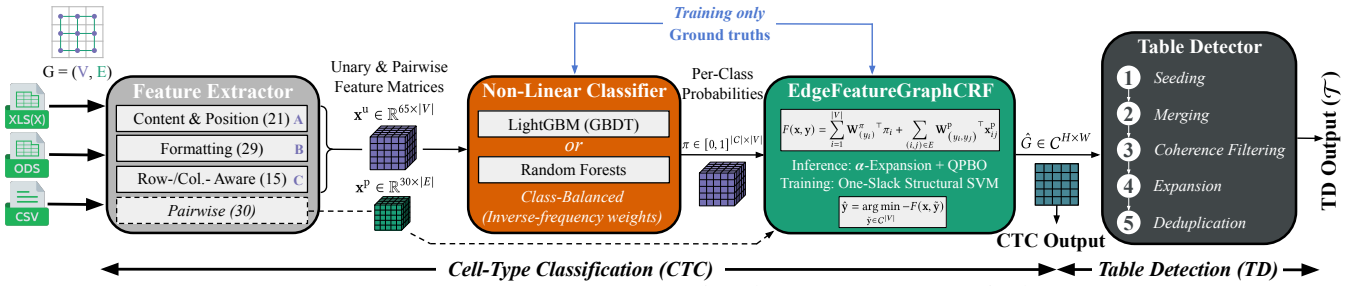


Figure 1: End-to-end *Cell-Type Classification* (CTC) and *Table Detection* (TD) pipeline.

Table 1: All 65 unary features used in the CTC task, organized into the three groups A, B, and C.

	ID	Name	Description	ID	Name	Description	ID	Name	Description
Group A	1	IsEmpty	No cell content	8	#Chars	Character count	15	RowPosRatio	Relative row index in grid
	2	IsNum	Value is numeric	9	#Digits	Digit count	16	ColPosRatio	Relative column index in grid
	3	IsStr	Value is a string	10	#Alpha	Alphabetic character count	17	IsSheetBorder	Cell on non-empty boundary
	4	IsDate	Value is a date	11	#Spaces	Space count	18	#Neighbors	Non-empty 4-neighbor count
	5	IsFormula	Cell contains a formula	12	#Other	Other character count	19	IsIsolated	All 4-neighbors are empty
	6	IsInt	Integer value	13	StartsAlpha	First char. is alphabetic	20	IsOds	File format is ODS
	7	IsFloat	Non-integer numeric value	14	StartsNum	First char. is numeric	21	IsExcel	File format is XLS/XLSX
Group B	22	Bold	Font weight is bold	27–29	FontColor	Font color (R, G, B $\in [0,1]$)	36	BorderTop	Has top border
	23	Italic	Font style is italic	30–32	BgColor	Background col. (R, G, B $\in [0,1]$)	37	BorderBottom	Has bottom border
	24	Underline	Text is underlined	33	IsMerged	Cell belongs to a merged region	38–44	HAlign	Horiz. alignment (7-class one-hot)
	25	FontSize	Font size (pt)	34	BorderLeft	Has left border	45–50	VAlign	Vert. alignment (6-class one-hot)
	26	FontHash	Stable hash of font name	35	BorderRight	Has right border			
Group C	51	RowNonEmptyRatio	Fraction non-empty in row	56	IsAloneInRow	Only non-empty cell in row	61	IsAloneInCol	Only non-empty cell in column
	52	RowFracNum	Fraction numerical in row	57	ColNonEmptyRatio	Fraction non-empty in column	62	ColTypeOutlier	Type differs from col. majority
	53	RowFracStr	Fraction string in row	58	ColFracNum	Fraction numerical in column	63	MergeColRatio	Fraction merged cells in column
	54	RowHomog	Type homogeneity in row	59	ColFracStr	Fraction string in column	64	MergeRowRatio	Fraction merged cells in row
	55	RowIsUniform	All non-empty row cells same type	60	ColHomog	Type homogeneity in column	65	MergeColSpan3+	Merge spans $\geqslant 3$ columns

are binary or categorical (Bold, IsNum, StartsAlpha…), and class membership typically depends on their joint interactions rather than individual signals; as an example, HEADER cells often combine formatting and row-level homogeneity cues that are not linearly separable. This motivates a *non-linear classifier*, which captures feature conjunctions better than linear models, as confirmed empirically in Sec. 5.1. Second, cell labels exhibit strong spatial dependencies in the grid: a cell may be HEADER partly because it precedes a DATA region, and TITLE cells are often defined by their structural position relative to headers and data blocks. To model these dependencies, we use a *pairwise CRF* to propagate information across adjacent cells and enforce neighborhood consistency.

Step 1: LightGBM classifier. We train a Gradient-Boosted Decision Tree (GBDT) classifier using LightGBM [19] on the 65 unary features of all training cells. Compared to Random Forests (RF) [2], GBDT iteratively fits trees to correct residual errors of the current ensemble, yielding stronger learners on heterogeneous feature sets combining binary flags, counts, ratios, and color values. Class imbalance (DATA and EMPTY cells dominate most sheets) is handled by reweighting classes inversely to their frequency in the loss. The model outputs a per-class probability vector $\boldsymbol{\pi}_i = \text{LightGBM}\left(\mathbf{x}_i^{\text{u}}\right) \in [0,1]^{|C|}$ for each cell i, where $\mathbf{x}_i^{\text{u}}$ is the 65D feature vector.

Step 2: EdgeFeatureGraphCRF. Building on the grid G defined above, we define a first *CRF scoring* function as:

$$F^{\text{Linear}}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{|V|} \mathbf{W}^{\text{u}}_{(y_i)}{}^{\top}\mathbf{x}_i^{\text{u}} + \sum_{(i,j)\in E} \mathbf{W}^{\text{p}}_{(y_i,y_j)}{}^{\top}\mathbf{x}_{ij}^{\text{p}} \qquad (1)$$

where $\mathbf{x}_i^{\text{u}} \in \mathbb{R}^{65}$ and $\mathbf{x}_{ij}^{\text{p}} \in \mathbb{R}^{30}$ are the unary and pairwise feature vectors, $\mathbf{W}^{\text{u}} \in \mathbb{R}^{65\times|C|}$ contains one weight vector per class, and $\mathbf{W}^{\text{p}} \in \mathbb{R}^{30\times|C|^2}$ one per ordered label pair. We use the EdgeFeatureGraphCRF of [27], which learns a distinct pairwise weight vector $\mathbf{W}^{\text{p}}_{(k,l)}$ per label pair (k,l), allowing label-specific pairwise interactions rather than shared across transitions. For instance, a HEADER→DATA transition may be encouraged when neighboring cells differ in alignment or font weight, and discouraged otherwise. Eq. (1) corresponds to a purely linear CRF using the 65 unary features. To incorporate the non-linear LightGBM classifier, we replace the feature vector $\mathbf{x}_i^{\text{u}}$ with the class-probability vector $\boldsymbol{\pi}_i$, yielding:

$$F^{\text{LightGBM}}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{|V|} \mathbf{W}^{\pi}_{(y_i)}{}^{\top}\boldsymbol{\pi}_i + \sum_{(i,j)\in E} \mathbf{W}^{\text{p}}_{(y_i,y_j)}{}^{\top}\mathbf{x}_{ij}^{\text{p}} \qquad (2)$$

where $\mathbf{W}^{\pi} \in \mathbb{R}^{|C|\times|C|}$. In Sec. 5.1, we compare both cases experimentally: Eq. (1) as CRF-Linear, and Eq. (2) as CRF-LightGBM.

MAP (Maximum A Posteriori) inference. At inference, the labeling for a spreadsheet $\mathbf{x}$ is obtained by maximizing the function F. Exact MAP inference over a general pairwise CRF is NP-hard [29], so we use the α-expansion algorithm [1], via the *PyQPBO* library. It is a move-making algorithm that iteratively proposes a candidate label $\alpha \in C$ and, for each α, jointly optimizes over all cells the binary decision of keeping the current label or switching to α. This reduces the multi-class problem to a sequence of binary graph-cut subproblems until no move decreases the energy (convergence). Each binary subproblem is solved using Quadratic Pseudo-Boolean Optimization (QPBO) [28], which reformulates binary energy minimization ($-F(\mathbf{x},\tilde{\mathbf{y}})$) as a minimum graph cut: pairwise terms penalizing label disagreement between neighbors (attractive terms) are solved exactly in polynomial time [25], while terms that encourage label disagreement (repulsive terms) are handled approximately [28].

This yields exactly the MAP result when all pairwise interactions are attractive, and a strong approximation otherwise, while remaining tractable for grids of several thousand cells seen in practice [25].

CRF training. CRF parameters $\mathbf{W}$ ($\mathbf{W}^{\mathrm{u}}/\mathbf{W}^{\pi}$ and $\mathbf{W}^{\mathrm{p}}$) are learned by minimizing the structured hinge loss via the structural SVM framework [31]. Given n training sheets with ground-truth labelings $\{\mathbf{y}_t^*\}_{t=1}^n$, the idea is to find $\mathbf{W}$ such that the score $F(\mathbf{x}_t, \mathbf{y}_t^*)$ exceeds the score of any competing labeling $\bar{\mathbf{y}}_t$ by a margin proportional to how wrong $\bar{\mathbf{y}}_t$ is (measured by the Hamming loss $\Delta(\mathbf{y}_t^*, \bar{\mathbf{y}}_t)$, i.e., the fraction of mislabeled cells). A slack variable $\xi \geqslant 0$ absorbs margin violations when the constraint cannot be satisfied exactly. In the n-slack variant, one slack per training sheet is introduced, leading to a cutting-plane problem with $O(n/\varepsilon^2)$ iterations to reach ε-accuracy. The one-slack formulation [18] instead introduces a *single* slack variable summed over all training examples:

$$\min_{\mathbf{W},\,\xi\geqslant 0} \frac{1}{2}\|\mathbf{W}\|^2 + \lambda\xi \quad \text{s.t.} \quad \forall\ (\bar{\mathbf{y}}_t)_{t=1}^n \in \prod_{t=1}^{n} C^{|V_t|},$$
$$\frac{1}{n}\sum_t \mathbf{W}^\top\left[\Psi(\mathbf{x}_t, \mathbf{y}_t^*) - \Psi(\mathbf{x}_t, \bar{\mathbf{y}}_t)\right] \geqslant \frac{1}{n}\sum_t \Delta(\mathbf{y}_t^*, \bar{\mathbf{y}}_t) - \xi$$

where $\Psi(\mathbf{x}_t, \mathbf{y})$ is the joint feature map (concatenation of unary and pairwise sufficient statistics) and $\lambda > 0$ controls the regularization strength. This constraint enforces the margin *on average* across all sheets rather than per sheet, reducing the cutting-plane complexity to $O(1/\varepsilon)$ iterations regardless of n. The trade-off is that per-sheet margin violations can be hidden by other sheets, potentially allowing a small number of poorly labeled sheets to go uncorrected. However, the n-slack variant does not scale to large datasets, and the *average* constraint remains sufficiently representative in our setting where n exceeds 500 sheets per fold (see Sec. 4 and Sec. 5 for dataset information and experimental setup).

Handling class imbalance in the linear CRF baseline. For the linear CRF ablation (CRF-Linear, Sec. 5.1) using raw unary features $\mathbf{x}_i^{\mathrm{u}}$ instead of $\boldsymbol{\pi}_i$, class imbalance must be handled explicitly. We extend pystruct's EdgeFeatureGraphCRF with a class-balanced structured hinge loss, weighting each cell by the inverse square root of its class frequency within the training fold. We additionally evaluate an optional *hard-EMPTY clamping* strategy: for cells with IsEmpty=1, a large negative bias is applied to all non-EMPTY unary scores, effectively preventing the CRF from predicting other classes. Analogously, we prevent prediction of EMPTY when IsEmpty=0. These mechanisms are unnecessary in the final CRF-LightGBM, as class imbalance is already handled by LightGBM, and empty cells naturally receive near-zero values to non-EMPTY classes.

3.2 Table Detection

For a spreadsheet region of H rows and W columns, we have $H \times W = |V|$ where V is the cell (vertex) set from Problem 1. Having obtained the cell type prediction $\hat{\mathbf{y}}$ as discussed in Sec. 3.1, let $\hat{G} \in C^{H\times W}$ be the predicted cell-type label grid. Formally:

Problem 2 (Table Detection). *Given $\hat{G}$, the TD problem consists of predicting a set of non-overlapping, axis-aligned rectangles $\mathcal{T} = \{T_k\}$, where each T_k is the minimal bounding box enclosing a maximal coherent group of HEADER and DATA cells forming a single table in $\hat{G}$.*

We solve this with a deterministic algorithm that requires no learning. It follows a five-stage pipeline mirroring the heuristics a human annotator would apply when delineating tables on a grid of class labels: (1) *seeding* extracts maximal connected regions of HEADER/DATA cells; (2) *merging* fuses pairs of regions separated only by within-table artifacts (sub-header rows, inter-table titles, thin empty bands); (3) *filtering* discards sparse or incoherent candidates; (4) *expansion* recovers locally disconnected top/left/right header bands; (5) *deduplication* resolves residual overlaps. The method uses a small set of interpretable hyperparameters with default values provided when introduced (summarized in [14]).

Notation and definitions. Rows are indexed by $r \in \{0, \ldots, H-1\}$ and columns by $c \in \{0, \ldots, W-1\}$. We introduce:

Definition 3 (Ranges and Densities). *A* range *is a non-empty axis-aligned rectangle $R = [r_0, r_1] \times [c_0, c_1]$ with area $|R| = (r_1 - r_0 + 1)(c_1 - c_0 + 1)$. The* Intersection-over-Union *and the (asymmetric)* containment ratio *of two ranges are, respectively,* $\mathrm{IoU}(R, R') = \frac{|R\cap R'|}{|R\cup R'|}$, $\mathrm{cont}(R, R') = \frac{|R\cap R'|}{|R|}$. *For a class subset $S \subseteq C$, the S-*density *of R is $\rho_S(R) = \frac{1}{|R|}\sum_{(r,c)\in R} \mathbb{1}\left[\hat{G}_{r,c} \in S\right]$. For a row r and a column span $[c_0, c_1]$, the* absolute *and* relative *header densities are: $\eta(r, c_0, c_1) = \rho_{\{HEADER\}}(r \times [c_0, c_1])$ and $\eta^\star(r, c_0, c_1) = \frac{\rho_{\{HEADER\}}(r\times[c_0,c_1])}{\rho_{C\setminus\{EMPTY\}}(r\times[c_0,c_1])}$, with $\eta^\star = 0$ by convention when the denominator is 0. The definitions of $\eta(c, r_0, r_1)$ and $\eta^\star(c, r_0, r_1)$ are symmetric for columns.*

Above, the *absolute* score η measures how much of a column range is covered by HEADER cells. Empty cells reduce η, which is useful when deciding whether a row separates two table candidates: a partial header should not span both tables. The *relative* score $\eta^\star$, instead, measures the proportion of *non-empty* cells in a row that are headers, ignoring empty padding. This is useful when deciding whether a sparse row acts as a header: rows that span wider than the data block below them often contain blank cells above groups of columns, so η would be diluted by those empty cells even for a genuine header row, whereas $\eta^\star$ remains high.

Definition 4 (Seed Mask and Seeds). *The* seed mask *of $\hat{G}$ is the binary matrix $M \in \{0,1\}^{H\times W}$ defined by $M_{r,c} = \mathbb{1}\left[\hat{G}_{r,c} \in \{HEADER, DATA\}\right]$ for all (r, c). We define the graph (V_M, E_M) as $V_M = \left\{(r, c) : M_{r,c} = 1\right\}$, $E_M = \{\{(r, c), (r', c')\} : |r - r'| + |c - c'| = 1\}$ (horizontal/vertical adjacency). A* seed *S is a connected component of (V_M, E_M). We derive its* row *and* column extents $r_{\min}(S)$, $r_{\max}(S)$, $c_{\min}(S)$, $c_{\max}(S)$, *and* bounding box $B(S) = [r_{\min}(S), r_{\max}(S)] \times [c_{\min}(S), c_{\max}(S)]$. *We write $n_D(S) = \sum_{(r,c)\in S} \mathbb{1}[\hat{G}_{r,c} = DATA]$ and $n_H(S)$ analogously for HEADER. Seeds$(\hat{G})$ is the set of all seeds.*

The full detection procedure TableRangeExtractor$(\hat{G})$ is given in Algorithm 1: starting from a set of seeds, it repeatedly merges compatible candidates, filters incoherent ones, expands the remaining candidates into table ranges, and removes duplicates.

Stage 1: Seeding. We define a table as the bounding box covering its HEADER and DATA cells (excluding TITLE and OTHER cells). We compute the connected components of the graph (V_M, E_M); each yields a candidate seed (line 1 of Algorithm 1). We retain only seeds with $n_D \geqslant n_D^{\min} = 4$ data cells, or header-only seeds ($n_D = 0$) with $n_H \geqslant n_H^{\min} = 8$: the latter preserves genuine header bands disconnected from their data block by over-segmentation; such seeds can later be recovered during the merging stage. Figure 2 illustrates the seeding on a two-table grid, yielding three seeds.

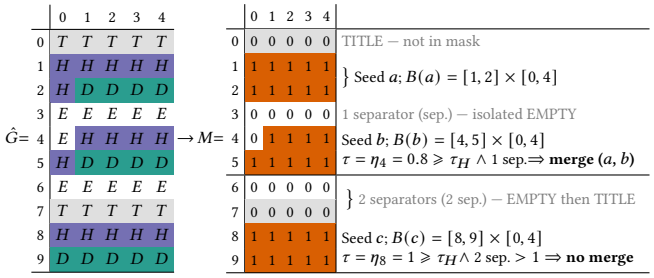


Figure 2: Seeding and merging on a two-table grid $\hat{G}$ (■ TITLE, ■ HEADER, ■ DATA, white EMPTY). $M_{r,c} = \mathbb{1}[\hat{G}_{r,c} \in \{H, D\}]$ yields seeds *a*, *b*, *c*. η_r abbreviates $\eta(r, 0, 4)$.

Algorithm 1 TableRangeExtractor($\hat{G}$)

Require: Cell-type grid $\hat{G} \in C^{H \times W}$
Ensure: Set of detected table ranges $\mathcal{T}$

$\mathcal{S} \leftarrow \textsc{Seeds}(\hat{G})$
$\mathcal{S} \leftarrow \{S \in \mathcal{S} : n_D(S) \geqslant n_D^{\min} \vee (n_H(S) \geqslant n_H^{\min} \wedge n_D(S) = 0)\}$
while $\exists (S_i, S_j) \in \mathcal{S}^2, i \neq j$, s.t. $\textsc{ShouldMerge}(S_i, S_j, \hat{G})$ **do**
  Pick any such pair (S_i, S_j)
  $\mathcal{S} \leftarrow (\mathcal{S} \setminus \{S_i, S_j\}) \cup \{S_i \cup S_j\}$
$\mathcal{S} \leftarrow \{S \in \mathcal{S} : \textsc{Coherent}(S, \hat{G})\}$
$\mathcal{T} \leftarrow \{\textsc{Expand}(S, \hat{G}) : S \in \mathcal{S}\}$
return $\textsc{Deduplicate}(\mathcal{T})$

Stage 2: Merging. Connected components in the raw seed mask are **systematically over-segmented** for two reasons. **1.** Tables may contain *sub-header rows* splitting the data region (e.g., section dividers between row groups). These rows are correctly classified as HEADER, but their column-wise structure may differ from that of the top header, leaving them disconnected from rows above and below, splitting a single table into multiple seeds. **2.** Adjacent tables on the same sheet are often separated by a thin band of EMPTY rows together with a TITLE row introducing the next table; on the seed mask this appears as empty space, which does not reliably distinguish *inter-table separation* from *intra-table structure*. Figure 2 illustrates both: seeds *a* and *b* are separated by an empty row, but row 4 acts as a sub-header sufficiently overlapping both column structures, thus triggering a merge. In contrast, seeds *b* and *c* are separated by an empty row *and* a title row; this blocks merging. We address both cases (**1.** and **2.**) via the predicate ShouldMerge (Algorithm 2), applied iteratively until a fixed point (lines 3–5 of Algorithm 1); each round strictly decreases $|\mathcal{S}|$, ensuring convergence. The *vertical* case (lines 1–11 of Algorithm 2) handles seeds with overlapping column spans separated by a vertical gap. Along the gap, an intermediate row with header density $\eta \geqslant \tau_H = 0.4$ over the shared column range is read as an interior sub-header and triggers immediate merge; otherwise, the decision combines the lower seed's first-row density with the number of separator rows (containing only EMPTY/TITLE cells over the shared span), reflecting that intra-table interruptions are typically shorter than inter-table whitespace ($g_v = 4$). The *horizontal* case (lines 12–17) is symmetric and simpler: seeds are merged when fewer than $g_h = 2$ fully empty columns separate them. This suffices because horizontal gaps lack the sub-header and title artifacts that complicate the vertical case.

Algorithm 2 ShouldMerge: merge predicate.

Require: Seeds S_i, S_j; grid $\hat{G}$
Ensure: Boolean indicating whether to merge S_i and S_j

1: $c_l \leftarrow \max(c_{\min}(S_i), c_{\min}(S_j))$; $c_r \leftarrow \min(c_{\max}(S_i), c_{\max}(S_j))$
2: **if** $c_l \leqslant c_r$ **then**
3: $r_t \leftarrow \min(r_{\max}(S_i), r_{\max}(S_j))$; $r_b \leftarrow \max(r_{\min}(S_i), r_{\min}(S_j))$
4: **if** $r_b > r_t + 1$ **then**
5: sep $\leftarrow 0$
6: **for** r from $r_t + 1$ to $r_b - 1$ **do**
7: **if** $\eta(r, c_l, c_r) \geqslant \tau_H$ **then return true**
8: **if** $\hat{G}_{r,c} \in \{\text{EMPTY}, \text{TITLE}\}$ for every $c \in [c_l, c_r]$ **then**
9: sep $\leftarrow$ sep + 1
10: $\tau \leftarrow \eta(r_b, c_l, c_r)$
11: **return** $(\tau \geqslant \tau_H \wedge \text{sep} \leqslant 1) \vee (\tau < \tau_H \wedge \text{sep} < g_v)$
12: $r_l \leftarrow \max(r_{\min}(S_i), r_{\min}(S_j))$; $r_r \leftarrow \min(r_{\max}(S_i), r_{\max}(S_j))$
13: **if** $r_l \leqslant r_r$ **then**
14: $c_t \leftarrow \min(c_{\max}(S_i), c_{\max}(S_j))$; $c_b \leftarrow \max(c_{\min}(S_i), c_{\min}(S_j))$
15: **if** $c_b > c_t + 1$ **then**
16: ev $\leftarrow |\{c \in [c_t + 1, c_b - 1] : \forall r \in [r_l, r_r], \hat{G}_{r,c} = \text{EMPTY}\}|$
17: **return** $(\text{ev} < g_h)$
18: **return false**

Stage 3: Coherence filtering. Merging may still produce two types of **invalid candidates**: (*i*) large, sparse seeds formed by repeated merges connecting distant regions via narrow bridges of HEADER or DATA cells—large bounding boxes with little table content; (*ii*) data blocks whose header was misclassified, surviving seeding without corresponding to a valid table. The predicate $\textsc{Coherent}(S, \hat{G})$ handles both cases. It rejects a seed if both densities in $B(S)$ fall below thresholds, i.e., $\rho_{\{\text{DATA}\}}(B(S)) < \rho_D^{\min}$=0.05 *and* $\rho_{\{\text{HEADER}\}}(B(S)) < \rho_H^{\min} = 0.5$. Data-only seeds are rejected when they contain fewer than $n_D^{\dagger} = 20$ DATA cells and have no HEADER cell within Chebyshev radius r_N=3 of $B(S)$. Header-only seeds are retained iff $n_H \geqslant n_H^{\min}$.

Stage 4: Expansion. By construction, a surviving seed S contains the data block of a table and header cells 4-connected to it in the seed mask. However, **some header cells may remain disconnected** because of empty padding rows, merged-cell rendering artifacts, or isolated cells misclassified as TITLE or OTHER. We recover them via a directional expansion procedure (Algorithm in [14]) that grows the seed bounding box upward, leftward, and rightward, but not downward, since headers in statistical spreadsheets conventionally appear above or beside the data, never below. Expansion continues while the next row (resp. column) has relative header density $\eta^\star \geqslant \tau_H$ (resp. τ_H^c), two thresholds set to 0.4. We use $\eta^\star$ rather than η because header rows often contain empty padding above column groups, where η would be diluted by those blanks even on genuine headers. Column scans are evaluated over the seed's *original* row range, so side-header columns remain anchored to the original data region rather than to newly recovered top headers.

Stage 5: Deduplication. In rare cases, **two distinct seeds yield overlapping table ranges**, e.g., when a single header band spans two nearby column-aligned data blocks. We resolve these with a Non-Maximum-Suppression (NMS) step: ranges are processed in decreasing order of area, and a range R is accepted unless an already-accepted range R' satisfies $\text{IoU}(R, R') \geqslant \tau_{\text{IoU}} = 0.5$ (strong overlap)

or $\text{cont}(R, R') \geqslant \tau_{\text{cont}} = 0.8$ (near-containment). The remaining ranges form the final detection set $\mathcal{T}$.

4 Dataset

Given the limitations of existing datasets for CTC and TD (Sec. 2), we construct and release StatSheets [13], a new dataset of spreadsheets annotated for both tasks, originating from statistical datasets published by public organizations.

Construction

The dataset is highly heterogeneous, with respect to: (i) the data sources, (ii) the language used in the sheets, (iii) the covered topics, (iv) the sheet sizes, and (v) the file formats. This enables evaluating the robustness of cell-type classifiers and table detectors in highly variable settings, which is critical for downstream applications relying on clean and structured tables, such as fact-checking [11].

We consider 14 selected sources for the dataset (detailed composition in the extended version [14]), chosen for their geographical diversity, which also increases language diversity. Topic diversity results from the broad scope of the selected sources. The collection proceeds as follows. We first retrieve a large pool of spreadsheets from each source using a state-of-the-art focused Web crawler [12]. Each spreadsheet is then processed to identify candidate sheets likely to contain statistical tables using a simple but effective heuristic: a sheet is retained if at least 50% of its non-empty cells are either purely numerical or predominantly numerical (i.e., a majority of characters are digits, allowing symbols or precision annotations). To limit source imbalance, we retain at most 100 candidate sheets per source and only one sheet per spreadsheet file. The resulting candidates are randomly shuffled and manually annotated, keeping only sheets containing at least one statistical table. Annotation continues until reaching 750 sheets, corresponding to ∼150 test sheets under the 5-fold cross-validation setup of Sec. 5. A second sanitization pass then corrects annotation errors and removes ambiguous cases, yielding a final dataset of **737 annotated sheets**, written in six languages: English, French, Arabic, Japanese, Spanish, and Icelandic. The file formats are: XLSX (316 sheets, 42.9%), XLS (296, 40.2%), ODS (60, 8.1%), CSV (52, 7.1%), and TSV (13, 1.8%). Rich spreadsheet formats (XLS, XLSX, ODS) provide formatting information; plain-text formats (CSV, TSV) do not.

Annotation Protocol and Statistics

Each retained sheet is annotated for both the CTC and TD tasks. For **CTC**, we annotate each cell within the minimal bounding rectangle containing all the non-empty cells of the sheet. We use the five labels: EMPTY (0), HEADER (1), DATA (2), TITLE (3), and OTHER (4). HEADER includes column/row headers and sub-headers, which may be adjacent to higher-level headers or appear between DATA cells. Each DATA cell is associated with at least one (and typically several) HEADER cell. TITLE cells describe table content and are usually, though not necessarily, located above it. OTHER covers remaining non-empty cells (e.g., footnotes, annotations, data sources, measurement details), acting as a reject class. Overall, **7 570 354 cells** are annotated for CTC, with an average of 10 272 per sheet. Sheet sizes vary considerably, from 27 to 1 542 303 cells, illustrating strong structural variability in real-world spreadsheets. The largest sheet is particularly useful for scalability evaluation (Sec. 5.2). Class distribution is heavily imbalanced: DATA cells account for 62.2% of the annotations, followed by HEADER (20.4%) and EMPTY (17.1%), while TITLE (<0.1%) and OTHER (<0.2%) are rare.

Table 2: Main characteristics of the CTC systems.

Model	NL	Seq.	Fmt.	Spat.	Sem.	GPU
CRF-Linear	✗	✓	✓	✓	✗	✗
RF-Koci	✓	✗	✓	✗	✗	✗
RF	✓	✗	✓	✓	✗	✗
LightGBM	✓	✗	✓	✓	✗	✗
CRF-RF	✓	✓	✓	✓	✗	✗
CRF-LightGBM	✓	✓	✓	✓	✗	✗
TUTA	✓	✓	✓	✓	✓	✓

For the **TD** task, we annotate *table ranges*. Since each DATA cell is associated with at least one HEADER, a table is defined as the rectangular bounding box containing all HEADER and DATA cells belonging to it. Consequently, table ranges contain only HEADER and DATA cells, excluding TITLE and OTHER. Moreover, each HEADER or DATA cell can belong to only one table, meaning that overlapping tables are not allowed. In practice, if two regions share HEADER or DATA cells, they are considered part of the same table, potentially with hierarchical headers. These constraints are used as automatic sanitization checks to detect annotation inconsistencies before a manual sanitization. After annotation, sequences of more than two consecutive fully empty rows or columns are compressed to exactly two, to save space and without losing information.

Our 737 sheets lead to **818 annotated tables**. Most sheets (690) contain a single table; 29 contain two, and 18 contain three or more.

5 Experiments

Secs. 5.1 and 5.2 respectively present the CTC and TD experiments. All systems are evaluated using 5-fold cross-validation: sheets are shuffled with a fixed random seed and split into five folds of ≈147 sheets each, yielding an 80/20 train/test split per fold. Depending on the system, the training fold is used for training, fine-tuning, hyperparameter selection, or not used. The test fold is shared across tasks for consistency. When a system fails on some sheets, we report results on the subset successfully processed by all systems.

5.1 Cell-Type Classification

We compare several CTC systems, classified in Table 2 by their inclusion of spatial awareness (**Spat.**), sequential modeling (**Seq.**), and non-linearity (**NL**). Hyperparameter choices are discussed in the extended version [14].

CRF-Linear is the CRF model described in *Step 2* of Sec. 3.1. It uses $\mathbf{x}^{\text{u}}$ and $\mathbf{x}^{\text{p}}$ for the unary and pairwise components, and thus follows the score function of Eq. (1). The model includes the class-balanced loss and EMPTY-case clamping described in Sec. 3.1. This method has sequential and spatial modeling, but is linear.

RF-Koci is a baseline based on *Step 1* of Sec. 3.1, where LightGBM is replaced by a RF classifier. It does not use the full unary feature set, but only the non-spatial features, excluding all Group C features as well as feature #19 from Group A. We retain feature #18, the only neighborhood-aware feature from [23]. This baseline is therefore a non-linear approach comparable to [23], without sequential modeling, and with minimal spatial awareness. **RF** is similar to RF-Koci but uses the complete unary feature set ($\mathbf{x}^{\text{u}}$), thus it is non-linear.

LightGBM is a baseline following *Step 1* of Sec. 3.1, based on optimized GBDT. Like RF, it is non-linear, models spatial awareness, but without sequential modeling.

CRF-RF applies a CRF on top of the RF baseline (Sec. 3.1), with LightGBM replaced by RF (non-linear, sequential, with spatial awareness). We train the RF model and the CRF sequentially (the CRF using RF probabilities $\boldsymbol{\pi}$ as unary inputs, following Eq. (2)) on each training fold, keeping the RF hyperparameters fixed and selecting CRF hyperparameters via the same 5-fold validation procedure.

CRF-LightGBM combines a CRF model on top of LightGBM, similarly to CRF-RF. The CRF scoring function thus follows Eq. (2).

TUTA [34] (Sec. 2) is pre-trained on large collections of unlabeled Web and spreadsheet tables, then fine-tuned on CTC (which we focus on) and table type classification (TTC), which assumes known table boundaries, i.e., the output of our TD task. TUTA is the only DL baseline we consider; it outperforms other DL-based CTC methods [34]. We use the official implementation [26], adapted to our label set and pipeline. We fine-tune the last two layers of the 12-layer backbone and the classification head, reinitializing from the pre-trained checkpoint at each fold.

CTC Metrics

The five cell types (recall C from Sec. 3.1) are heavily imbalanced across sheets: DATA cells usually dominate, while TITLE and OTHER cells are sparse and unevenly distributed. Standard cell-level accuracy is therefore uninformative, and cell-level macro F1 is distorted by sheet size as larger sheets weigh more in the aggregate score.

Instead, we measure **per-class file-macro F1** (FM-F1), which treats each sheet as a single observation. Let $\mathcal{F}$ be the test sheets and C the classes. For $f \in \mathcal{F}$ and $c \in C$, let $F_1(f, c)$ denote the F1 score of class c on sheet f. Denoting $\mathcal{F}_c \subseteq \mathcal{F}$ the set of sheets in which class c is present (in ground truth or predictions), the per-class file-macro F1 is $\text{FM-F1}_c = \frac{1}{|\mathcal{F}_c|} \sum_{f \in \mathcal{F}_c} F_1(f, c)$. Absent classes from a sheet are excluded from FM-F1_c, avoiding both trivially perfect scores and unfair penalization of rare classes. We summarize system performance with the **mean file-macro F1** (M-FM-F1), the macro-average of FM-F1_c across all classes $\text{M-FM-F1} = \frac{1}{|C|} \sum_{c \in C} \text{FM-F1}_c$.

CTC Computational Efficiency

All systems except TUTA run on **Machine 1**, a Dell PowerEdge R730 with two Intel Xeon E5-2640 v4 processors (40 logical cores at 2.40 GHz), 128 GB of DDR4 ECC RAM, and **no GPU**. TUTA needs a GPU and we ran it on **Machine 2**, a Dell PowerEdge R770 having two Intel Xeon 6505P processors (48 logical cores), 256 GB of DDR5 RAM, and one NVIDIA RTX PRO 6000 Blackwell Server Edition GPU (96 GB VRAM). To estimate equivalent cloud costs, we map both machines to the closest Amazon EC2 On-Demand instances available in `us-east-1`: `m5.8xlarge` (32 vCPUs, 128 GiB RAM, $1.536/hr) for Machine 1 (denoted **C**) and `g7e.8xlarge` (32 vCPUs, 256 GiB RAM, 1× NVIDIA RTX PRO 6000 Blackwell, $5.268/hr) for Machine 2 (**G**). Tables 3 and 4 report runtimes and estimated costs over the full 5-fold cross-validation. File loading and feature extraction (*Load*) are reported separately from model computation (*Train*/*Infer*), although both contribute to the total billed time. The results show a clear computational gap between the proposed systems and TUTA. Training all six non-DL systems combined costs <$21, while TUTA alone exceeds $135 due to extended GPU training. Inference costs are smaller overall, but CPU-only TUTA inference remains much slower and costlier than GPU inference. All non-DL systems complete inference within 15 minutes at negligible cost.

Table 3: Training time and estimated cloud cost (5-fold total). C = `m5.8xlarge` ($1.536/hr); G = `g7e.8xlarge` ($5.268/hr).

Model	Load (s)	Train (s)	Total (h)	Inst.	Cost ($)
RF-Koci	256.7	31.6	0.08	C	0.12
RF	252.2	120.3	0.10	C	0.16
LightGBM	254.7	177.1	0.12	C	0.18
CRF-Linear	252.3	14 658.2	4.14	C	6.36
CRF-RF	250.5	15 848.4	4.47	C	6.87
CRF-LightGBM	254.3	14 952.5	4.22	C	6.49
TUTA	178.5	92 376.0	25.71	G	135.44

Table 4: Inference time and estimated cloud cost (5-fold total). †CPU-only run of TUTA for reference.

Model	Load (s)	Infer (s)	Total (h)	Inst.	Cost ($)
RF-Koci	540.0	40.0	0.16	C	0.25
RF	555.2	49.0	0.17	C	0.26
LightGBM	541.8	95.9	0.18	C	0.27
CRF-Linear	540.1	268.5	0.22	C	0.35
CRF-RF	550.6	300.8	0.24	C	0.36
CRF-LightGBM	542.8	288.7	0.23	C	0.35
TUTA†	363.8	30 994.6	8.71	C	13.38
TUTA	203.4	679.1	0.25	G	1.29

CTC Results

Figure 3 reports the M-FM-F1 scores of all seven systems averaged over the 5 folds. Markers indicate per-fold values, and error bars, the min–max range. Figure 4 shows the corresponding per-class FM-F1_c scores using the same color scheme.

Effect of spatial awareness, non-linearity, and sequential modeling. We assess the contribution of each component comparing successive bars in Figure 3. CRF-Linear (sequential, spatial, linear) achieves an M-FM-F1 of 0.894, close to RF-Koci (0.899), non-linear but with minimal spatial awareness and no sequential modeling. Adding full spatial features (RF-Koci → RF) improves performance by +2.1 pp (0.899 → 0.920). Adding sequential modeling through the CRF layer (RF → CRF-RF) yields another +1.1 pp, while replacing RF with LightGBM (CRF-RF → CRF-LightGBM) adds +0.6 pp. Overall, *all three components contribute positively and combine effectively.* TUTA reaches 0.943, only +0.6 pp above CRF-LightGBM (0.937).

Per-class analysis. The overall gain from RF-Koci to CRF-LightGBM is +3.8 pp in M-FM-F1, but improvements are uneven across classes. As shown in Figure 4, EMPTY is almost perfectly classified by all systems, while DATA is already saturated (FM-F1 ≈ 0.976–0.981). Gains are larger for harder classes: HEADER improves by +2.8 pp, TITLE by +8.5 pp (0.816 → 0.901), and OTHER by +7.4 pp (0.773 → 0.847). These classes are harder to distinguish using formatting alone and benefit from added structural context provided by spatial and sequential modeling. TUTA performs notably better on OTHER (+4.5 pp), the most semantically dependent class, but slightly worse on TITLE (−1.9 pp), which relies more on structural and formatting cues. These results suggest that *semantic modeling mainly benefits the most content-dependent class*, while *structural and non-linear*

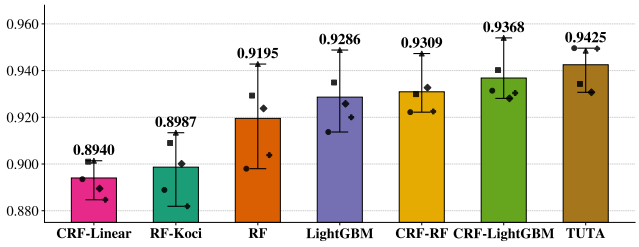


Figure 3: Mean File-Macro F_1 per model. Error bars span per-fold min/max; individual fold scores are shown as markers.

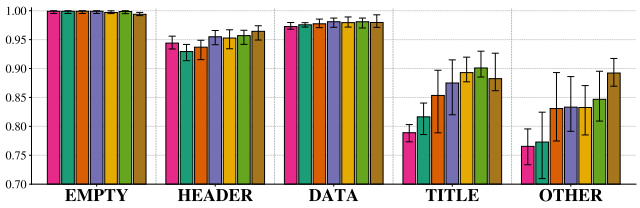


Figure 4: Per-class file-macro F_1 breakdown; error bars span per-fold min/max.

modeling already capture most of the remaining signal. However, TUTA's benefits come at a substantially higher computational cost.

5.2 Table Detection

We compare six TD systems, as follows.

TD(RF-Koci) and **TD(CRF-LightGBM)** apply the deterministic TableRangeExtractor (Algorithm 1, Sec. 3.2) to the grids predicted by RF-Koci and CRF-LightGBM, respectively.

TD(Oracle) applies TableRangeExtractor to ground-truth cell-type grids, providing an upper bound on TD performance under perfect CTC predictions and isolating the TD stage from CTC errors.

Connected-Component (CC) is the region-based baseline from [32], following [6]. We use the implementation from the Mondrian repository [33]. As discussed when presented in Sec. 2, CC detects contiguous *regions* rather than semantic tables, potentially splitting tables or merging them with metadata; it is evaluated in *region-anchored* mode, yielding an optimistic precision estimate.

Mondrian [32] is the unsupervised, training-free approach for spreadsheet region detection and layout template inference presented in Sec. 2. Like CC, Mondrian detects generic regions rather than semantic tables and is therefore evaluated in *region-anchored* mode. We use the authors' region detection implementation [33], specifically the *dynamic* variant, which selects the clustering radius independently for each file, as this setting achieved the best results in the original work [33]; also, we noted substantial variability in suitable radii across spreadsheets. Mondrian only operates on CSV/TSV files and deliberately ignores formatting information.

SpreadsheetLLM (Mistral-7B) [9] is an LLM-based framework for spreadsheet understanding, evaluated on TD and question answering. Its core component, *SheetCompressor*, converts spreadsheet grids into compact token sequences by (*i*) identifying heterogeneous boundary rows and columns; (*ii*) encoding the cells located on or near these boundaries (within a fixed neighborhood of k rows/columns) into an inverted-index representation mapping values to compressed cell addresses. Lacking a complete implementation, we built one in Python (available in [13]), using the partial C# source code provided in the supplementary material as a starting point for the *SheetCompressor*. The original work reports better results fine-tuning proprietary LLMs. We choose to fine-tune `mistralai/Mistral-7B-Instruct-v0.2`, which achieves the best performance among the open models reported by the authors, motivated by reproducibility and cost considerations. We follow the hyperparameters from [9, Appendix G]. The model predicts table ranges in Excel notation (e.g., `A1:B5`). Even after fine-tuning, some predictions are malformed, requiring post-processing to recover valid results and reject those which cannot be interpreted: a practical limitation of generative approaches for structured prediction.

Table 5: Inference time and estimated cloud cost for TD (5-fold total). *Train.*: training time for SpreadsheetLLM; loading and CTC inference time for TD variants. †Assumes perfect CTC at no cost. ‡Fold 5 timed out after 7 days.

Model	Train. (s)	Infer. (s)	Total (h)	Inst.	Cost ($)
SpreadsheetLLM (train)	33 096.3	—	9.19	G	48.43
SpreadsheetLLM (infer)	—	1 954.8	0.54	G	2.86
TD(RF-Koci)	580.0	301.5	0.24	C	0.38
TD(CRF-LightGBM)	831.5	326.3	0.32	C	0.49
TD(Oracle)†	—	355.9	0.10	C	0.15
CC	—	2 723.7	0.76	C	1.16
Mondrian‡	—	>653 896	>181.6	C	>279.00

TD Metrics

A table is represented as a rectangular bounding box defined by its row and column extents in the cell grid. For a predicted box $\hat{t}$ and a ground-truth box t, we measure overlap with the standard $\mathrm{IoU}(\hat{t}, t) = \frac{|\hat{t} \cap t|}{|\hat{t} \cup t|}$ where areas are counted in cells.

Precision curve. We report precision as a function of the IoU threshold $\tau \in [0, 1]$. At threshold τ, a matched pair $(\hat{t}, t)$ is counted as a true positive if $\mathrm{IoU}(\hat{t}, t) \geqslant \tau$, and as a false positive otherwise. Precision at τ is thus $\mathrm{Precision}(\tau) = \frac{|\{(\hat{t},t):\mathrm{IoU}(\hat{t},t)\geq\tau\}|}{N_{\mathrm{pred}}}$. We report both **micro** precision (pooling all predictions across sheets) and **macro** precision (averaging per-sheet precision curves.

Matching. Predicted and ground-truth tables are matched greedily in decreasing order of IoU: each table on either side is matched at most once (tables cannot overlap), and pairs with IoU = 0 are not matched. Unmatched predicted tables count as false positives; unmatched ground-truth tables count as false negatives. Systems that predict tables directly use N_{pred} as the total number of predictions (*table-anchored* mode). For region-based systems (CC and Mondrian), evaluated in *region-anchored* mode, N_{pred} is capped at $\min(N_{\mathrm{pred}}, N_{\mathrm{gt}})$ to avoid penalizing over-segmentation that is intrinsic to the region detection paradigm. This favors region-based methods by discarding excess predictions, yielding an optimistic precision estimate. In both modes, since matching is one-to-one, every predicted table is either matched or unmatched, yielding $\mathrm{Recall}(\tau) = \mathrm{Precision}(\tau) \times N_{\mathrm{pred}}/N_{\mathrm{gt}}$ for the micro curve: recall is a constant rescaling of precision, thus we do not report it.

TD Computational Efficiency

Experimental infrastructure and cost estimation follow Sec. 5.1, using the same instances **C** and **G**. SpreadsheetLLM requires a GPU and runs on Machine 2 (**G**); others run on Machine 1 (**C**). Table 5

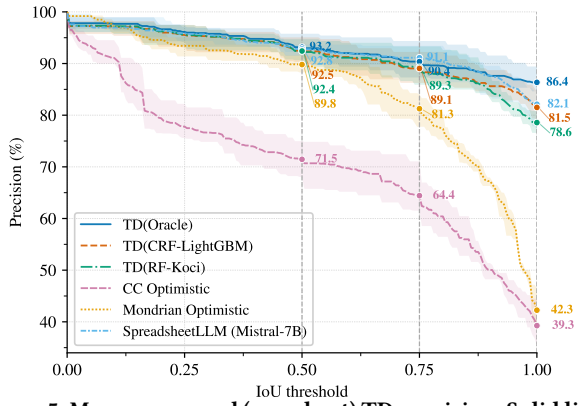


Figure 5: Macro-averaged (per-sheet) TD precision. Solid lines show fold means; shaded areas indicate min–max.

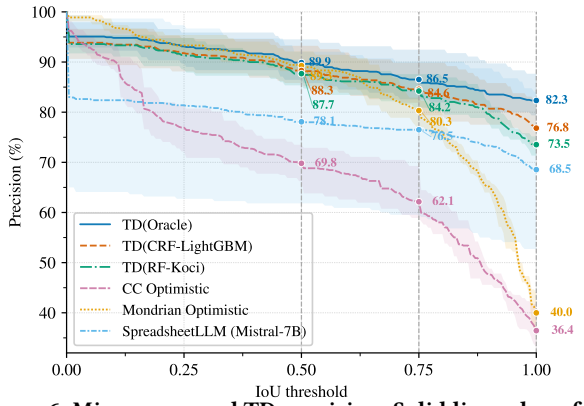


Figure 6: Micro-averaged TD precision. Solid lines show fold means; shaded areas indicate min–max.

reports runtimes and estimated costs over the full 5-fold cross-validation. For SpreadsheetLLM, *Train.* and *Infer.* correspond to fine-tuning and inference. For TD variants, *Train.* reports *loading and CTC inference time* of the underlying CTC model (in Table 4), while *Infer.* measures the deterministic TD stage alone. TD(Oracle) has zero training cost as perfect CTC predictions are assumed (unrealistic upper-bound). For CC and Mondrian, only inference time is reported, including file loading. For Mondrian, we interrupted fold 5 after 7 days (604 800 s) spent processing a spreadsheet with over 1M cells. The reported values combine those for folds 1 to 4 (49 095.9 s) with the full timeout duration; they are lower bounds.

The computational cost of TD is dominated by two outliers. Mondrian is by far the most expensive system due to the 7-day timeout in fold 5, yielding a lower-bound cost of $279. Excluding this, total cost is $53.47, of which $51.29 comes from SpreadsheetLLM alone due to GPU fine-tuning. By contrast, our TD variants are inexpensive: TD(RF-Koci)/TD(CRF-LightGBM) cost $0.38/$0.49 (CTC inference included), and CC costs $1.16. Accounting for both CTC training and inference, the full TD(CRF-LightGBM) pipeline amounts to $6.98 in total. Overall, the deterministic TD pipeline adds little overhead on top of the underlying CTC system and remains substantially cheaper than both GPU-based and unsupervised alternatives.

TD Results

Figures 5 and 6 report macro and micro precision as a function of τ for all systems. All methods consistently achieve higher macro than micro precision, indicating over-prediction of tables in larger/more complex spreadsheets, where micro metrics are more sensitive.

Effect of CTC quality on table detection. The TD variants directly reflect the quality of their underlying CTC systems. Replacing RF-Koci with CRF-LightGBM improves precision at IoU = 1 by +2.9 pp on macro (78.6 → 81.5%) and +3.3 pp on micro (73.5 → 76.8%). As shown in Figures 5 and 6, these gains mainly appear at high IoU thresholds, indicating more accurate boundary recovery rather than better coarse localization. This is consistent with the CTC results of Sec. 5.1: DATA and EMPTY cells are already well classified by all systems, while remaining errors mostly affect boundary-related classes such as TITLE, HEADER, and OTHER. Using perfect CTC predictions, TD(Oracle) reaches 86.4% macro and 82.3% micro precision at IoU = 1. Relative to TD(CRF-LightGBM), this leaves a margin of +4.9 pp and +5.5 pp respectively, suggesting that CTC quality gains would also translate into better table detection.

Region-based methods. Despite the optimistic evaluation protocol for CC and Mondrian, both region-based approaches remain substantially below TD variants at high IoU thresholds. At IoU = 1, Mondrian reaches 42.3% macro and 40.0% micro precision, trailing TD(CRF-LightGBM) by −39.2 pp and −36.8 pp. However, at lower IoU thresholds, it performs comparably to several other systems, indicating that its predicted regions often overlap tables but fail to recover precise boundaries. These results suggest that *generic region detection is insufficient for precise spreadsheet table extraction.*

Comparison with SpreadsheetLLM. SpreadsheetLLM is close to TD(CRF-LightGBM) in macro precision (+0.6 pp at IoU = 1), but substantially worse on micro precision (−8.3 pp, 76.8 → 68.5%). This larger macro/micro gap suggests *stronger over-prediction on complex spreadsheets.* We also observe *higher variance across folds*, indicating greater sensitivity to the training distribution. In addition, model outputs are occasionally inconsistent with the requested output format and require post-processing. Overall, TD(CRF-LightGBM) achieves competitive results while remaining substantially cheaper than SpreadsheetLLM. The oracle results further indicate that additional gains remain possible through improved CTC predictions.

6 Conclusion

High-quality, cost-efficient and reproducible methods for extracting information from spreadsheets are still needed. We provide a comprehensive annotated dataset and propose a new CTC-TD pipeline combining strong performance, low resource needs, and full TD interpretability. Our CTC method, combining spatial, sequential, and non-linear features, and our deterministic, learning-free TD method, match or outperform state-of-the-art DL competitors (TUTA for CTC, SpreadsheetLLM for TD) at a fraction of their cost and without requiring a GPU. With no black-box effects in TD, our corpus and pipeline advance the state of the art in spreadsheet understanding.

Acknowledgments

This work was funded in part by the French government under management of Agence Nationale de la Recherche (ANR) as part of the "France 2030" program, references ANR-23-IACL-0008 (PR[AI]RIE-PSAI) and ANR-23-IACL-0005 (Hi! PARIS Cluster 2030).

GenAI Usage Disclosure

We used an LLM tool (*Claude Code*, running Claude Sonnet 4.6) during the development of this work, on two occasions.

First, we used it to assist in the reproduction of the SpreadsheetLLM system described in [9], for which no full implementation code was provided. In particular, we leveraged it to reconstruct the *SheetCompressor* component from the C# code fragments provided in their supplementary material, to translate them into Python, and integrate them with the rest of the system's code. We also used Claude to re-implement the overall pipeline of [9] based on the methodological details described in the paper and its appendices. All generated or assisted code was manually reviewed and corrected when necessary. We verified correctness against the paper descriptions and ensured that no modifications altered the intended behavior of the original algorithms.

In addition, we used the LLM to clean our own codebase and improve its organization, as well as to refine CLI descriptions to enhance usability for reproducing the experiments. All such changes were reviewed to ensure behavioral equivalence and consistency with the experimental setup described in the paper.

No generative AI system was used to write any part of the current paper.

References

[1] Y. Boykov, O. Veksler, and R. Zabih. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23 (2001), 1222–1239.

[2] Leo Breiman. 2001. Random Forests. *Machine Learning* 45 (2001), 5–32.

[3] Zhe Chen and Michael Cafarella. 2014. Integrating spreadsheet data via accurate and low-effort extraction. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014*. 1126–1135.

[4] Zhoujun Cheng, Haoyu Dong, Ran Jia, Pengfei Wu, Shi Han, Fan Cheng, and Dongmei Zhang. 2022. FORTAP: Using Formulas for Numerical-Reasoning-Aware Table Pretraining. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL 2022*. 1150–1166.

[5] Christina Christodoulakis, Eric B. Munson, Moshe Gabel, Angela Demke Brown, and Renée J. Miller. 2020. Pytheas: Pattern-based Table Discovery in CSV Files. *Proc. VLDB Endow.* 13 (2020), 2075–2089.

[6] Remi Coletta, Emmanuel Castanier, Patrick Valduriez, Christian Frisch, DuyHoa Ngo, and Zohra Bellahsene. 2012. Public data integration with WebSmatch. In *Proceedings of the First International Workshop on Open Data, WOD 2012*. 5–12.

[7] Haoyu Dong, Shijie Liu, Zhouyu Fu, Shi Han, and Dongmei Zhang. 2019. Semantic Structure Extraction for Spreadsheet Tables with a Multi-task Learning Architecture. In *Proceedings of the 1st Workshop on Document Intelligence at NeurIPS 2019, DI 2019*.

[8] Haoyu Dong, Shijie Liu, Shi Han, Zhouyu Fu, and Dongmei Zhang. 2019. TableSense: Spreadsheet Table Detection with Convolutional Neural Networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI 2019*. 69–76.

[9] Haoyu Dong, Jianbo Zhao, Yuzhang Tian, Junyu Xiong, Mengyu Zhou, Yun Lin, José Cambronero, Yeye He, Shi Han, and Dongmei Zhang. 2024. Encoding Spreadsheets for Large Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024*. 20728–20748.

[10] Julian Eberius, Christopher Werner, Maik Thiele, Katrin Braunschweig, Lars Dannecker, and Wolfgang Lehner. 2013. DeExcelerator: a framework for extracting relational data from partially structured documents. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, CIKM 2013*. 2477–2480.

[11] Antoine Gauquier, Simon Ebel, Helena Galhardas, Théo Galizzi, Ioana Manolescu, Aurélien Peden, and Pierre Senellart. 2026. Efficient and Scalable Search for Statistics. In *Proceedings of the 42nd IEEE International Conference on Data Engineering, ICDE 2026*. Montréal, Canada, 1113–1126.

[12] Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. 2026. Efficient Crawling for Scalable Web Data Acquisition. In *Proceedings of the 29th International Conference on Extending Database Technology, EDBT 2026*. Tampere, Finland, 372–385.

[13] Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. 2026. StatSheets: dataset and reproducibility kit for spreadsheet table understanding. https://doi.org/10.5281/zenodo.21967189.

[14] Antoine Gauquier, Ioana Manolescu, and Pierre Senellart. 2026. Structured Prediction for Scalable Spreadsheet Table Understanding: From Cell Types to Table Ranges (Extended Version). arXiv:2608.16050 [cs.IR] https://arxiv.org/abs/2608.16050

[15] Majid Ghasemi-Gol, Jay Pujara, and Pedro A. Szekely. 2019. Tabular Cell Classification Using Pre-Trained Cell Embeddings. In *Proceedings of the 2019 IEEE International Conference on Data Mining, ICDM 2019*. 230–239.

[16] Julius Gonsior, Josephine Rehak, Maik Thiele, Elvis Koci, Michael Günther, and Wolfgang Lehner. 2020. Active Learning for Spreadsheet Cell Classification. In *Proceedings of the SEA Data 2020 Workshop of the EDBT/ICDT 2020 Joint Conference*.

[17] Felienne Hermans and Emerson R. Murphy-Hill. 2015. Enron's Spreadsheets and Related Emails: A Dataset and Analysis. In *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015*. 7–16.

[18] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. 2009. Cutting-plane training of structural SVMs. *Machine Learning* 77 (2009), 27–59.

[19] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, Vol. 30. 3146–3154.

[20] Elvis Koci, Dana Kuban, Nico Luettig, Dominik Olwig, Maik Thiele, Julius Gonsior, Wolfgang Lehner, and Oscar Romero. 2019. XLIndy: Interactive Recognition and Information Extraction in Spreadsheets. In *Proceedings of the 19th ACM Symposium on Document Engineering, DocEng 2019*. 25:1–25:4.

[21] Elvis Koci, Maik Thiele, Wolfgang Lehner, and Oscar Romero. 2018. Table Recognition in Spreadsheets via a Graph Representation. In *Proceedings of the 13th IAPR International Workshop on Document Analysis Systems, DAS 2018*. 139–144.

[22] Elvis Koci, Maik Thiele, Josephine Rehak, Oscar Romero, and Wolfgang Lehner. 2019. DECO: A Dataset of Annotated Spreadsheets for Layout and Table Recognition. In *Proceedings of the 15th International Conference on Document Analysis and Recognition, ICDAR 2019*. 1280–1285.

[23] Elvis Koci, Maik Thiele, Oscar Romero, and Wolfgang Lehner. 2016. A Machine Learning Approach for Layout Inference in Spreadsheets. In *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016)*. 77–88.

[24] Elvis Koci, Maik Thiele, Oscar Romero, and Wolfgang Lehner. 2019. A Genetic-Based Search for Adaptive Table Recognition in Spreadsheets. In *Proceedings of the 15th International Conference on Document Analysis and Recognition, ICDAR 2019*. 1274–1279.

[25] V. Kolmogorov and R. Zabin. 2004. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26 (2004), 147–159.

[26] Microsoft. 2021. Official Code Release for TUTA Table Understanding. https://github.com/microsoft/TUTA_table_understanding/tree/main/tuta.

[27] Andreas C. Müller and Sven Behnke. 2014. PyStruct - Learning Structured Prediction in Python. *Journal of Machine Learning Research* 15 (2014), 2055–2060.

[28] Carsten Rother, Vladimir Kolmogorov, Victor S. Lempitsky, and Martin Szummer. 2007. Optimizing Binary MRFs via Extended Roof Duality. In *Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2007*. 1–8.

[29] Solomon Eyal Shimony. 1994. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence* 68 (1994), 399–410.

[30] Kexuan Sun, Harsha Rayudu, and Jay Pujara. 2021. A Hybrid Probabilistic Approach for Table Understanding. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence, AAAI 2021*. 4366–4374.

[31] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research* 6, 50 (2005), 1453–1484. http://jmlr.org/papers/v6/tsochantaridis05a.html

[32] Gerardo Vitagliano, Lan Jiang, and Felix Naumann. 2021. Detecting Layout Templates in Complex Multiregion Files. *Proc. VLDB Endow.* 15 (2021), 646–658.

[33] Gerardo Vitagliano, Lucas Reisener, Lan Jiang, Mazhar Hameed, and Felix Naumann. 2021. Official Code Release for Mondrian. https://github.com/HPI-Information-Systems/Mondrian.

[34] Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. TUTA: Tree-based Transformers for Generally Structured Table Pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD 2021*. 1780–1790.