TableFormer: Table Structure Understanding with Transformers
Supplementary Material

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 7 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 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: