TableFormer: Table Structure Understanding with Transformers
Supplementary Material

Ahmed Nassar, Nikolaos Livathinos, Maksym Lysak, Peter Staar
IBM Research
{ahn,nli,mly,taa}@zurich.ibm.com

1. Details on the datasets

1.1. Data preparation

As a first step of our data preparation process, we have calculated statistics over the datasets across the following dimensions: (1) table size measured in the number of rows and columns, (2) complexity of the table, (3) strictness of the provided HTML structure and (4) completeness (i.e. no omitted bounding boxes). A table is considered to be simple if it does not contain row spans or column spans. Additionally, a table has a strict HTML structure if every row has the same number of columns after taking into account any row or column spans. Therefore a strict HTML structure looks always rectangular. However, HTML is a lenient encoding format, i.e. tables with rows of different sizes might still be regarded as correct due to implicit display rules. These implicit rules leave room for ambiguity, which we want to avoid. As such, we prefer to have "strict" tables, i.e. tables where every row has exactly the same length.

We have developed a technique that tries to derive a missing bounding box out of its neighbors. As a first step, we use the annotation data to generate the most fine-grained grid that covers the table structure. In case of strict HTML tables, all grid squares are associated with some table cell and in the presence of table spans a cell extends across multiple grid squares. When enough bounding boxes are known for a rectangular table, it is possible to compute the geometrical border lines between the grid rows and columns. Eventually this information is used to generate the missing bounding boxes. Additionally, the existence of unused grid squares indicates that the table rows have unequal number of columns and the overall structure is non-strict. The generation of missing bounding boxes for non-strict HTML tables is ambiguous and therefore quite challenging. Thus, we have decided to simply discard those tables. In case of PubTabNet we have computed missing bounding boxes for 48% of the simple and 69% of the complex tables. Regarding FinTabNet, 68% of the simple and 98% of the complex tables require the generation of bounding boxes.

Figure 1 illustrates the distribution of the tables across different dimensions per dataset.

1.2. Synthetic datasets

Aiming to train and evaluate our models in a broader spectrum of table data we have synthesized four types of datasets. Each one contains tables with different appearances in regard to their size, structure, style and content. Every synthetic dataset contains 150k examples, summing up to 600k synthetic examples. All datasets are divided into Train, Test and Val splits (80%, 10%, 10%).

The process of generating a synthetic dataset can be decomposed into the following steps:

1. Prepare styling and content templates: The styling templates have been manually designed and organized into groups of scope specific appearances (e.g. financial data, marketing data, etc.) Additionally, we have prepared curated collections of content templates by extracting the most frequently used terms out of non-synthetic datasets (e.g. PubTabNet, FinTabNet, etc.).

2. Generate table structures: The structure of each synthetic dataset assumes a horizontal table header which potentially spans over multiple rows and a table body that may contain a combination of row spans and column spans. However, spans are not allowed to cross the header - body boundary. The table structure is described by the parameters: Total number of table rows and columns, number of header rows, type of spans (header only spans, row only spans, column only spans, both row and column spans), maximum span size and the ratio of the table area covered by spans.

3. Generate content: Based on the dataset *theme*, a set of suitable content templates is chosen first. Then, this content can be combined with purely random text to produce the synthetic content.

4. Apply styling templates: Depending on the domain of the synthetic dataset, a set of styling templates is first manually selected. Then, a style is randomly selected to format the appearance of the synthesized table.

5. Render the complete tables: The synthetic table is finally rendered by a web browser engine to generate the

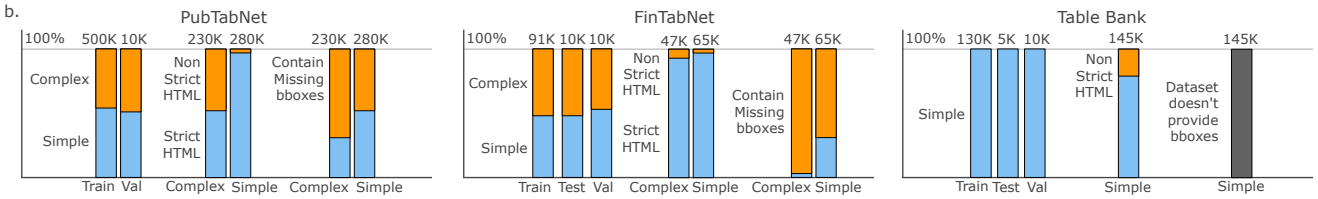

Figure 1: Distribution of the tables across different dimensions per dataset. Simple vs complex tables per dataset and split, strict vs non strict html structures per dataset and table complexity, missing bboxes per dataset and table complexity.

bounding boxes for each table cell. A batching technique is utilized to optimize the runtime overhead of the rendering process.

2. Prediction post-processing for PDF documents

Although TableFormer can predict the table structure and the bounding boxes for tables recognized inside PDF documents, this is not enough when a full reconstruction of the original table is required. This happens mainly due the following reasons:

- TableFormer output does not include the table cell content.
- There are occasional inaccuracies in the predictions of the bounding boxes.

However, it is possible to mitigate those limitations by combining the TableFormer predictions with the information already present inside a programmatic PDF document. More specifically, PDF documents can be seen as a sequence of PDF cells where each cell is described by its content and bounding box. If we are able to associate the PDF cells with the predicted table cells, we can directly link the PDF cell content to the table cell structure and use the PDF bounding boxes to correct misalignments in the predicted table cell bounding boxes.

Here is a step-by-step description of the prediction post-processing:

1. Get the minimal grid dimensions - number of rows and columns for the predicted table structure. This represents the most granular grid for the underlying table structure.

2. Generate pair-wise matches between the bounding boxes of the PDF cells and the predicted cells. The Intersection Over Union (IOU) metric is used to evaluate the quality of the matches.

3. Use a carefully selected IOU threshold to designate the matches as "good" ones and "bad" ones.

3.a. If all IOU scores in a column are below the threshold, discard all predictions (structure and bounding boxes) for that column.

4. Find the best-fitting content alignment for the predicted cells with good IOU per each column. The alignment of the column can be identified by the following formula:

$$\begin{aligned} alignment &= \arg\min_{c}\{D_c\} \\ D_c &= max\{x_c\} - min\{x_c\} \end{aligned} \tag{1}$$

where c is one of {left, centroid, right} and x_c is the x-coordinate for the corresponding point.

5. Use the alignment computed in step 4, to compute the median x-coordinate for all table columns and the median cell size for all table cells. The usage of median during the computations, helps to eliminate outliers caused by occasional column spans which are usually wider than the normal.

6. Snap all cells with bad IOU to their corresponding median x-coordinates and cell sizes.

7. Generate a new set of pair-wise matches between the corrected bounding boxes and PDF cells. This time use a modified version of the IOU metric, where the area of the intersection between the predicted and PDF cells is divided by the PDF cell area. In case there are multiple matches for the same PDF cell, the prediction with the higher score is preferred. This covers the cases where the PDF cells are smaller than the area of predicted or corrected prediction cells.

8. In some rare occasions, we have noticed that TableFormer can confuse a single column as two. When the post-processing steps are applied, this results with two predicted columns pointing to the same PDF column. In such case we must de-duplicate the columns according to highest total column intersection score.

9. Pick up the remaining orphan cells. There could be cases, when after applying all the previous post-processing steps, some PDF cells could still remain without any match to predicted cells. However, it is still possible to deduce the correct matching for an orphan PDF cell by mapping its bounding box on the geometry of the grid. This mapping decides if the content of the orphan cell will be appended to an already matched table cell, or a new table cell should be created to match with the orphan.

9a. Compute the top and bottom boundary of the horizontal band for each grid row (min/max y coordinates per row).

9b. Intersect the orphan's bounding box with the row bands, and map the cell to the closest grid row.

9c. Compute the left and right boundary of the vertical band for each grid column (min/max x coordinates per column).

9d. Intersect the orphan's bounding box with the column bands, and map the cell to the closest grid column.

9e. If the table cell under the identified row and column is not empty, extend its content with the content of the orphan cell.

9f. Otherwise create a new structural cell and match it wit the orphan cell.

Aditional images with examples of TableFormer predictions and post-processing can be found below.

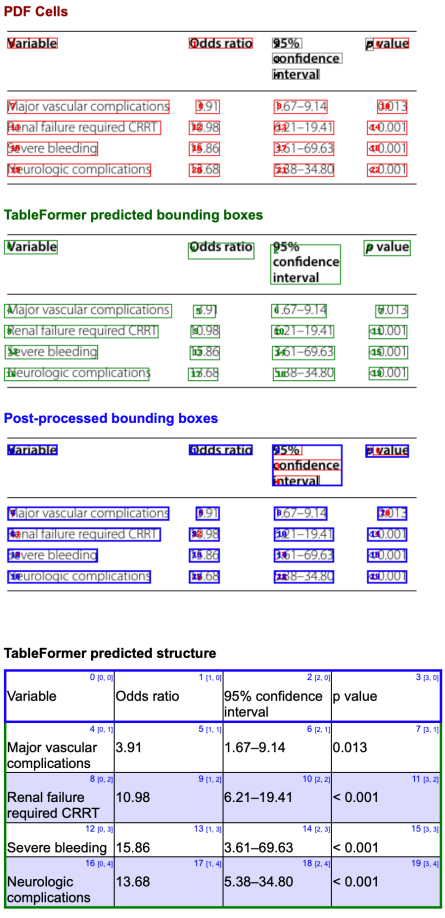

Figure 2: Example of a table with multi-line header.

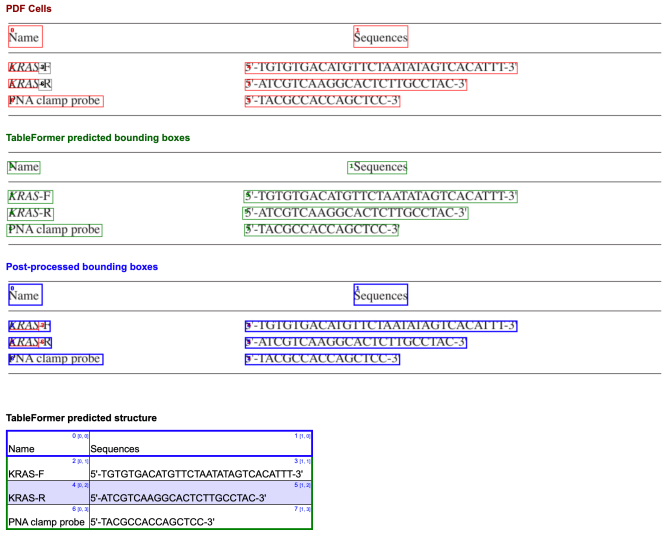

Figure 3: Example of a table with big empty distance between cells.

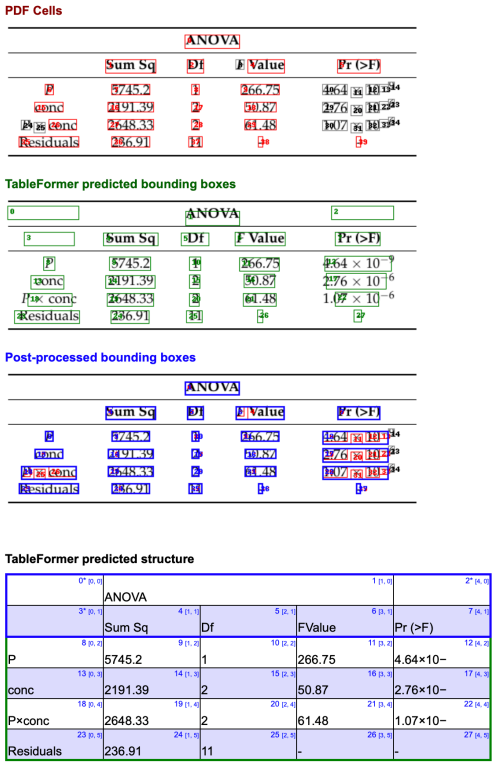

Figure 4: Example of a complex table with empty cells.

Figure 5: Simple table with different style and empty cells.

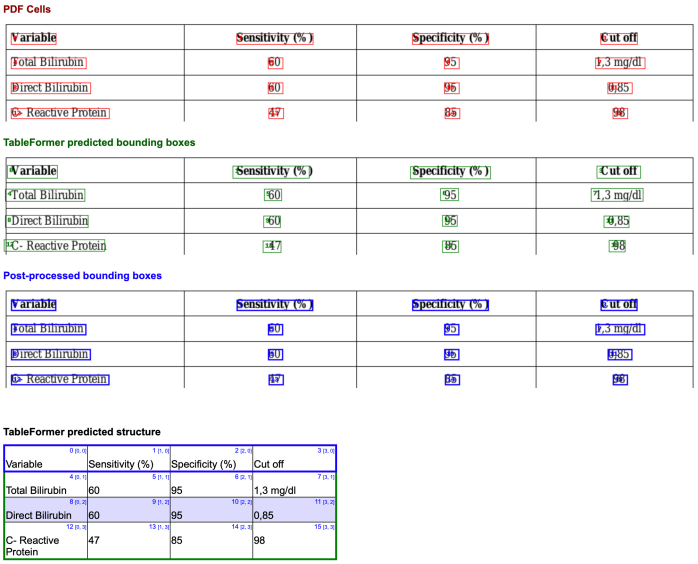

Figure 6: Simple table predictions and post processing.

Figure 7: Table predictions example on colorful table.

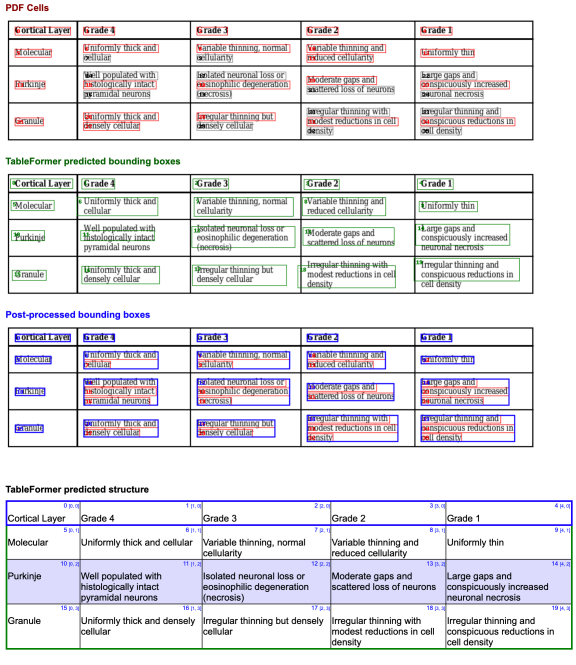

Figure 8: Example with multi-line text.

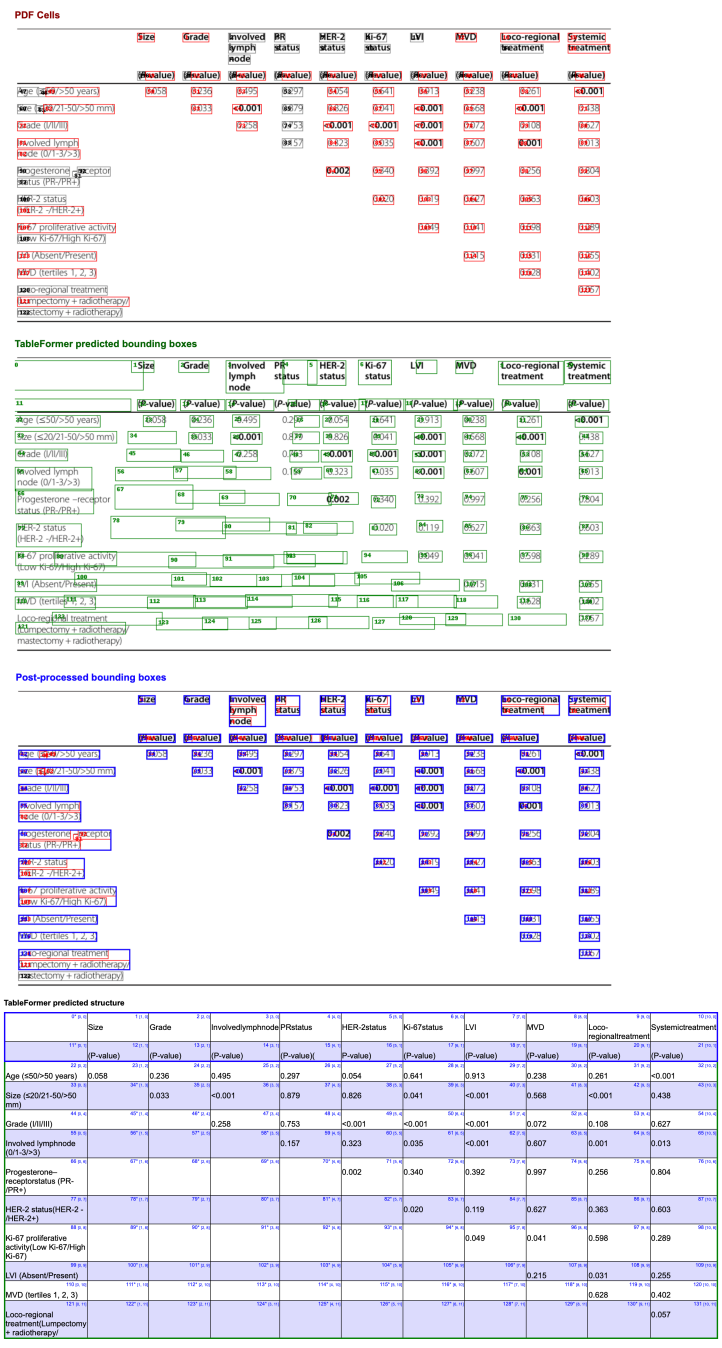

Figure 9: Example with triangular table.

PDF Cells

Parameter	Value
Gain	2851 V/V (69.09 dB)
Low Cut-Off Frequency	285 Hz
High Cut-Off Frequency	6580 Hz
Input-Referred Noise	2.1 μV (rms)
CMRR	110 dB @ 1 KHz
Number of Analog Channels	2
Power Consumption	1 mA @ 3.0 V (3 mW)
Precision	Selectable, 12 or 8 bits

TableFormer predicted bounding boxes

Parameter	Value
Gain	2851 V/V (69.09 dB)
Low Cut-Off Frequency	285 Hz
High Cut-Off Frequency	6580 Hz
Input-Referred Noise	2.1 μV (rms)
CMRR	110 dB @ 1 KHz
Number of Analog Channels	2
Power Consumption	1 mA @ 3.0 V (3 mW)
Precision	Selectable, 12 or 8 bits

Post-processed bounding boxes

Parameter	Value
Gain	2851 V/V (69.09 dB)
Low Cut-Off Frequency	285 Hz
High Cut-Off Frequency	6580 Hz
Input-Referred Noise	2.1 μV (rms)
CMRR	110 dB @ 1 KHz
Number of Analog Channels	2
Power Consumption	1 mA @ 3.0 V (3 mW)
Precision	Selectable, 12 or 8 bits

TableFormer predicted structure

Parameter	Value
Gain	2851 V/V (69.09 dB)
Low Cut-Off Frequency	285 Hz
High Cut-Off Frequency	6580 Hz
Input-Referred Noise	2.1μV (rms)
CMRR	110 dB @ 1 KHz
Number of Analog Channels	2
Power Consumption	1mA @ 3.0 V (3 mW)
Precision	Selectable, 12 or 8 bits

Figure 10: Example of how post-processing helps to restore mis-aligned bounding boxes prediction artifact.

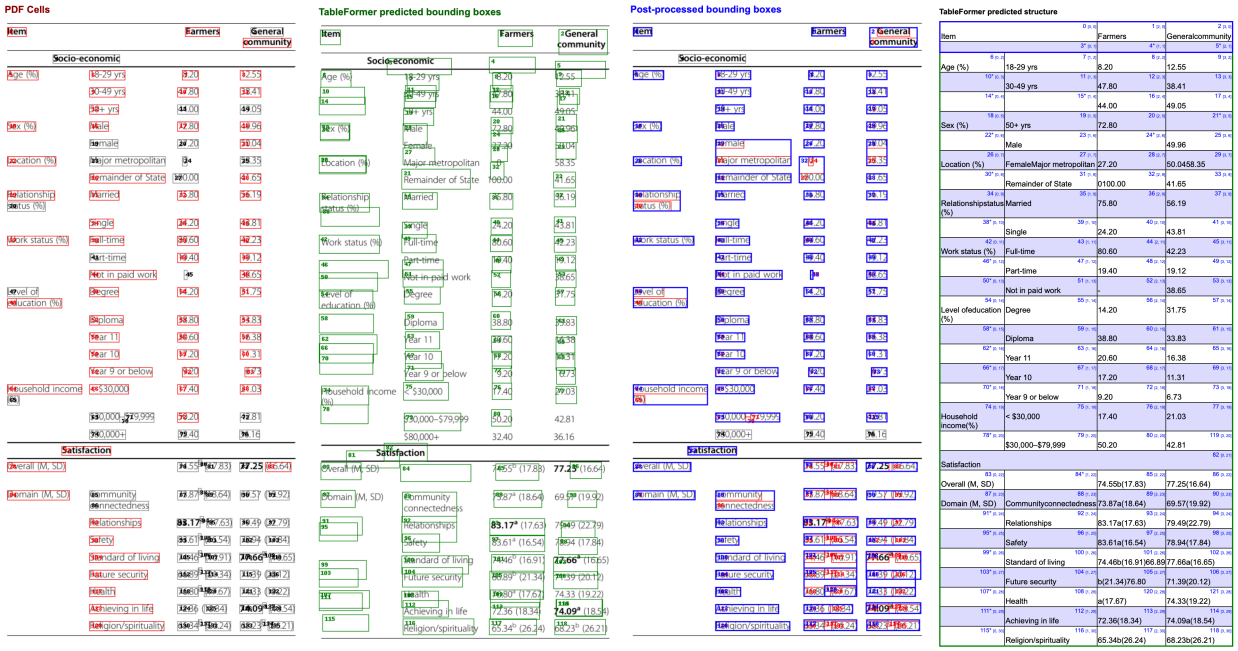

Figure 11: Example of long table. End-to-end example from initial PDF cells to prediction of bounding boxes, post processing and prediction of structure.