A Hybrid Probabilistic Approach for Table Understanding

Kexuan Sun Harsha Rayudu Jay Pujara
University of Southern California
Information Sciences Institute
kexuansu@usc.edu hrayudu@usc.edu jpujara@isi.edu

Abstract

Tables of data are used to record vast amounts of socioeconomic, scientific, and governmental information. Although humans create tables using underlying organizational principles, unfortunately AI systems struggle to understand the contents of these tables. This paper introduces an end-to-end system for *table understanding*, the process of capturing the relational structure of data in tables. We introduce models that identify cell types, group these cells into blocks of data that serve a similar functional role, and predict the relationships between these blocks. We introduce a hybrid, neuro-symbolic approach, combining embedded representations learned from thousands of tables with probabilistic constraints that capture regularities in how humans organize tables. Our neuro-symbolic model is better able to capture positional invariants of headers and enforce homogeneity of data types. One limitation in this research area is the lack of rich datasets for evaluating end-to-end table understanding, so we introduce a new benchmark dataset comprised of 431 diverse tables from data.gov. The evaluation results show that our system achieves the state-of-the-art performance on cell type classification, block identification, and relationship prediction, improving over prior efforts by up to 7% of macro F1 score.

1 Introduction

Tables are one of the most common ways to organize and present data. Due to the explosion of information published on the Web, billions of Web tables (Cafarella et al. 2008a) covering various topic domains are available. Such tables typically contain valuable information in a richly structured, relational form. Tables include information from UN surveys aiding developing nations (Division 2020), efforts tracking the spread of pandemics (Atlantic 2020), and the operation of the global economy (Bank 2020). For intelligent systems to have meaningful impacts, they must be able to understand the information contained within these tables. However, although tables are created following established logical structures and topological arrangements (Wang 1996), it is difficult for a machine to understand and extract knowledge from them due to the diversity and complexity of their layouts and contents. Figure 1 shows such an example table. It consists of cells with various data types and functional roles.

Tables generally contain relational information between different types of values, such as entities and quantitative measurements. To understand a table, an intelligent system must be able to understand the types of data within a table and the relationships between them. Tables are comprised of individual cells which contain a particular type of information. Cells are spatially organized into regions (or blocks) that share a common function. The relationship between the values in a table can be expressed as a relationship between these blocks. Our table understanding system adopts the paradigm of Pujara et al. (2019), which decomposes this problem into three tasks: *cell classification*, *block detection*, and *layout prediction*.

Many prior works have addressed the problems of table understanding in different, piecemeal ways. Traditionally, semantic typing approaches have sought to understand the data types within tables, but only operate when data are organized into well-defined, homogeneous columns. Prior cell classification approaches have focused on identifying functional roles at a cell level rather than capturing broader spatial relationships between tabular regions. These methods range from using handcrafted cell stylistic, formatting typographic features (Koci et al. 2016; Chen and Cafarella 2013), or embedded vector representations to classify cells (Ghasemi-Gol, Pujara, and Szekely 2019). Koci et al. (2019) have investigated finding larger, block structures in tables, however the primary goal of this approach is to correct imperfect cell classification results as a post-processing step for cell classification. To our knowledge, the task of layout prediction in tables has not been studied in prior work, although the broader problem of semantic modeling (Rümmele, Tyshetskiy, and Collins 2018) is an active area of research. Our work is the first to combine all three table understanding tasks in a single, end-to-end system. Furthermore, our novel neuro-symbolic method is able to capture both stylistic features and statistical patterns of prior approaches, and spatial relationships that encode how humans organize tabular data.

Despite the availability of vast amounts of tabular data on the Web, annotations for data types, functional regions, and relationships within these tables are very rare. Our approach is designed for this environment by combining unsupervised representation learning and higher-level symbolic constraints that capture frequent patterns in data. Specifi-

cally, we use high-dimensional table embeddings learned from thousands of tables in concert with a probabilistic graphical model, probabilistic soft logic (PSL), that captures structural patterns, such as consistency of types among contiguous cells. Together, these two techniques are capable of harnessing vast amounts of data while explicitly incorporating the intuitions human use when constructing tables.

Since previous work does not have a rich enough learning resource that covers all aspects of these tasks, we introduce a new benchmark dataset comprised of 431 tables downloaded from the U.S. Government's open data (Data.gov 2019). These tables are in different formats such as spreadsheets and comma-separated values, and from various topic domains. Most existing benchmark datasets (such as DeEx (DeEx 2013), SAUS (SAUS 2014) and CIUS (CIUS 2019)) consist of only Excel files, are from narrow domains and cover only cell functional types. Cross validation shows that the proposed system outperforms other baseline approaches in most cases. The results align with our hypotheses: the pre-trained cell embeddings encapsulate useful information of cells and probabilistic constraints can help enforce the logical consistency of prediction. Accordingly, the neuro-symbolic approach gets the best of both worlds. The idea of neuro-symbolic combination could potentially be applied to investigate other tasks for tables such as data transformation and relation extraction. In addition, the blocks and the relationships extracted by our system could potentially be used for automated tabular analysis such as finding important patterns from a table (Zhou et al. 2020).

We summarize our primary contributions:

- An integrated system solves three table understanding tasks inspired by human organization of tables: cell classification, block detection and layout prediction.
- A hybrid neuro-symbolic approach leveraging the strength of embedded cell representations with constraints capturing human layout patterns.
- A new and richer benchmark dataset with annotations for three tasks to support ongoing research in table understanding.

2 Preliminaries

2.1 PSL

Probabilistic Soft Logic (Bach et al. 2017) is a probabilistic inference framework. A PSL model is defined on a set of predicates and a set of first-order logic rules consisting of the predicates. Here is an example PSL rule:

$$w : P1(X, Y) \wedge P2(X, Z) \Rightarrow P3(Y, Z)$$

where w is the weight of this rule, $P1$, $P2$ and $P3$ are three predicates, and X, Y and Z are variables. During inference, the variables are grounded by constants. PSL uses hinge-loss Markov Random Fields (MRFs) to efficiently solve the convex optimization problem.

PSL has been successfully applied to solve different tasks such as knowledge base completion (Chen et al. 2019), recommender systems (Kouki et al. 2015), and entity resolution (Kouki et al. 2017).

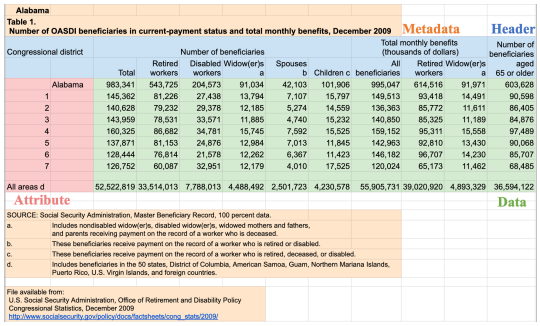


Figure 1: An example table with colored regions of cells.

2.2 Cell Embeddings

Cell embeddings are vector representations of tabular cells. Ghasemi-Gol, Pujara, and Szekely (2019) proposed to use neural networks to learn the cell embeddings. Each embedding consists of two parts, i.e. the *contextual embedding* captures the semantic information of the cell content, and the *stylistic embedding* encapsulates the stylistic features of the cell. There are different ways to capture the contextual and stylistic information. Specifically, the authors exploit pre-trained language models to identify the local context of cells and auto-encoders to encode the stylistic features. Accordingly, in our system, we use their pre-trained cell embedding model learned from thousands of tables.

3 Problem Definition

In this section, we reformalize tabular data and three table understanding tasks introduced in (Pujara et al. 2019). A table is a matrix $T = \{v_{i,j} | 1 \leq i \leq N, 1 \leq j \leq M\}$ where N is the number of rows, M is the number of columns, and $v_{i,j}$ is the cell in the i^{th} row and the j^{th} column.

3.1 Cell Classification

The goal of cell classification is to assign a label to each cell. In most prior work, cell classification is to assign each cell a label indicating the functional role the cell presents in the data layout. This process should consider both the cell information and context information of neighboring cells or even the whole table. In this paper, we classify cells only based on cell contents without considering the contextual information. We leave the task of identifying cell functional roles to the block detection task.

There are different ways to classify cell contents. Since the content usually is a sequence of characters, the most basic types can be *Number*, *String*, *Datetime* and *Empty*. However, these types are not informative enough. For example, knowing a cell contains a string is not helpful enough for understanding the table. In this paper, we use more fine-grained semantic data types. A number can be a *Nominal*, *Cardinal* or *Ordinal*, and a string can be the name of a *Person*, an *Organization*, a *Location*, or *Other* string. Particularly, this task can be even challenging for human. For example, a zip code is a *Nominal* but can be easily identified as a *Cardinal*, and a year is a *Datetime* but can also be classified as a *Cardinal* .

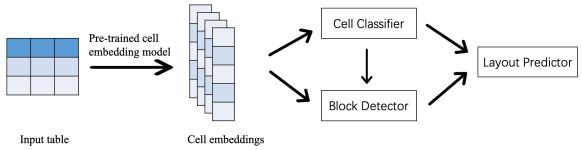


Figure 2: Our system architecture.

3.2 Block Detection

The goal of block detection is to identify regions of cells playing the same functional role. We refer to each such a region as a **block**. We define a block as a rectangle denoted as $\langle T, L, B, R \rangle$ where the four indices represent the *Top* row, *Left* column, *Bottom* row and *Right* column of the block, respectively. The smallest block is a single cell and the largest block is the table itself. When we treat each cell as a block, the system eventually solves the cell functional type classification. However, our goal is to identify regions as large as possible such that cells in the same region indeed play the same functional role. As shown in (Koci et al. 2019), considering blocks rather than individual cells could potentially reduce the number of misclassifications.

We consider the following functional roles: *Metadata* denotes the global information of the table (such as the title and source); *Header* indicates attribute names of the table columns; *Attribute* presents attribute names of the table rows; and *Data* shows the main content of the table.

3.3 Layout Prediction

Given the blocks identified from the previous task, the goal of layout prediction is to determine the relationships between blocks. Each pair of blocks will be assigned a relationship. We consider the following four relationships: *Subset_of* denotes that a block shows the subcategory information of another block (usually between two Header or Attribute blocks); *Header_of* indicates a block to contain the header information of another block (usually between a Header and a Data block); *Attribute_of* marks a block to record the attribute information of another block (usually between an Attribute and a Data block); and *Global_Attribute* marks a block containing the global information of another block (usually between a Metadata block and another block).

4 Method

We introduce our system for table understanding, illustrated in Figure 2. In specific, the system takes a table as the input and uses a pre-trained cell embedding model to represent each cell that is to be processed by several downstream predictors. More specifically, the cell classifier seeks to predict cell data types and the block detector uses both the embeddings and the data types to generate blocks. The layout predictor then predicts relationships between blocks. The cell classifier, block extractor and layout predictor model the three corresponding tasks as PSL problems. Our system depends on the computational performance of PSL. Empirically, the inference of PSL scales linearly with the number of potentials and constraints (Bach et al. 2017) which is at most linear to the number of cells in our problem.

4.1 Cell Classifier

The cell classifier operates in two processes. In training phase, a statistical learning model learns to predict a *candidate data type* using cell embeddings. In inference phase, a PSL model enforces constraints between the data types.

To model this task, we first identify several simple features based on cell content, such as `IsNumber`, `IsDate` and `IsEmpty`, indicating if the content could be parsed as a number, a date/time, or other special cases such as empty cells or reserved values such as "n/a", respectively. We then extract dependencies between the features and data types. The PSL model consists of a set of rules presenting such dependencies. In our model, we are interested in predicting the cell data types (`DataType`). Some example rules are:

$$\texttt{IsDate}(C) \Rightarrow \texttt{DataType}(C, \text{“}datetime\text{”})$$
$$\texttt{CELabel}(C, T) \Rightarrow \texttt{DataType}(C, T)$$

The first rule expresses: if the cell content C is successfully parsed as a datetime, we could confidently label it as "Datetime". `CELabel` shows the candidate data type predicted by the statistical learning model.

The above implication rules are introduced to avoid cell embeddings being overfitted and making mistakes in obvious situations. However, the above rules are not always sufficient to differentiate data types from each other, especially for numbers. For example, "2020" could be a number indicating a quantity or a specific year. To alleviate this issue, we also introduce several conjunctive rules. For example:

$$\texttt{IsNum}(C) \wedge !\,\texttt{IsInt}(C) \Rightarrow \texttt{DataType}(C, \text{“}cardinal''\text{”})$$
$$\texttt{HasAlpha}(C) \wedge !\,\texttt{HasNum}(C) \Rightarrow !\,\texttt{DataType}(C, \text{“}cardinal''\text{”})$$
$$\texttt{OneWord}(C) \wedge \texttt{HasNum}(C) \Rightarrow !\,\texttt{DataType}(C, \text{“}person''\text{”})$$

These rules are derived from constraints upon how humans express different types of cell content. The first two rules demonstrate how numbers exist in tables. Usually, floating (not integer) numbers are only used to show quantities, such as average numbers of people and average scores of classes, which are cardinals. In addition, cardinals always contain some numeric characters. The third rule shows a person's name usually is not a single token with numeric values. We note that the above rules are only a subset of rules used in the system. We provide the full list of rules in Appendix.

4.2 Block Detector

The block detector takes the cell data types and cell embeddings as inputs to identify blocks, and assign each block a functional label. This component operates in three steps: generating candidate blocks, enforcing probabilistic constraints, and coalescing blocks.

Generating Candidate Blocks Different from the region-based approach (Koci et al. 2019) that groups adjacent cells into rectangular regions, we introduce a top-down approach. It starts from a whole table and recursively splits the table into smaller blocks. This idea is inspired by the Bayesian CART model which constructs a decision tree by recursively partitioning a space into subsets (Chipman, George, and McCulloch 1998). Figure 3 demonstrates such a decision tree.

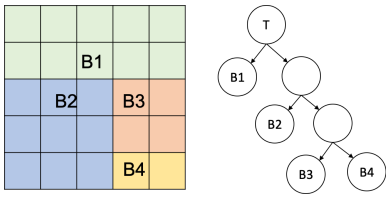


Figure 3: An illustration of block generation. T is the original table, $B1$, $B2$, $B3$ and $B4$ are non-overlapping blocks in the table (left) and leaf nodes in the decision tree (right).

Instead of constructing the tree greedily, Chipman et al. introduced a Markov Chain Monte-Carlo (MCMC) approach to finding the tree. Formally, we start from the table T, at each step i, for each active node B_{ij}, we choose to either split this node into two children nodes or stop at this node with the *splitting probability* p_{split}. If a node is chosen to be split, we select the row/column to split following the *rule distribution* p_{rule}, otherwise we stop splitting this node and it becomes a leaf node. For example, in fig. 3, T is split into B_1 and another node. B_1 is not split so it becomes a leaf node while the other node is further split into two extra nodes. Each leaf node becomes a candidate block in our problem. Following this process, the system generates N such trees and randomly select one tree, with the weight function W_{ent}, to finalize candidate blocks.

Algorithm 1 shows the details. We call the function `SampleATree` to generate candidate blocks. Similar to the Bayesian CART model, we set the splitting probability to be $p_{split}(d) = \frac{1}{(1+d)^\beta}$ where β is a hyperparameter and d is the depth of the node. A larger β indicates a smaller tree.

We make use of the cell data types provided by the cell classifier to decide the rule distribution. At node B_i, suppose it can be split into two children nodes B_{i1} and B_{i2}, the data type distributions of B_{i1} and B_{i2} are $\mathbf{D_{i1}} = \left[d_1^1, d_2^1, \cdots, d_k^1\right]$ and $\mathbf{D_{i2}} = \left[d_1^2, d_2^2, \cdots, d_k^2\right]$ where k is the number of data types. We set the rule distribution to be $p_{rule}(B_{i1}, B_{i2}) = \lambda \cdot e^{-\frac{\lambda}{|\mathbf{D_{i1}} - \mathbf{D_{i2}}|^2}}$ where λ is a hyperparameter. This is designed based on the assumption that a split which makes the distributions more diverse is more likely to be chosen. We apply the exponential family $\lambda \cdot e^{-\lambda}$ to make the difference between the distributions more significant.

We use the *entropy* W_{ent} to determine the weight of a tree. $W_{ent}(B) = \lambda \cdot e^{-\frac{\lambda}{\sum_{b:B}\left(-\frac{|b|}{\sum_{b':B}|b'|}\sum_t p_b^t \log p_b^t\right)}}$ where B is a set of blocks , b is a block, $|b|$ is the size (area) of b, and p_b^t is the ratio of the cells with the data type t in b. The λ here is the same as the λ in p_{rule}.

Enforcing Probabilistic Constraints In the second step, we assign a functional label to each candidate block using a PSL model. Same as the cell classifier, we perform this step with two components: a statistical learning model that takes a cell embedding as an input to predict a functional label for each cell, and a PSL model that enforces probabilistic constraints. A set of example rules are listed below:

Algorithm 1: Candidate Block Generation

Function `Split` ($block$) **:**
  $queue \longleftarrow \{(block, 0)\}$; $blocks \longleftarrow \{\}$;
  while $queue \neq \emptyset$ **do**
    $(\langle t, b, l, r\rangle, d) \longleftarrow queue.get()$
    `// t, b, l, and r are indices.`
    Randomly select a number v within $[0, 1]$
    if $v < p_{split}(d)$ **then**
      Randomly split $\langle t, b, l, r\rangle$ into B_1 and B_2 using p_{rule}.
      $queue.push((B_1, d+1))$
      $queue.push((B_2, d+1))$
    else
      $blocks.add(\langle t, b, l, r\rangle)$
  return $blocks$;
Function `GenerateATree` (T, $types$) **:**
  row_blocks = `Split`(T); `// Row-wise`
  $blocks \longleftarrow \emptyset$;
  foreach B *in* row_blocks **do**
    $blocks.union$(`Split`(B)) `// Column-wise`
  return $blocks$;
Function `SampleATree` (T, $types$, N) **:**
  $trees \longleftarrow \emptyset$
  foreach $1 \leq i \leq N$ **do**
    $trees.add$(`GenerateATree`($T, types$))
  return Sample a tree from $trees$ using W_{ent}.

$$\texttt{CELabel}(B, L) \Rightarrow \texttt{BT}(B, L)$$
$$\texttt{FirstRow}(B) \Rightarrow \texttt{BT}(B, \text{"}header\text{"})$$

$\texttt{BT}(B, L)$ indicates the possibility that block B is assigned label L. Same as the cell classifier, `CELabel` shows how the block detector uses cell embeddings. The statistical learning model assigns each cell a label. Since a block may have more than one cell, $\texttt{CELabel}(B, L)$ reveals the possibility that block B is assigned label L. This corresponds to the ratio of the number of cells assigned label L over the total number of cells in B. The second rule expresses: blocks on the first row usually is a *Header* block.

In addition to the simple rules listed above, we also apply several conjunctive rules that fuse the inherent positional constraints within the table layouts. For example:

$$\texttt{SameRow}(B_1, B_2) \wedge \texttt{BT}(B_1, \text{"}header\text{"}) \Rightarrow \texttt{BT}(B_2, \text{"}header\text{"})$$
$$\texttt{Abv}(B_1, B_2) \wedge \texttt{Abv}(B_2, B_3) \wedge \texttt{BT}(B_1, C) \wedge \texttt{BT}(B_3, C) \Rightarrow \texttt{BT}(B_2, C)$$

The first rule indicates that if a block is a *Header* block, blocks that on the same row as this block are also *Header* blocks. The last rule takes neighboring blocks into consideration. If the neighbor above and the below it have the same label, it should also be assigned this label. These conjunctive constraints exploit the power of collective classification that probabilistic models perform.

Block Coalescing After assigning labels to the candidate blocks, we apply a post-processing step to merge small blocks into large blocks. In this step, we first merge neighboring blocks if they have the same top row, bottom row and labels. Similarly, we then merge neighboring blocks if they have the same left column, right column and labels. The block detector finally passes the merged blocks to the layout predictor. We design this step to resolve the issue of over-partitioning and produce better blocks.

4.3 Layout Predictor

The layout predictor is the last component of our system. It predicts a relationship between each pair of blocks identified by the block detector. We model the task as a PSL problem utilizing relative positional relationships between the blocks. A set of example rules are listed below:

$$\texttt{Adj}(B_1, B_2) \wedge \texttt{BT}(B_1, \text{"}data\text{"}) \wedge \texttt{BT}(B_2, \text{"}data\text{"}) \Rightarrow \texttt{Rel}(B_1, B_2, \text{"}empty\text{"})$$

$$\texttt{Hrz}(B_1, B_2) \wedge \texttt{BT}(B_1, \text{"}attr\text{"}) \wedge \texttt{BT}(B_2, \text{"}data\text{"}) \Rightarrow \texttt{Rel}(B_1, B_2, \text{"}attribute\text{"})$$

These rules illustrate our hypotheses about positional relationships between blocks. If two data blocks are neighbors, they might not have any special relationship. If two blocks are horizontally aligned, one is an attribute block and the other one is a data block, the attribute block might reveal attributes of the data within the data block.

5 Datasets for Table Understanding

5.1 Existing Datasets

There are three main datasets DeEx, CIUS and SAUS designed for evaluating cell functional type classification. The DeEx dataset was collected in the DeExcelerator project (Eberius et al. 2013) and contains 457 annotated sheets. The CIUS dataset was originally from the Crime In the US (CIUS) database and contains 268 sheets. The SAUS dataset was downloaded from the U.S. Census Burea (Chen and Cafarella 2014) and contains 210 sheets. Both the CIUS and SAUS datasets were annotated by Ghasemi-Gol, Pujara, and Szekely (2019). We evaluate our approach on all these three datasets for the task of block detection. These three datasets are cell-level annotations based on functional roles.

5.2 A New Dataset from Data.gov

The existing datasets, designed for evaluating cell functional type classification, have relatively narrow domains and focus only on Excel files. To evaluate the three aforementioned table understanding tasks, we introduce a new dataset DG. We downloaded 1837 files from the U.S. Open Data website (data.gov). These files are from different topic domains such as agriculture, climate, ocean and ecosystem, and are in different formats (i.e. csv and Excel). We sampled 431 tables from these files and annotated them for the three table understanding tasks. To show inter-annotator agreements, we ask two annotators to independently annotate 25 tables and evaluate the Cohen's kappa coefficients (McHugh 2012) for three tasks. The results are 0.937 for cell classification, 0.960 for block deteection, and 0.936 for layout prediction, which indicate the good reliability of the annotations. Specifically, for block detection, we align the blocks from annotator A with those from annotator B and then compare the labels.

6 Evaluation

In this section, we present the experimental evaluation of the proposed system based on the four datasets. In the process of creating PSL rules, we randomly selected 10 tables from each dataset as a rule development set. These tables are not included in any training and testing. In all experiments, we perform 5-fold cross validation on the rest of the tables: for each dataset, we randomly split the tables into 5 folds, train/validate a model using 4 folds and test on 1 fold. For the 4 folds, we randomly split the tables with 9:1 ratio into training and validation sets. The results are averaged on the 5 test folds. Each cell embedding consists of 40-dimensional stylistic features and 512-dimensional contextual embedding.

6.1 Cell Classification

As described in Section 3.1, we evaluate our cell classifier on the DG dataset. Each cell is classified into 1 of the 9 data types: empty (**Emp**), cardinal (**Card**), Nominal (**Nom**), ordinal (**Ord**), datetime (**Date**), location (**Loc**), organization (**Org**), person (**Per**), and other string (**Str**). In the PSL model, we leverage 32 logical rules.

We compare our system with the following baselines: 1) **Random Forest** (RF): We use the RandomForest classifier in scikit-learn library (Buitinck et al. 2013). It takes cell embeddings as input the predict a cell data type. We select n_estimator among $[100, 300]$, max_depth among $[5, 50, None]$, min_sample_split among $[2, 10]$ and min_samples_leaf among $[1, 10]$. We use the bootstrap mode with balanced sub-sampling. 2) **Conditional Random Field** (CRF): CRFs are a type of PGMs, which takes the context (neighboring cells in tabular data) into consideration. In this experiment, it uses a feature set introduced in in (Chen and Cafarella 2013) to make predictions. We choose 2-dimensional CRF to represent row-wise and column-wise neighborhood interactions. We use GridCRF class from the pystruct library (Müller and Behnke 2014). We set the max_iter to be 500, tolerance to be 0.01 and select c_range among $[0.01, 0.1, 1.0]$. 3) **Multi-layer Perceptron** (MLP) We use the pytorch library (Paszke et al. 2019) to create a two-layer neural network with the Rectified Linear activation function (ReLU). It also takes cell embeddings as input. We set batch size to be 32, learning rate to be 0.0001, and epoch to be 50. We use cross entropy loss.

Results: Table 1 shows the results of this experiment. For both RF and MLP, the PSL model improves over their results. The results demonstrate that the logical rules are able to provide useful high-level constraints between data types and cell-level features. Compared to CRF, it is more flexible to enforce explicit constraints in PSL. For each table, the average running time of PSL is 10 seconds.

Table 1: Data type classification results (F1 scores) on DG. Avg is the macro F1 score (%) ± the standard deviation.

	Emp	Crd	Str	Dat	Loc	Org	Ord	Nom	Per	Avg
CRF	81.9	82.5	42.4	56.2	34.4	16.8	0.0	36.0	1.3	39.1±1.8
MLP	84.5	85.6	69.1	59.3	54.9	46.8	0.0	52.0	1.2	50.4±5.4
RF	85.0	84.4	73.2	61.4	65.2	**55.5**	**0.3**	**53.4**	**39.3**	57.5±4.7
PSL (MLP)	96.5	**88.3**	70.2	77.8	55.8	43.3	**0.3**	52.4	1.0	54.0±3.1
PSL (RF)	**96.8**	87.8	**74.3**	**78.4**	**66.1**	52.5	0.2	53.0	31.7	**60.1±3.2**

6.2 Block Detection

We use 19 rules in the PSL model with the same weight. We conduct two experiments. First, since the existing datasets are for cell role type classification, however, our system identifies block role type. To compare with baseline models on these datasets, we assign each cell a functional role according to the predictions for blocks. Second, we measure how precisely predicted blocks are aligned with ground-truth blocks. We borrow the idea of Error-of-Boundary (EoB) proposed in Dong et al. (2019):

$$EoB(B^{gt}, B^{p}) = \max\left(|B_t^{gt} - B_t^{p}|, |B_b^{gt} - B_b^{p}|, |B_l^{gt} - B_l^{p}|, |B_r^{gt} - B_r^{p}|\right)$$

where B_t^{gt}, B_b^{gt}, B_l^{gt}, and B_r^{gt} are the top row, bottom row, left column, and right column of the ground-truth block B^{gt}. The annotations are same for the predicted block B^{p}. Since a table may have several blocks, instead of evaluating a single block, we use averaged EoB over all blocks:

$$EoB_{avg} = \sum_{1 \le i \le N, 1 \le j \le M} \frac{1}{|B^{gt^{ij}} \cap B^{p^{ij}}|} EoB(B^{gt^{ij}}, B^{p^{ij}})$$

where N is the number of rows, M is the number of columns, $B^{gt^{ij}}$ is the block that the cell on the i^{th} row and j^{th} column belongs to. Same for $B^{p^{ij}}$.

In the system, we set the number of sample trees to be 50, select the α among $[0.01, 0.05,]$ and λ among $[5, 10]$. Following (Ghasemi-Gol, Pujara, and Szekely 2019), we compare the system with the following baselines: 1) **Random Forest** (RF): In the first experiment, it takes cell embeddings as input to predict a role type for each cell. We also use the RandomForest class from scikit-learn. We select n_estimator among $[100, 300]$, max_depth among $[5, 50, None]$, min_sample_split among $[2, 10]$ and min_samples_leaf among $[1, 10]$. We use the bootstrap mode with balanced sub-sampling. 2) **Conditional Random Field** (CRF): We follow the instructions in (Chen and Cafarella 2013) to implement the linear CRF. We use the ChainCRF class from the pystruct library. We set max_iter to be 1000, tol to be 0.01, and select C_range from [0.1, 0.3, 0.5, 0.7, 1.0]. We use the same feature set as is used in the CRF model in the previous experiment. The CRF takes the inter-dependencies between rows and predict the functional type (only metadata, data and header) of each row. For data rows, cells with actual numbers are classified as data and other cells are attributes. 3) **Recurrent Neural Network** (RNN):

Table 2: Results of block detection models.

(a) Cell functional type classification results. The annotations are same as those in table 1. All models use the outputs of the PSL (RF) cell classifier.

		MD	DT	HD	AT	Avg
CIUS	CRF	96.5	67.6	94.9	36.8	73.9±8.9
	RNN	**99.5**	99.3	97.4	90.5	96.7±4.1
	RF	95.9	**99.7**	88.9	97.0	95.4±0.6
	PSL(RNN)	94.8	99.2	**97.8**	89.3	95.3±4.1
	PSL(RF)	93.6	**99.7**	96.0	**97.6**	**96.7±1.1**
SAUS	CRF	80.7	82.2	**95.7**	38.2	74.2±5.8
	RNN	**94.3**	97.5	84.1	79.5	88.9±2.3
	RF	79.1	98.6	78.8	91.1	86.9±4.0
	PSL(RNN)	87.6	97.8	86.7	79.5	87.9±1.4
	PSL(RF)	80.6	**99.0**	85.4	**92.8**	**89.4±2.5**
DeEx	CRF	35.6	55.7	48.0	1.7	35.3±6.9
	RNN	33.8	96.1	47.2	39.5	54.2±5.9
	RF	53.4	98.4	51.0	26.5	57.3±2.0
	PSL(RNN)	38.5	97.2	53.5	**44.9**	58.5±8.0
	PSL(RF)	**65.4**	**98.8**	**60.5**	26.0	**62.7±3.9**
DG	CRF	41.3	53.1	**94.1**	34.8	55.9±9.3
	RNN	45.4	**95.9**	82.9	**78.8**	75.8±4.3
	RF	74.0	95.8	80.7	77.8	82.1±2.5
	PSL(RNN)	69.9	95.7	89.2	77.4	83.1±5.2
	PSL(RF)	**77.2**	95.7	91.4	77.4	**85.4±4.8**

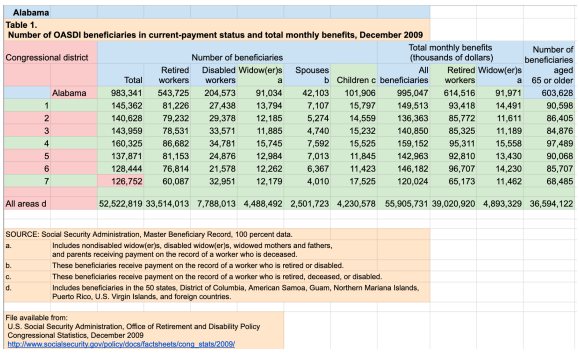
(b) Average EoB scores of all models on the DG dataset.

Method	CRF	RNN	RF	PSL (RNN)	PSL (RF)
EoB_{avg}	5179	24192	59403	2828	**1995**

It is the classification model proposed in (Ghasemi-Gol, Pujara, and Szekely 2019). We set the epoch to be 50 and learning rate to be 0.0001. In the second experiment, we use the region-based approach from (Koci et al. 2019) to create blocks: it builds row-label intervals (RLI) (i.e. neighboring cells with the same label on the same row), and then merge RLIs in adjacent rows into regions.

Results Table 2a shows the results of the first experiment. The PSL model improves the performance over baseline classifiers (i.e. RF and RNN) using cell embeddings. The reasons are: 1) the rectangular block representations guarantee the cells within a block have the same type, and 2) the explicit constraints between positional features and the functional types, and between the functional types themselves further assist the performance. The major challenge that PSL leads to lower accuracy in some cases (such as MD in DeEx and AT in DG) is that the method uses data type distributions to decide the division of a block. If two adjacent rows/columns have very similar data type distributions, they are less likely to be split.

Table 2b presents the results of the second experiment. In terms of the average EoB, our model shows better results. We present the results of the example table of Figure 1 in



Alabama

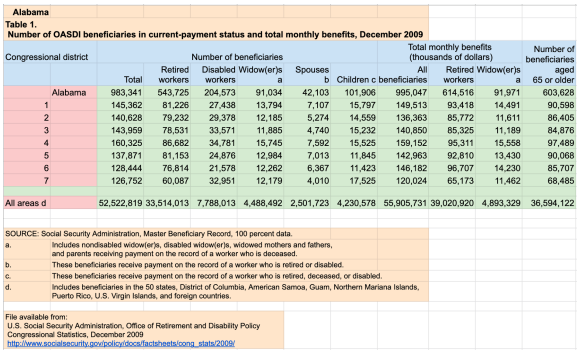
Table 1.
Number of OASDI beneficiaries in current-payment status and total monthly benefits, December 2009

Congressional district		Number of beneficiaries						Total monthly benefits (thousands of dollars)			Number of beneficiaries
		Total	Retired workers	Disabled workers	Widow(er)s a	Spouses b	Children c	All beneficiaries	Retired workers	Widow(er)s a	aged 65 or older
	Alabama	983,341	543,725	204,573	91,034	42,103	101,906	995,047	614,516	91,971	603,628
1		145,362	81,226	27,438	13,794	7,107	15,797	149,513	93,418	14,491	90,598
2		140,628	79,232	29,378	12,185	5,274	14,559	136,363	85,772	11,611	86,405
3		143,959	78,531	33,571	11,885	4,740	15,232	140,850	85,325	11,189	84,876
4		160,325	86,682	34,781	15,745	7,592	15,525	159,152	95,311	15,558	97,489
5		137,871	81,153	24,876	12,984	7,013	11,845	142,963	92,810	13,430	90,068
6		128,444	76,814	21,578	12,262	6,367	11,423	146,182	96,707	14,230	85,707
7		126,752	60,087	32,951	12,179	4,010	17,525	120,024	65,173	11,462	68,485
All areas d		52,522,819	33,514,013	7,788,013	4,488,492	2,501,723	4,230,578	55,905,731	39,020,920	4,893,329	36,594,122

SOURCE: Social Security Administration, Master Beneficiary Record, 100 percent data.
a. Includes nondisabled widow(er)s, disabled widow(er)s, widowed mothers and fathers, and parents receiving payment on the record of a worker who is deceased.
b. These beneficiaries receive payment on the record of a worker who is retired or disabled.
c. These beneficiaries receive payment on the record of a worker who is retired, deceased, or disabled.
d. Includes beneficiaries in the 50 states, District of Columbia, American Samoa, Guam, Northern Mariana Islands, Puerto Rico, U.S. Virgin Islands, and foreign countries.

File available from:
U.S. Social Security Administration, Office of Retirement and Disability Policy
Congressional Statistics, December 2009
http://www.socialsecurity.gov/policy/docs/factsheets/cong_stats/2009/

(a) The result of the RF model (using cell embeddings only).



Alabama

Table 1.
Number of OASDI beneficiaries in current-payment status and total monthly benefits, December 2009

Congressional district		Number of beneficiaries						Total monthly benefits (thousands of dollars)			Number of beneficiaries
		Total	Retired workers	Disabled workers	Widow(er)s a	Spouses b	Children c	All beneficiaries	Retired workers	Widow(er)s a	aged 65 or older
	Alabama	983,341	543,725	204,573	91,034	42,103	101,906	995,047	614,516	91,971	603,628
1		145,362	81,226	27,438	13,794	7,107	15,797	149,513	93,418	14,491	90,598
2		140,628	79,232	29,378	12,185	5,274	14,559	136,363	85,772	11,611	86,405
3		143,959	78,531	33,571	11,885	4,740	15,232	140,850	85,325	11,189	84,876
4		160,325	86,682	34,781	15,745	7,592	15,525	159,152	95,311	15,558	97,489
5		137,871	81,153	24,876	12,984	7,013	11,845	142,963	92,810	13,430	90,068
6		128,444	76,814	21,578	12,262	6,367	11,423	146,182	96,707	14,230	85,707
7		126,752	60,087	32,951	12,179	4,010	17,525	120,024	65,173	11,462	68,485
All areas d		52,522,819	33,514,013	7,788,013	4,488,492	2,501,723	4,230,578	55,905,731	39,020,920	4,893,329	36,594,122

SOURCE: Social Security Administration, Master Beneficiary Record, 100 percent data.
a. Includes nondisabled widow(er)s, disabled widow(er)s, widowed mothers and fathers, and parents receiving payment on the record of a worker who is deceased.
b. These beneficiaries receive payment on the record of a worker who is retired or disabled.
c. These beneficiaries receive payment on the record of a worker who is retired, deceased, or disabled.
d. Includes beneficiaries in the 50 states, District of Columbia, American Samoa, Guam, Northern Mariana Islands, Puerto Rico, U.S. Virgin Islands, and foreign countries.

File available from:
U.S. Social Security Administration, Office of Retirement and Disability Policy
Congressional Statistics, December 2009
http://www.socialsecurity.gov/policy/docs/factsheets/cong_stats/2009/

(b) The result of the PSL model.

Figure 4: The block detection results of the table in Figure 1.

Figure 4 to demonstrate the reason. The RF (similar for CRF and RNN) model depends only on the cell classification results so that the misclassified cells scattered over the whole table make the generated blocks small and affect the EoB. For a table in CIUS, SAUS, DeEx, and DG, the average running times of PSL are 39, 3, 17, and 7 seconds, respectively.

6.3 Layout Prediction

We use 15 logical rules in the system. We evaluate the performance of the layout predictor on the DG dataset. We use 5 aforementioned relationships: empty (**EP**), header of (**HO**), attribute of (**AO**), global attribute (**GA**) and supercategory of (**SC**). We compare the predictor with the following two baselines: 1) **Random Forest** (RF): For every two blocks, we use a few manually crafted features: their functional labels (predicted by the block detector), and several relationships between blocks (*below*, *above*, *left right*, *adjacent*, and *overlap*). We use the RandomForest from the scikit-learn library. We select n_estimators among [100, 300], max_depth among [5, 50, None], min_samples_split among [2, 10] and min_samples_leaf among [1, 10]. 2) **Conditional Random Field** (CRF) We construct a graph CRF for this task. If we treat each block as a node and the relationship between two blocks as an edge, the above features are features associated with edges. We use the EdgeFeatureGraphCRF from the pystruct library. We set max_iter to be 50, tol to be 0.01, and select C_range among [0.01, 0.1],

Results We use the blocks generated by the block detector to run different layout prediction methods. For each block b, we match it with the ground-truth block b' which shares the most overlapping cells with b. All predicted relationships for b are added to b' and compared with the ground-truth relationships. Table 3 shows the results. The PSL model performs better than the compared models in most cases. For each table, the average running time of PSL is 0.4 seconds.

Table 3: Layout prediction results on the DG dataset. All models use the outputs of the PSL (F) block detector.

Method	EP	HO	AO	GA	SC	Avg
RF	81.7	1.1	2.1	22.7	0.0	21.5±1.2
CRF	88.5	33.7	32.2	40.0	0.0	38.9±3.1
PSL	**89.6**	**70.3**	**32.8**	**43.0**	**25.6**	**52.3±3.4**

7 Related Work

In recent years, a large amount of research efforts seek to solve different tasks for automated processing tabular data, such as table detection, table classification, and data transformation. Wang and Hu (2002); Wang et al. (2012); Fang et al. (2012) focus on extracting tables from HTML pages, document images and PDFs, and Dong et al. (2019) leverages the convolutional neural networks (CNN) to develop an end-to-end framework for spreadsheet table detection. Cafarella et al. (2008b), Crestan and Pantel (2010) and Eberius et al. (2015) introduced content, lexical, global and local features, and Nishida et al. (2017) proposed a hybrid architecture TabNet using recurrent neural networks and CNN to encode tabular data, to perform table type classification. The task of tabular data transformation is transform tabular data into more formal tables, i.e. relational databases. Su et al. (2017); Shigarov et al. (2016); Dou et al. (2018) used either engineered or automatically inferred features to construct rule-based engines.

8 Conclusions and Future Work

In this paper, we proposed an end-to-end system for solving three table understanding tasks, i.e. cell classification, block detection and layout prediction. Our system exploits the rich information within cell embeddings as well as logical constraints following the principles of the layouts of the tabular data. We introduced probabilistic rules and used the probabilistic inference framework PSL to enforce the rules. We experimentally evaluated our system and showed the results. The output of our system can potentially be used for solving several downstream tasks such as semantic modeling, table summarization, and question answering on tables. For example, in semantic modeling, each identified block can be treated as an attribute and the relationships between blocks are helpful for predicting better semantic models. In the current system, we use the distributions of cell data types to generate candidate blocks. Alternative measures and sampling strategies could also be leveraged. In addition, the idea of combining embeddings and probabilistic constraints could potentially be used for solving the layout prediction and other tasks related to tables.

9 Acknowledgements

We thank Majid Ghasemi Gol for his kind help at the beginning of the project, and Muhao Chen for his excellent feedback and advice for the paper. We also thank anonymous reviewers for their valuable comments and suggestions. This research is supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under contracts FA8650-17-C-7715 and FA8750-17-C-0106.

References

Atlantic, T. 2020. Covid Tracking Data. https://covidtracking.com/data/national/cases.

Bach, S. H.; Broecheler, M.; Huang, B.; and Getoor, L. 2017. Hinge-Loss Markov Random Fields and Probabilistic Soft Logic 18(1): 3846–3912. ISSN 1532-4435.

Bank, T. W. 2020. Global economic perspectives. https://www.worldbank.org/en/publication/global-economic-prospects.

Buitinck, L.; Louppe, G.; Blondel, M.; Pedregosa, F.; Mueller, A.; Grisel, O.; Niculae, V.; Prettenhofer, P.; Gramfort, A.; Grobler, J.; Layton, R.; VanderPlas, J.; Joly, A.; Holt, B.; and Varoquaux, G. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 108–122.

Cafarella, M. J.; Halevy, A.; Wang, D. Z.; Wu, E.; and Zhang, Y. 2008a. WebTables: Exploring the Power of Tables on the Web. *Proc. VLDB Endow.* 1(1): 538–549. ISSN 2150-8097. doi: 10.14778/1453856.1453916.

Cafarella, M. J.; Halevy, A.; Wang, D. Z.; Wu, E.; and Zhang, Y. 2008b. WebTables: Exploring the Power of Tables on the Web. *Proc. VLDB Endow.* 1(1): 538–549. ISSN 2150-8097. doi: 10.14778/1453856.1453916.

Chen, X.; Chen, M.; Shi, W.; Sun, Y.; and Zaniolo, C. 2019. Embedding Uncertain Knowledge Graph. In *AAAI*.

Chen, Z.; and Cafarella, M. 2013. Automatic Web Spreadsheet Data Extraction. In *Proceedings of the 3rd International Workshop on Semantic Search Over the Web*. ISBN 9781450324830.

Chen, Z.; and Cafarella, M. 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*, 1126–1135. ISBN 9781450329569.

Chipman, H. A.; George, E. I.; and McCulloch, R. E. 1998. Bayesian CART Model Search. *Journal of the American Statistical Association* 93(443): 935–948. doi:10.1080/01621459.1998.10473750.

CIUS. 2019. CIUS. https://ucr.fbi.gov/crime-in-the-u.s.

Crestan, E.; and Pantel, P. 2010. A Fine-Grained Taxonomy of Tables on the Web. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, 1405–1408. ISBN 9781450300995. doi:10.1145/1871437.1871633.

Data.gov. 2019. Data.gov. https://www.data.gov/.

DeEx. 2013. DeExcelarator. https://wwwdb.inf.tu-dresden.de/research-projects/deexcelarator/.

Division, U. N. S. 2020. UNdata. http://data.un.org/.

Dong, H.; Liu, S.; Han, S.; Fu, Z.; and Zhang, D. 2019. TableSense: Spreadsheet Table Detection with Convolutional Neural Networks. In *AAAI*.

Dou, W.; Han, S.; Xu, L.; Zhang, D.; and Wei, J. 2018. Expandable Group Identification in Spreadsheets. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, 498–508. ISBN 9781450359375. doi: 10.1145/3238147.3238222.

Eberius, J.; Braunschweig, K.; Hentsch, M.; Thiele, M.; Ahmadov, A.; and Lehner, W. 2015. Building the Dresden Web Table Corpus: A Classification Approach. In *2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)*, 41–50.

Eberius, J.; Werner, C.; Thiele, M.; Braunschweig, K.; Dannecker, L.; and Lehner, W. 2013. DeExcelerator: a framework for extracting relational data from partially structured documents. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2477–2480.

Fang, J.; Mitra, P.; Tang, Z.; and Giles, C. L. 2012. Table Header Detection and Classification. In *AAAI*.

Ghasemi-Gol, M.; Pujara, J.; and Szekely, P. 2019. Tabular Cell Classification Using Pre-Trained Cell Embeddings. In *International Conference on Data Mining*.

Koci, E.; Thiele, M.; Romero, O.; and Lehner, W. 2016. A Machine Learning Approach for Layout Inference in Spreadsheets. In *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, 77–88. ISBN 9789897582035.

Koci, E.; Thiele, M.; Romero, O.; and Lehner, W. 2019. Cell Classification for Layout Recognition in Spreadsheets. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, 78–100. Cham: Springer International Publishing.

Kouki, P.; Fakhraei, S.; Foulds, J.; Eirinaki, M.; and Getoor, L. 2015. HyPER: A Flexible and Extensible Probabilistic Framework for Hybrid Recommender Systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*, 99–106. ISBN 9781450336925. doi:10.1145/2792838.2800175.

Kouki, P.; Pujara, J.; Marcum, C.; Koehly, L.; and Getoor, L. 2017. Collective Entity Resolution in Familial Networks. In *IEEE International Conference on Data Mining*.

McHugh, M. 2012. Interrater reliability: The kappa statistic. *Biochemia medica : časopis Hrvatskoga društva medicinskih biokemičara / HDMB* 22: 276–82. doi:10.11613/BM.2012.031.

Müller, A. C.; and Behnke, S. 2014. PyStruct: Learning Structured Prediction in Python. *J. Mach. Learn. Res.* 15(1): 2055–2060. ISSN 1532-4435.

Nishida, K.; Sadamitsu, K.; Higashinaka, R.; and Matsuo, Y. 2017. Understanding the Semantic Structures of Tables with a Hybrid Deep Neural Network Architecture. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, 168–174.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, 8024–8035.

Pujara, J.; Rajendran, A.; Ghasemi-Gol, M.; and Szekely, P. 2019. A Common Framework for Developing Table Understanding Models. In *International Semantic Web Conference - Posters*.

Rümmele, N.; Tyshetskiy, Y.; and Collins, A. 2018. Evaluating approaches for supervised semantic labeling. *CoRR* abs/1801.09788.

SAUS. 2014. SAUS. http://dbgroup.eecs.umich.edu/project/sheets/datasets.htm.

Shigarov, A. O.; Paramonov, V. V.; Belykh, P. V.; and Bondarev, A. I. 2016. Rule-Based Canonicalization of Arbitrary Tables in Spreadsheets. In Dregvaite, G.; and Damasevicius, R., eds., *Information and Software Technologies*, 78–91. Cham: Springer International Publishing. ISBN 978-3-319-46254-7.

Su, H.; Li, Y.; Wang, X.; Hao, G.; Lai, Y.; and Wang, W. 2017. Transforming a Nonstandard Table into Formalized Tables. 311–316. doi:10.1109/WISA.2017.38.

Wang, J.; Wang, H.; Wang, Z.; and Zhu, K. Q. 2012. Understanding Tables on the Web. In Atzeni, P.; Cheung, D.; and Ram, S., eds., *Conceptual Modeling*, 141–155. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-34002-4.

Wang, X. 1996. Tabular Extraction, Editing, and Formatting.

Wang, Y.; and Hu, J. 2002. A Machine Learning Based Approach for Table Detection on the Web. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, 242–250. ISBN 1581134495. doi:10.1145/511446.511478.

Zhou, M.; Tao, W.; Ji, P.; Shi, H.; and Zhang, D. 2020. Table2Analysis: Modeling and Recommendation of Common Analysis Patterns for Multi-Dimensional Data. In *AAAI*.