ReS^{2}TIM: Reconstruct Syntactic Structures from Table Images

Wenyuan Xue*, Qingyong Li*, Dacheng Tao†
*Beijing Key Lab of Transportation Data Analysis and Mining, Beijing Jiaotong University, China
†UBTECH Sydney AI Centre, School of Computer Science, Faculty of Engineering, The University of Sydney, Australia
{wyxue17, liqy}@bjtu.edu.cn, dacheng.tao@sydney.edu.au

***Abstract*—Tables often represent densely packed but structured data. Understanding table semantics is vital for effective information retrieval and data mining. Unlike web tables, whose semantics are readable directly from markup language and contents, the full analysis of tables published as images requires the conversion of discrete data into structured information. This paper presents a novel framework to convert a table image into its syntactic representation through the relationships between its cells. In order to reconstruct the syntactic structures of a table, we build a cell relationship network to predict the neighbors of each cell in four directions. During the training stage, a distance-based sample weight is proposed to handle the class imbalance problem. According to the detected relationships, the table is represented by a weighted graph that is then employed to infer the basic syntactic table structure. Experimental evaluation of the proposed framework using two datasets demonstrates the effectiveness of our model for cell relationship detection and table structure inference.**

***Keywords*-layout analysis, table recognition, visual relationship**

I. Introduction

Data in a table is usually arranged in rows and columns or even more complex configurations. Tables are widely used for data presentation and analysis in all walks of life, appearing in electronic documents, print media, handwritten notes, web pages, computer software, traffic signs, and many other places. It has recently been shown that understanding table semantics on the web enhances the quality and efficiency of web searching [1]–[3]. However, in contrast to web tables, which are extracted from HTML with syntactic information, the syntactic representation must first be analyzed when using tables from images (Fig. 1). The basic syntactic representation of a table contains the number of rows and columns, and the coordinates of each table cell. Then the syntactic information, combined with Optical Character Recognition (OCR) results, is integrated as semantic information that can be utilized for downstream tasks such as data mining and question answering. The focus of this study is the first stage: *the syntactic representation of table images*.

High degree of intra-class variability is one of the most challenging problems when analyzing the syntactic representation of table images. Tables are drawn in a great variety of ways, including different layouts and the erratic use of borders. Further, the diverse table contents can determine

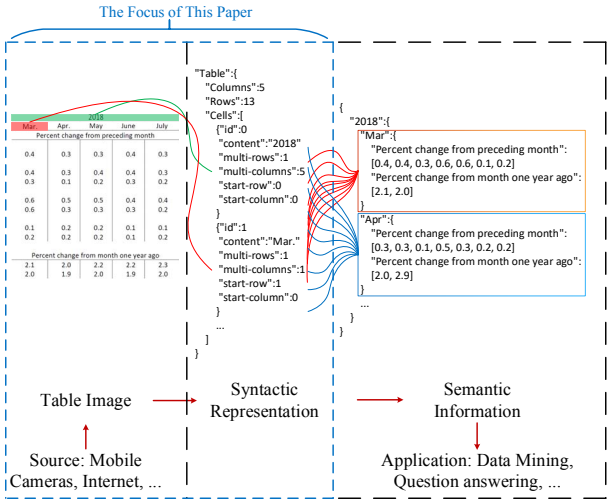


Figure 1: The focus of this paper. Before serving downstream tasks, table images from the internet or mobile devices are converted to syntactic representation and then integrated as semantic information.

different table structures, making it hard to define what a table looks like or how the cells are organized.

There are two ways to depict the syntactic structure of tables. The first is to classify table cells into specific categories [4], [5], e.g., '*column header*', '*row header*', and '*data*', and then formalize the syntactic table structure analysis as a classification problem. However, this method fails to represent complex structures such as embedded headers. The other approach is to utilize the visual features to distinguish the row and column header hierarchy [6]–[8]. Most of these methods rely on some format information (e.g., indentation, boldness, and capitalization) directly extracted from PDF or HTML files. However, it is more difficult to acquire such cues from images.

This paper focuses on a fine-grained table recognition problem, whose purpose is to predict the coordinates of each table cell. We present a novel framework to achieve this goal. Specifically, we first build a cell relationship network, which merges the visual features and spatial features to detect the neighbors of each table cell. A distance-based sample weight is proposed to handle the class imbalance

2379-2140/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDAR.2019.00125

problem encountered in the training stage. Then, according to the detected relationships, a weighted graph is created to infer the table syntactic structure. The proposed framework is verified on two table datasets, and our results demonstrate the effectiveness of the proposed model at detecting the relationships between table cells and converting the table image into its syntactic representation. The contributions of this paper can be summarized as follows:

- We present a novel framework that combines cell relationship detection and graph inference to convert a table image into its syntactic representation.
- We devise a distance-based weight, which is applied to deal with the class imbalance problem when training the cell relationship network.

II. Related Work

A. Table Structure Recognition

Early studies on table structure recognition started by identifying the table boundary [9], [10]. Then, the methods in [11], [12] further segment the table rows and columns. Yildiz *et al.* [13] and Fang *et al.* [14] developed methods to classify cells as either *'data cells'* or *'header cells'* representing the coarse-grained structure of a table. More recent works [4], [5] have used deep-learning methods to recognize table structure. More specifically, Schreiber *et al.* [5] presented a deep learning-based solution for the identification of rows, columns, and cells, where transfer learning is performed by augmenting and fine-tuning a FCN semantic segmentation model [15]. Nishida *et al.* [4] proposed a combined RNN-CNN architecture to classify tables into six types. Chen *et al.* [6] used tokens from the PDF file as features to detect the hierarchical relationships between pairs of rows, while Xue *et al.* [16] used a projection method to segment two columns of interest in a table image. In contrast to these previous works, we recognize the fine-grained table structure from an image by inferring the specific coordinates for each table cell.

B. Visual Relationship Detection

A visual relationship is usually characterized by a triplet in the form of (*s, r, o*), where *s*, *r*, and *o* represent the *subject*, *relationship predicate*, and *object*, respectively, e.g., (*man, on, bicycle*). Some previous methods [17]–[19] regard each combination of (*s, r, o*) as a distinct class, but then face difficulties when enumerating all possible combinations. Recent works consider an alternative paradigm that separates the object detection and relationship recognition. Lu *et al.* [20] merged appearance features and a language prior as the representation of a relational pair, and Dai *et al.* [21] proposed a Deep Relational Network that exploits both spatial configurations and statistical dependencies among the relational triplets. Zhuang *et al.* [22] proposed a context-aware interaction recognition framework, which can associate the interaction with certain visual appearances in a semantic space. The method of [23] integrates multiple cues and designs a structural ranking loss to detect visual relationships. Inspired by these works, our method detects the relationships between pairs of table cells, which are then used for the table structure inference.

III. The Proposed Framework

This section presents the proposed framework to reconstruct the syntactic representation from a table image. The approach starts with a cell relationship network (III-A) that detects the neighbors for each table cell in four directions ('*Left*', '*Right*', '*Up*', and '*Down*')[1]. During the training stage, we propose a type of sample weight based on the distance between two cells to solve the class imbalance problem. Finally, we describe the algorithm (III-B) that infers the table structure on the graph created through the detected neighbors.

A. Neighbor Cells Detection

A crucial step in our method is to represent the relationships between table cells. In this paper, the neighbor position is taken as the relationship representation of a pair of cells. The target of this section is to detect the neighbors of a table cell in all four directions ('*L*', '*R*', '*U*', and '*D*'). Specifically, we use C to denote the set of all cells with content bounding boxes. For each pair $\{(c_i, c_j)|i \neq j\} \in C$, c_i represents the target cell and c_j is one of the possible neighbors around c_i. The relationship between c_i and c_j is denoted as $r_{i,j} \in R$, where $R = \{'None', 'L', 'R', 'U', 'D'\}$. '*None*' means that c_j is not a neighbor of c_i. A deep cell relationship network is built to detect all cells' neighbors, which merges the deep visual features and relative spatial features as the representation for each relational pair of cells. During training and inference, we assume that all content bounding boxes are observed wherever they are acquired from the ground-truth or a text detector. For each relational pair, the loss is multiplied by a distance-based weight to optimize the training process. Further details about the network are presented below.

Deep Cell Relationship Network: As illustrated in Fig. 2, given a table image with content bounding boxes, the cell relationship network concatenates the deep visual features and relative spatial features as the joint feature of a relational pair. For the visual features, a spatial mask is first generated to represent the table, which has the same pixel-level width and height as the table area and all content bounding boxes are filled with 255. Second, the mask goes through a VGG16 [24] network to obtain a group of feature maps that are then separated for each table cell by a region of interest (RoI) pooling [25]. The separated features are fed into two fully connected layers to get the final deep visual features. We use $b_i = (x_1, y_1, x_2, y_2)$ to denote the coordinates of the

[1]For convenience, the four directions will be abbreviated as '*L*', '*R*', '*U*', and '*D*' in this paper.

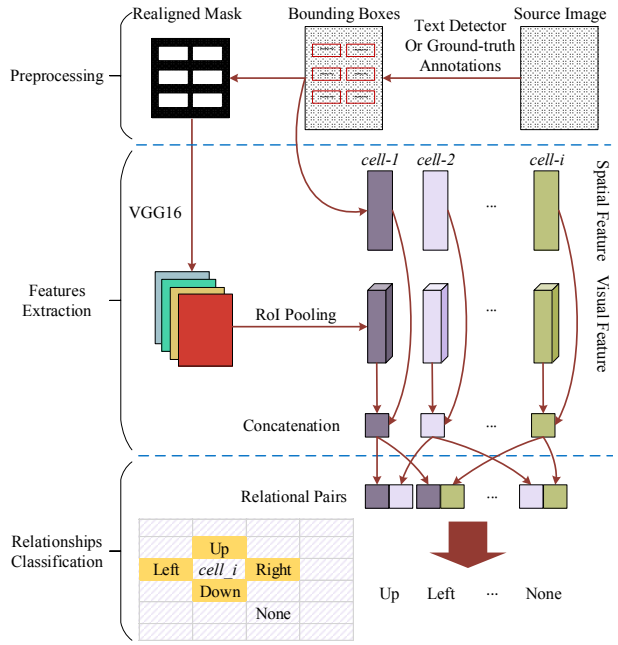


Figure 2: The cell relationship network architecture. The network takes a realigned mask as the input, and concatenates the visual and spatial features as the representation for each table cell. Then, the features of any two cells are combined to predict their neighbor relationship.

content bounding box for c_i. The relative spatial feature is represented as $b_i' = (\frac{x_1}{W}, \frac{y_1}{H}, \frac{x_2}{W}, \frac{y_2}{H})$, where W and H are the width and height of the table image, respectively. The visual feature and the spatial feature are concatenated as the representation of a table cell. We fuse the features of two table cells to predict the neighbor relationship, which includes five categories: '*None*', '*Left*', '*Right*', '*Up*', and '*Down*'.

Distance-based Sample Weight (DSW): During training, the learning process is susceptible to the class imbalance problem. Consider for example a $n \times n$ table. If all table cells have contents, the proportion of all relationship classes is about $\{None : L : R : U : D = O(n^4) : O(n^2) : O(n^2) : O(n^2)\}$. The number of negatives ('*None*') is much greater than the positives ('*L*', '*R*', '*U*', and '*D*'). Two common ways to tackle this problem are data resampling and cost-sensitive learning [26], [27]. Data resampling tries to alter the data distribution by random under-sampling or over-sampling. However, under-sampling will make the classification boundaries vague, and replication-based over-sampling usually causes overfitting. The latter method aims to give more emphasis to minority classes. Specifically, for neighbor cell detection, we find that the hard negatives (neighbors in diagonals) and minority positives (neighbors in the four directions) distribute around the target cell. Therefore, a distance-based sample weight is proposed for the loss of a pair of cells (c_i, c_j):

$$\begin{cases} \mathcal{L}'(r_{i,j}) = \lambda_{i,j} \cdot \mathcal{L}(r_{i,j}) \\ \lambda_{i,j} = \exp\{-(\frac{(x_j^{'}-x_i^{'})^2 \cdot \alpha}{W_t} + \frac{(y_j^{'}-y_i^{'})^2 \cdot \alpha}{H_t})\} + \beta \end{cases} \quad (1)$$

where $\mathcal{L}(r_{i,j})$ is the cross-entropy loss of the neighbor relationship of $r_{i,j}$; $\lambda_{i,j}$ is the proposed distance-based sample weight (DSW); $(x_i^{'}, y_i^{'})$ and $(x_j^{'}, y_j^{'})$ are the central point coordinates of c_i and c_j, respectively; W_t and H_t are the width and height of the table area, respectively; and α and β are two parameters that are set to 3.0 and 2.0 respectively in our experiments. As c_j approaches c_i in the table, it becomes increasingly likely that c_j is a neighbor of c_i or a hard negative. On this condition, the DSW assigns a larger weight to the loss so that the learning process is not dominated by majority negatives.

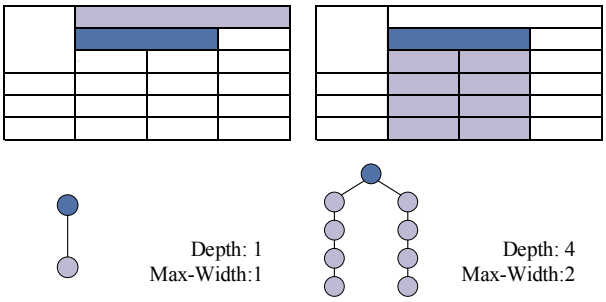
B. Table Structure Inference

The syntactic structure of a table includes *the shape of the table* and *the coordinates of each table cell*. The coordinates of c_i can be denoted as $s_i = (col_1^i, row_1^i, mc_i, mr_i)$, where col_1^i and row_1^i represent the cell's start column and start row. mc_i and mr_i are the multi-column and multi-row attributes, which represent the number of columns and rows occupied by c_i. We use $(tRows, tCols)$ to denote the shape of a table, which indicates the number of columns and rows in the table. $tRows$ can be obtained by calculating the maximum sum of row_1^i and mr_i. Similarly, $tCols$ is the maximum sum of col_1^i and mc_i. In this section, a weighted graph is created for table structure inference. Algorithm 1 summarizes the whole process.

Graph Creation: We first create a graph based on the detected neighbor relationships. Then mc_i and mr_i are inferred from the graph, and in turn used to update the graph.

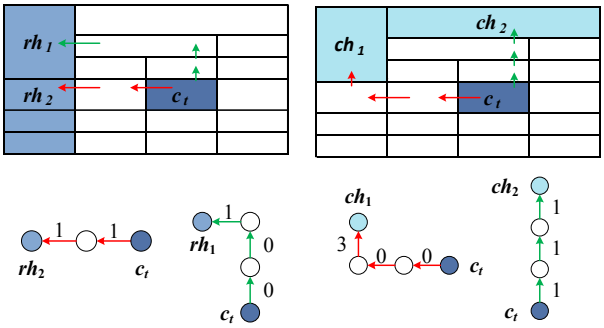
For one table cell c_i, we use $N_i \in \mathcal{N}$ to denote the set of all its detected neighbors. Each element $n_j^d \in N_i$ represents that the jth table cell c_j is a neighbor of c_i in the direction of d, where $d \in \{$ 'L', 'R', 'U', 'D'$\}$. Then, a weighted graph is created as $G = (C, E)$, where C is the set of all table cells. The edge of two table cells is denoted as $e_{i,j} \in E$, which represents the number of columns or rows needed to move c_j from its start row or start column to that of c_i. $e_{i,j}$ is initialized with 1 if $n_j \in N_i$; otherwise, it is set to 0. Then, the created G is used to infer the multi-column attribute mc_i and the multi-row attribute mr_i. For each table cell, we search the graph G along one of the four directions to build a subtree. The searching along a branch is stopped when the last cell in this branch has no neighbors in the searching direction or it has multiple neighbors in the opposite direction. After the four subtrees have been built, mc_i and mr_i can be represented as:

$$\begin{cases} mc_i = \max(w_d | d \in \{\text{'}U\text{'}, \text{'}D\text{'}\}) \\ mr_i = \max(w_d | d \in \{\text{'}L\text{'}, \text{'}R\text{'}\}) \end{cases} \quad (2)$$



(a) Up-subTree. (b) Down-subTree.

Figure 3: Multi-column attribute inference. Depth-first searching is applied in the (a) 'U' and (b) 'D' directions to build subtrees for the target cell. The maximum width of up-subtree and down-subtree determines the number of columns occupied by the target cell.



(a) Distances from the target cell to row headers. (b) Distances from the target cell to column headers.

Figure 4: Cell localization. Calculation of the distances from a target cell to the (a) row and (b) column headers.

where w_d is the maximum width of the subtree built in the direction d. Fig. 3 gives an example of multi-column inference to help understand this process.

In Fig. 3a, the target cell (dark blue) has only one neighbor cell in the 'U' direction, and this sole neighbor cell does not have any other neighbors in this direction. Therefore, the up-subtree built from the target cell only has one node. In Fig. 3b, a similar process is applied in the 'D' direction to build the down-subtree, whose depth and maximum width are 4 and 2, respectively. Finally, we choose the maximum width of the up-subtree and down-subtree as the number of columns occupied by the target cell. If a cell occupies more than one column or row, it may need more steps to move its neighbor cells to its start row or column. The edges $e_{i,j}$ and $e_{j,i}$ in the graph G thus need to be updated as:

$$e_{j,i} = e_{i,j} = \begin{cases} mc_i, & \text{if } n_j^d \in N_i \text{ and } d \in \{\text{'}R\text{'}\} \\ mr_i, & \text{if } n_j^d \in N_i \text{ and } d \in \{\text{'}D\text{'}\} \end{cases} \quad (3)$$

Cell Location: After inferring mc_i and mr_i, col_1^i and row_1^i are located via two steps: finding table headers and calculating the distances from cells to table headers.

First, we search the graph G to select cells with no neighbors in the 'L' and 'U' directions as the candidate row and column headers, respectively. For each row header candidate, its up- and down-subtree are built as described above. A candidate will be filtered if a table cell in its up-subtree or down-subtree has left neighbors but the neighbors are not included in the two subtrees. A similar process can be applied to filter the column headers.

We then use Dijkstra's algorithm to calculate the shortest path from a table cell to each column or row header. Among all of the paths from the target cell to column or row headers, the longest distance is selected as the start row (row_1^i) or start column (col_1^i) because the distance may be shortened if a cell with multiple columns or rows exists in the path.

In Fig. 4a, two paths from the dark cell c_t to the row headers (rh_1, rh_2) are marked in red and green, respectively. The red path ($c_t \rightarrow rh_2$) goes through a left-neighbor of the cell c_t to the row header rh_2. The distance of this path is the sum of the two edges (i.e., 2). The green path ($c_t \rightarrow rh_1$) first heads upward two cells, and then turns left to the target row header rh_1. As only the horizontal distance is of interest when finding the distance to the row header, the edges traversed in the vertical direction are not considered. Therefore, the distance of the green path is only one (the edges are calculated according to Eq. 3). Because a cell in the green path spans multiple columns, the distance is shorter than that of the red path. The longest distance from c_t to row headers is chosen as the column coordinate col_1^t. An example of how to calculate the row coordinate is shown in Fig. 4b.

IV. Experiments

A. Dataset and Metrics

Chinese Medical Documents Dataset (CMDD) [16]. The CMDD is an image dataset of medical laboratory reports containing 238 documental images, each including two tables. The first table lists a patient's information in five rows and four columns. The second table reports the details of test results, which consists of $n(n \geq 1)$ rows and six columns. Some of the cells in the second table may be empty. The tables are divided into 80% for training and 20% for testing.

ICDAR 2013 Table Competition Dataset [28]. The ICDAR 2013 table dataset consists of 67 PDF documents with 156 tables, which were collected from the internet. This dataset can be divided into two categories: the US set and the EU set. The tables are in different styles and from various domains. The US set has more non-ruled tables and complex header structures than the EU set. Because this paper focuses on table reconstruction from images, all PDF files are converted to images prior to experimentation. Only

Algorithm 1: Table Structure Inference

Input: $\mathcal{N}$, the neighbor relationships set.
C, the cells set.
Output: S, the coordinates set for all cells.
$(tRows, tCols)$, the shape of table.

$G = (C, E)$; S // Initialization.
for $c_i \in C$ **do**
  $mc_i = 1, mr_i = 1$
  for $d \in \{'L', 'R', 'U', 'D'\}$ **do**
    $subTree, width =$ `BuildTree` (G, c_i, d)
    if $d \in \{'L', 'R'\}$ **then**
      $mr_i =$ `Max` $(mr_i, width)$
    else
      $mc_i =$ `Max` $(mc_i, width)$
    $S = S.update(mc_i, mr_i)$
    $G = G.update(mc_i, mr_i)$
$rowHeaders = list(), colHeaders = list()$
for $N_i \in \mathcal{N}$ **do**
  if `Count` $(n_j^{'L'} \in N_i)$ == 0 **then**
    $rowHeaders.append(c_i)$
  if `Count` $(n_j^{'U'} \in N_i)$ == 0 **then**
    $colHeaders.append(c_i)$
$rowHeaders =$ `Filter` $(rowHeaders)$
$colHeaders =$ `Filter` $(colHeaders)$
$tRows = 0, tCols = 0$
for $c_i \in C$ **do**
  /* Distances to row headers (RH) and column headers (CH). */
  $disToRH = list(), disToCH = list()$
  for $c_j \in rowHeaders$ **do**
    $minPath =$ `Dijkstra` (G, c_i, c_j)
    $disToRH.append($`Dis` $(G, minPath))$
  for $c_j \in colHeaders$ **do**
    $minPath =$ `Dijkstra` (G, c_i, c_j)
    $disToCH.append($`Dis` $(G, minPath))$
  $row_1^i =$ `Max` $(disToCH)$
  $col_1^i =$ `Max` $(disToRH)$
  $S = S.update(col_1^i, row_1^i)$
  $tRows =$ `Max` $(tRows, row_1^i + mr_i)$
  $tCols =$ `Max` $(tCols, col_1^i + mc_i)$
return S, (tRows,tCols)

50% of the tables are used for training to ensure sufficient samples for evaluation.

Following the text detection workflow, only cells containing data are annotated with bounding boxes and cell coordinates, meaning that empty cells are unannotated.

We evaluate the proposed method in two ways: **neighbor relationship detection** and **cell location inference**. Neighbor relationship detection follows the metrics used in [28]. The prediction of the deep cell relationship network is presented as a tuple $(c_i, c_j, r_{i,j})$, which is compared to the ground truth by using precision and recall measures. For cell location inference, a table cell coordinate is represented as (col_1, row_1, mc, mr) in our method. We transform this representation into $(col_1, row_1, col_2, row_2)$ as in ICDAR 2013 table dataset, where $col_2 = col_1 + mc$ and $row_2 = row_1 + mr$. The accuracy of the four coordinate values is calculated as the metric of cell location inference metric. We also denote *cell-loc* as the accuracy of all these four values being predicted correctly for a table cell.

Table I: Results of neighbor relationship detection.

	CMDD		ICDAR 2013 Dataset	
Method	Precision	Recall	Precision	Recall
Projection	0.979	0.987	0.157	0.666
Ours(RUS)	0.868	**0.997**	0.795	0.547
Ours(CW)	0.962	0.992	0.822	0.412
Ours(DSW)	**0.999**	**0.997**	**0.926**	0.447
Ours(DSW+RUS)	∼	∼	0.734	**0.747**

B. Implementation Details

For neighbor cells detection, we fork the architecture in [23] to build our neural network. The network is trained with the RMSprop optimizer, and the learning rate is set to 0.00005. In theory, given a table with n cells, the proposed model can output all the detection results for $n(n-1)$ relational pairs. However, GPU memory requirements rise dramatically as n increases. So, the relational instances are split into multiple batches for training and testing. When training on ICDAR 2013 table dataset, the model is initialized with the parameters trained on CMDD because of the insufficient data and diverse styles in ICDAR 2013 table dataset.

C. Comparative Results

We first compare the proposed **DSW** with the methods of **projection** [16], random under-sampling (**RUS**) and class weight (**CW**) [29] on CMDD. The realigned mask in Section III-A is directly treated as the input of the projection method because we assume that content bounding boxes are observed before experiments. The result of projection is postprocessed by a median and Gaussian filter, respectively. The proportion of negatives and positives in CMDD is about 95 : 5. For the RUS, redundant negatives are randomly deleted to keep the proportion no more than 2 : 1. According to the proportion of five cell relationship classes (*None*, *L*, *R*, *U*, *D*), the CW is set to 0.6 : 1.0 : 1.0 : 1.2 : 1.2. As shown in Table I, the RUS can result in a high recall by deleting redundant negatives. However, some hard samples are inevitably removed. Rather than only paying attention to minority positives like the CW, the proposed DSW also gives

Table II: Results of cell location inference.

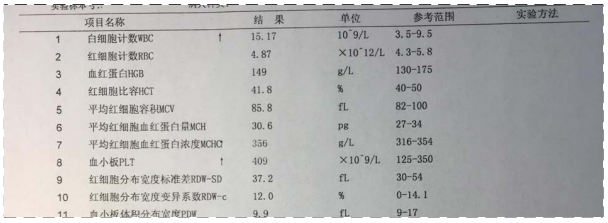
CMDD					
Method	*cell-loc*	row_1	row_2	col_1	col_2
Projection	0.957	0.976	0.975	0.976	0.976
Ours	**0.999**	**0.999**	**0.999**	**0.999**	**0.999**
ICDAR 2013 Dataset					
Method	*cell-loc*	row_1	row_2	col_1	col_2
Projection	**0.031**	0.096	0.099	0.141	0.142
Ours(RUS)	0.015	0.089	0.091	0.135	0.131
Ours(CW)	0.020	0.101	0.099	0.118	0.113
Ours(DSW)	0.015	0.053	0.064	**0.166**	**0.163**
Ours(DSW+RUS)	0.019	**0.111**	**0.119**	0.164	0.145

more weight to hard samples, and achieves the highest recall (0.997) and precision (0.999). Furthermore, to evaluate the performance of our method for cell location inference, we compare it with the projection method. The results presented in Table II show that the proposed method can effectively infer the cell location according to the neighbor relationships between cells.

We next conduct experiments on ICDAR 2013 table dataset to evaluate the performance of the proposed method on more complex data. This dataset has twice the number of negatives than CMDD with an equivalent number of positives. So, referring the configuration of CMDD, the CW is changed to $0.4 : 1.0 : 1.0 : 1.2 : 1.2$. Because ICDAR 2013 table dataset contains more complex samples, the proportion of negatives and positives for the RUS is maintained at no more than $40 : 1$ according to experiments. The DSW parameters are the same as those used on CMDD. The results of neighbor relationship detection in Table I show that the proposed method achieves the highest precision of all methods tested. However, not all methods perform well at recall. Then, we apply DSW and RUS simultaneously, which balances the precision and recall. The cell location inference results in Table II show that the DSW can significantly improve the accuracy of col_1 and col_2. When both DSW and RUS are applied, the accuracy of row_1 and row_2 is maximized. However, the projection method attains the highest accuracy on *cell-loc*.

D. Discussion

Why is the cell location accuracy so low on ICDAR dataset, even with high precision and recall of neighbor cells detection: Most tables in ICDAR 2013 table dataset have different styles and complex structure compared to the tables in CMDD, as shown in Fig. 5. The model is hardly trained to convergence with insufficient data. On the other hand, the location inference for one cell depends on other cells. For example, if the only column header cell gets a false prediction, the remaining cells will be assigned the incorrect coordinates. However, like CMDD, when the precision and recall for neighbor cells detection approaches to 1.0, the cell location accuracy will also improve significantly.



	项目名称		结 果	单位	参考范围	实验方法
1	白细胞计数WBC	↑	15.17	10^9/L	3.5-9.5	
2	红细胞计数RBC		4.87	$\times10^{12}$/L	4.3-5.8	
3	血红蛋白HGB		149	g/L	130-175	
4	红细胞比容HCT		41.8	%	40-50	
5	平均红细胞容积MCV		85.8	fL	82-100	
6	平均红细胞血红蛋白量MCH		30.6	pg	27-34	
7	平均红细胞血红蛋白浓度MCHC		356	g/L	316-354	
8	血小板PLT	↑	409	$\times10^9$/L	125-350	
9	红细胞分布宽度标准差RDW-SD		37.2	fL	30-54	
10	红细胞分布宽度变异系数RDW-c		12.0	%	0-14.1	
11	血小板体积分布宽度PDW		9.9	fL	9-17	

(a)

Year of data			
2007–08	2008–09	2009–10	2010–11
Enrollment, in thousands			
49,293	49,266	49,373	49,484
Projected enrollment, in thousands			
49,644	49,825	50,067	50,353
49,470	49,623	49,788	50,034
†	49,265	49,312	49,386
†	†	49,282	49,306

(b)

Figure 5: Example table images from (a) CMDD and (b) ICDAR 2013 table dataset.

What is the weakness of the proposed method: When the table is sparse, some cells may have no other neighbor cells around them. So, these cells cannot connect with others when creating the graph from detected neighbor relationships. This is also a drawback of the evaluation [28] for table structure recognition, in which the neighbor relationship between two adjacent blank cells, or a content cell and a blank cell, are not considered.

V. Conclusion

This paper presents a novel framework to convert a table image into its syntactic representation. Specifically, given a table image with content bounding boxes, a deep cell relationship network is built to detect neighbors for each table cell. During the training stage, a distance-based sample weight is proposed to deal with the class imbalance problem. According to the detected relationships, a weighted graph is created to connect the table cells. By searching this graph, we infer the syntactic table structure: the shape of the table and the coordinates of each table cell. Experimental evaluation of the proposed method on two datasets demonstrates that it can effectively detect the neighbor relationships and infer the location for each table cell. We will explore methods to deal with more complex table structures in future studies.

References

[1] P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao, and C. Wu, "Recovering semantics of tables on the web," in *Proceedings of the VLDB Endowment*, vol. 4, no. 9, 2011, pp. 528–538.

[2] J. Wang, H. Wang, Z. Wang, and K. Q. Zhu, "Understanding tables on the web," in *International Conference on Conceptual Modeling*, 2012, pp. 141–155.

[3] T. T. Nguyen, Q. V. H. Nguyen, M. Weidlich, and K. Aberer, "Result selection and summarization for web table search," in *Proceedings of the 31st International Conference on Data Engineering (ICDE)*. IEEE, 2015, pp. 231–242.

[4] K. Nishida, K. Sadamitsu, R. Higashinaka, and Y. Matsuo, "Understanding the semantic structures of tables with a hybrid deep neural network architecture." in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2017, pp. 168–174.

[5] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, "Deepdesrt: Deep learning for detection and structure recognition of tables in document images," in *Proceedings of the 14th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 1162–1167.

[6] X. Chen, L. Chiticariu, M. Danilevsky, A. Evfimievski, and P. Sen, "A rectangle mining method for understanding the semantics of financial tables," in *Proceedings of the 14th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 268–273.

[7] Z. Chen and M. Cafarella, "Automatic web spreadsheet data extraction," in *Proceedings of the 3rd International Workshop on Semantic Search over the Web*. ACM, 2013, p. 1.

[8] ——, "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)*. ACM, 2014, pp. 1126–1135.

[9] Y. Liu, P. Mitra, and C. L. Giles, "Identifying table boundaries in digital documents via sparse line detection," in *Proceedings of the 17th International Conference on Information and Knowledge Management*. ACM, 2008, pp. 1311–1320.

[10] Y. Liu, K. Bai, P. Mitra, and C. L. Giles, "Improving the table boundary detection in pdfs by fixing the sequence error of the sparse lines," in *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2009, pp. 1006–1010.

[11] B. Coüasnon and A. Lemaitre, "Recognition of tables and forms," in *Handbook of Document Image Processing and Recognition*. Springer, 2014, pp. 647–677.

[12] S. Seth and G. Nagy, "Segmenting tables via indexing of value cells by table headers," in *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2013, pp. 887–891.

[13] B. Yildiz, K. Kaiser, and S. Miksch, "pdf2table: A method to extract table information from pdf files," in *Proceedings of the 2nd Indian International Conference on Artificial Intelligence*, 2005, pp. 1773–1785.

[14] J. Fang, P. Mitra, Z. Tang, and C. L. Giles, "Table header detection and classification," in *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 2012, pp. 599–605.

[15] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 3431–3440.

[16] W. Xue, Q. Li, Z. Zhang, Y. Zhao, and H. Wang, "Table analysis and information extraction for medical laboratory reports," in *Proceedings of the 4th International Conference on Cyber Science and Technology*. IEEE, 2018, pp. 193–199.

[17] M. A. Sadeghi and A. Farhadi, "Recognition using visual phrases," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011, pp. 1745–1752.

[18] P. Das, C. Xu, R. F. Doell, and J. J. Corso, "A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2013, pp. 2634–2641.

[19] S. K. Divvala, A. Farhadi, and C. Guestrin, "Learning everything about anything: Webly-supervised visual concept learning," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014, pp. 3270–3277.

[20] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei, "Visual relationship detection with language priors," in *Proceedings of European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 852–869.

[21] B. Dai, Y. Zhang, and D. Lin, "Detecting visual relationships with deep relational networks," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 3298–3308.

[22] B. Zhuang, L. Liu, C. Shen, and I. Reid, "Towards context-aware interaction recognition for visual relationship detection," in *Proceedings of the International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 589–598.

[23] K. Liang, Y. Guo, H. Chang, and X. Chen, "Visual relationship detection with deep structural ranking," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2018.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.

[26] C. Huang, Y. Li, C. Change Loy, and X. Tang, "Learning deep representation for imbalanced classification," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 5375–5384.

[27] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge & Data Engineering*, no. 9, pp. 1263–1284, 2008.

[28] M. Göbel, T. Hassan, E. Oro, and G. Orsi, "Icdar 2013 table competition," in *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2013, pp. 1449–1453.

[29] A. More, "Survey of resampling techniques for improving classification performance in unbalanced datasets," *arXiv preprint arXiv:1608.06048*, 2016.