IM-TQA: A Chinese Table Question Answering Dataset with Implicit and Multi-type Table Structures

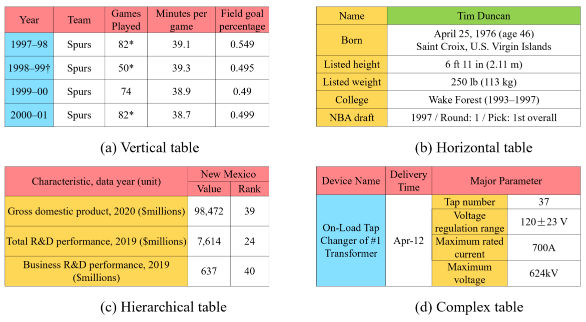
Anonymous ACL submission

Abstract

Various datasets have been proposed to promote the development of Table Question Answering (TQA) technique. However, the problem setting of existing TQA benchmarks suffers from two limitations. First, they directly provide TQA models with explicit table structures where row headers and column headers of the table are explicitly annotated during inference. Second, they only consider tables of limited types and ignore other tables especially complex tables. Such simplified problem setting cannot cover practical scenarios where TQA models need to process tables without header annotations in the inference phase or tables of different types. To address this issue, we construct a new TQA dataset with implicit and multi-type table structures, named IM-TQA, which not only requires the model to answer questions without header annotations beforehand but also to handle multi-type tables including previously neglected complex tables. We investigate the performance of recent methods on our dataset and find that existing methods struggle in processing implicit and multi-type table structures. Correspondingly, we propose an RGCN-RCI framework outperforming recent baselines. We will release our dataset to facilitate future research.

1 Introduction

To gain useful information from tables, Table Question Answering (TQA) technique was developed and used to answer natural language questions about tables (Zheng et al., 2022; Hui et al., 2022; Herzig et al., 2020). Researchers also proposed various TQA datasets which aim at different scenarios (Zhong et al., 2017; Iyyer et al., 2017; Chen et al., 2020). As the performance of TQA models continues to improve, they have been widely used in the intelligent data analysis tools, e.g., Power



Year	Team	Games Played	Minutes per game	Field goal percentage
1997–98	Spurs	82*	39.1	0.549
1998–99†	Spurs	50*	39.3	0.495
1999–00	Spurs	74	38.9	0.49
2000–01	Spurs	82*	38.7	0.499

(a) Vertical table

Name	Tim Duncan
Born	April 25, 1976 (age 46) Saint Croix, U.S. Virgin Islands
Listed height	6 ft 11 in (2.11 m)
Listed weight	250 lb (113 kg)
College	Wake Forest (1993–1997)
NBA draft	1997 / Round: 1 / Pick: 1st overall

(b) Horizontal table

Characteristic, data year (unit)	New Mexico	
	Value	Rank
Gross domestic product, 2020 ($millions)	98,472	39
Total R&D performance, 2019 ($millions)	7,614	24
Business R&D performance, 2019 ($millions)	637	40

(c) Hierarchical table

Device Name	Delivery Time	Major Parameter	
On-Load Tap Changer of #1 Transformer	Apr-12	Tap number	37
		Voltage regulation range	120±23 V
		Maximum rated current	700A
		Maximum voltage	624kV

(d) Complex table

Figure 1: Different types of tables and their header cells: Column Attribute (red), Column Index (green), Row Attribute (yellow), Row Index (blue).

BI[1], Tableau[2] and Ideas[3].

Though previous datasets have promoted the development of TQA technique, the problem setting of existing benchmarks suffers from two limitations. First, **in the depth of table structure understanding, existing benchmarks only evaluate the performance of TQA models on tables with explicit table structures,** which means during inference locations and directions of headers are annotated and treated as model input. For example, Text2SQL benchmarks offer annotated column headers (Zhong et al., 2017; Yu et al., 2018) and recent hierarchical table datasets also contain hierarchical header annotations (Katsis et al., 2022; Cheng et al., 2022), which are available at inference time. This setting artificially lowers the difficulty of the task. Nevertheless, in practical scenarios, TQA model may encounter plenty of tables without labeled headers. Manually annotating headers for these tables is prohibitively expensive and time-consuming. As a result, a benchmark is needed to evaluate the performance of TQA models on tables with implicit table structures. Here, "implicit table structures" represents that the annotations of headers in a table are not available during inference.

[1]https://powerbi.microsoft.com/en-us/
[2]https://www.tableau.com/
[3]https://support.microsoft.com/en-us/office/analyze-data-in-excel-3223aab8-f543-4fda-85ed-76bb0295ffc4

Second, **in the breadth of supported table types, existing TQA benchmarks only consider limited table types and ignore complex tables with flexible header locations.** Previous datasets mainly focus on vertical or hierarchical tables whose header cells only locate on the top or left side (Pasupat and Liang, 2015; Wang et al., 2020b; Guo et al., 2021; Cheng et al., 2022), as shown in Figure 1 (a) and (c), but neglect the fact that TQA model may require handling multi-type tables, especially complex tables whose header cells may appear at any position, as illustrated in Figure 1 (d). Complex tables are prevalent in professional equipment specifications and record sheets, which are beyond the ability of current TQA systems.

Above analyses show that the problem setting of previous TQA benchmarks restricts the application of TQA models. In this paper, we define a new problem, table question answering over implicit and multi-type table structures. In this problem setting, annotations of table headers are not available during inference and models need to comprehend implicit and multi-type table structures to answer questions. To facilitate the study on this problem, we build the first Chinese **T**able **Q**uestion **A**nswering dataset with **I**mplicit and **M**ulti-type table structures, named **IM-TQA**, which consists of 1,200 tables and 5,000 questions (Section 3). Our dataset includes tables of four types from different domains, especially complex tables neglected by published studies. We annotate different tables not only with questions of various types and their answer cells, but also with row headers and column headers.

IM-TQA presents a strong challenge to previous TQA models. Because of the inability to understand implicit and diverse table structures, the state-of-the-art baseline RCI (Katsis et al., 2022) only achieves 49.6% overall accuracy and 23.8% accuracy on complex tables. To improve the performance of RCI, we propose an RGCN-RCI framework (Section 4). Specifically, this framework solves the new problem in two stages: (1) Table is modeled as graph and an RGCN (Schlichtkrull et al., 2017) is used to comprehend table structure and predict table header cells. (2) Based on predictions of RGCN, header information is merged into the text sequence representation of each row or column, which helps RCI model to predict whether a row or column contains the answer or not. Our framework shows high effectiveness and improves the accuracy on all tables and complex tables by 3.8% and 8.2% respectively, proving the significance of understanding implicit and multi-type table structures. To summarize, we conclude our contributions as follows:

- Based on the practical demand for answering questions over various tables under the circumstance that header annotations are not available during inference, we define a new problem, table question answering over implicit and multi-type table structures, which is complementary to traditional TQA problem, i.e., TQA over explicit and limited (usually single-type) table structures.
- We construct and publicly release a new dataset, IM-TQA, to promote the research on this problem. Our dataset includes tables of four different types with implicit structures, especially complex tables ignored by former benchmarks.
- We investigate the performance of existing methods on our dataset and propose an RGCN-RCI framework which outperforms state-of-the-art baselines on all table types.

2 Problem Statement

In this section, we define the problem of table question answering over implicit and multi-type table structures. A table consists of multiple cells. A table t can be defined as $t \doteq \{\mathbb{P}, \mathbb{R}, \mathbb{V}\}$, where $\mathbb{P}$, $\mathbb{R}$ and $\mathbb{V}$ represents the set of position information, functional roles and values of cells in table t respectively. Table cells can be categorized into different types according to their functional roles, e.g., some cells are column headers and others are data cells.

Under the setting of traditional TQA problem, table headers are explicitly provided in the inference phase (Zhong et al., 2017; Yu et al., 2018) or can be easily obtained by heuristic methods designed for specific table type (Cheng et al., 2022; Katsis et al., 2022). In this setting, cell functional roles $\mathbb{R}$ are directly provided for the model. Thus, the model $f(\cdot)$ outputs answer y given a natural language question q, i.e., $y = f(q, t) = f(q, \mathbb{P}, \mathbb{R}, \mathbb{V})$. However, in the setting of real applications where the model may need process implicit and multi-type table structures, functional roles $\mathbb{R}$ are not available in advance. Thus, TQA over implicit and multi-type table structures can be formalized as:

$y = f(q, t^{'}) = f(q, \mathbb{P}, \mathbb{V})$. This problem setting poses a great challenge to existing methods. Unfortunately, none of previous benchmarks serves as testbed for this problem. To fill the gap, we build IM-TQA, which includes multi-type tables and requires understanding implicit table structures. Compared with previous benchmarks, the improvements of IM-TQA are demonstrated in Figure 2.

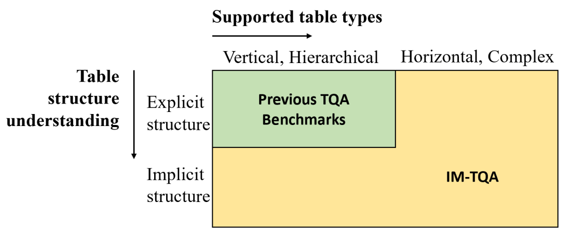


Figure 2: The improvements of IM-TQA compared with previous benchmarks.

3 Dataset Construction and Analysis

In this section, we first introduce table types considered in our dataset, and then elaborate the dataset construction procedure consisting of 4 steps. We recruit 10 professional annotators and provide them with sufficient instruction. The dataset construction totally costs 1,200 working hours. The ethical considerations will be discussed in the Section 9.

3.1 Considered Table Types

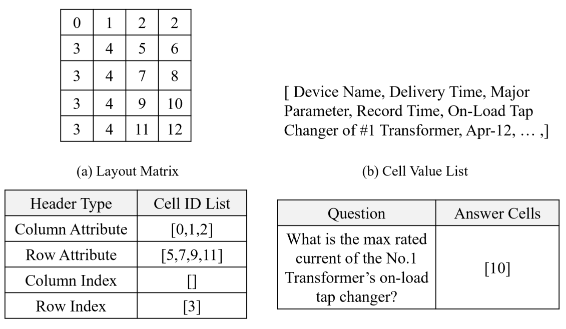
As shown in Figure 1, we divide tables into 4 types according to their layout and header locations, which is in line with previous work (Wang et al., 2021; Ghasemi-Gol and Szekely, 2018) with complex table as an important complement. More complex table examples are shown in Appendix D.

Vertical Table Table data is arranged in the vertical direction, with the first row as column headers and other rows as data tuples.

Horizontal Table Table data is arranged in the horizontal direction, with the first column as row headers and other columns as data tuples.

Hierarchical Table Table data is arranged in both vertical and horizontal directions, with headers exhibiting a multi-level hierarchical structure.

Complex Table In tables of above 3 types, header cells locate on the top or left side. But in complex tables, headers may locate at any position and can be mixed with data cells, as depicted in Figure 1 (d). Such tabular structures often appear in professional equipment specifications and record sheets, presenting a great challenge to existing methods.



0	1	2	2
3	4	5	6
3	4	7	8
3	4	9	10
3	4	11	12

(a) Layout Matrix

[Device Name, Delivery Time, Major Parameter, Record Time, On-Load Tap Changer of #1 Transformer, Apr-12, … ,]

(b) Cell Value List

Header Type	Cell ID List
Column Attribute	[0,1,2]
Row Attribute	[5,7,9,11]
Column Index	[]
Row Index	[3]

(c) Cell Type Annotation

Question	Answer Cells
What is the max rated current of the No.1 Transformer's on-load tap changer?	[10]

(d) QA Pair Annotation

Figure 3: An example of table storage format and annotations. Cell IDs constitute the layout matrix in (a) and are indices of cell value list in (b). Cells with same cell IDs stand for merged cells.

3.2 Table Collection and Storage

To ensure the diversity of table data, we collect tables from open websites of different domains. Data sources include company annual reports from different industries[4] such as manufacturing, medicine and education; entries from Baidu Encyclopedia[5] on science, professional equipment, etc. Tables in these web pages are extracted to Excel files by annotators. We correct typos in the collected tables and filter tables without meaningful data. Finally, we preserve 300 tables for every table type. Figure 4 shows the distribution of domains.

In order to store various tables, we design a storage method which separately stores table structure and table content.To store table structure, a *cell ID* is assigned to each table cell in the row-first order. For a table including m rows and n columns, its cell IDs constitute an $m \times n$ matrix representing cell locations. This layout matrix contains table structure information such as neighbouring relations between different cells. As for table content, every cell value is put into a list in the same row-first order. Figure 3 (a) and (b) demonstrate the storage format of the complex table in Figure 1 (d). The original table can be recovered based on the layout matrix and the cell value list. Applicable to 4 table types, our storage method does not waste extra space for merged cells and is also convenient to annotate header cells and answer cells.

3.3 Cell Type Annotation

In our problem setting, models are required to understand implicit and multi-type table structures and recognize functional roles of table cells, i.e., conducting cell type classification (CTC) task.

[4]http://eid.csrc.gov.cn/101111/index.html
[5]https://baike.baidu.com/

There are different taxonomies of cell types which focus on hierarchical tables and cannot be directly applied to complex tables (Ghasemi Gol et al., 2019; Dong et al., 2019; Zhang et al., 2021; Dong et al., 2020). To model different table structures, we design a new taxonomy of cell types with the concentration on header cells that are useful for TQA models to find correct answers. Specifically, table cells are categorized into 5 types based on their functional roles.

Row Attribute and Column Attribute Row attribute and column attribute describes table cells in the same row and in the same column respectively, e.g., yellow cells and red cells in Figure 1. Attribute cells are only used to describe other cells and they themselves are not data.

Row Index and Column Index Row index and column index are individual cells that can be used to index data records in the row or column orientation, e.g., blue cells and green cells in Figure 1. Index cells are also meaningful data. For instance, in vertical tables, cells in the primary key column are indices of each row.

Pure Data Pure data cells are the core body of a table. They do not have the function of describing or indexing other cells and their meanings should be understood with the help of header cells.

We instructed annotators in distinguishing five cell types and asked them to annotate the cell ID lists of four types of header cells, as shown in Figure 3 (c). The rest of table cells are deemed pure data cells.

3.4 QA Pairs Construction

After identifying header cells, we asked annotators to raise questions about data cells and label answer cell IDs, as illustrated in Figure 3 (d). Note that attribute cells are not allowed to be answers as we do not consider them as data cells. To avoid the annotation artifacts from the homogeneous patterns of questions, e.g., always using the same question expression, annotators were asked to use diverse language expressions to raise questions and answers' locations should change frequently. Annotators were also encouraged to paraphrase questions to increase difficulty.

Questions about table are broadly classified into two types: *Lookup* and *Aggregation* (Glass et al., 2021). *Lookup* questions require selecting table cells as answers whereas *Aggregation* questions are answered by performing an arithmetic operation over a subset of table cells, such as *Sum()*. In this paper, our primary focus is on *Lookup* questions. Besides single-cell answer, annotators were also allowed to select one row or one column or arbitrary table cells as answer. Figure 4 shows the distribution of question types. Four questions were raised for each vertical and horizontal table, and five questions for every hierarchical and complex table.

3.5 Data Review and Checking

In order to guarantee annotation quality, in each annotation step, all annotators conducted a trial annotation with 50 samples and the results were checked. Feedback was given to corresponding annotators until they fully comprehended the annotation requirements. After all annotation tasks finished, we and two most experienced annotators performed the final review to fix labeling errors. We inspected annotations for the correctness of cell type annotations and answer cell annotations. We also inspected the grammar and wording and filtered questions with obvious mistakes. We checked for offensive content and identifiers, and replaced identifying information with mono-directional hashes. Finally, we preserve 1,200 tables and 5,000 questions.

3.6 Dataset Analysis and Comparison

Table 1 shows a comprehensive comparison of IM-TQA to related TQA datasets. The advantages of the proposed dataset are as follows: (1) It is the first TQA dataset that contains implicit and multi-type table structures, especially complex tables ignored by former datasets. Though AIT-QA (Katsis et al., 2022) and HiTab (Cheng et al., 2022) include vertical and hierarchical tables, they ignore the other two table types. (2) It is annotated with cell functional roles and QA pairs, which supports both CTC and TQA task. Diverse table structures in our dataset challenge existing CTC and TQA models. (3) Compared with single-domain datasets like AIT-QA and WTQ, IM-TQA includes tables from various domains.

We split dataset based on table structures. If header cell locations of two tables are exactly same, we consider they share the same table structure. Table structures in the resulted train, valid and test split do not overlap with each other. Questions of the same table are assigned to its corresponding split. Table 2 shows the table number and the question number of each split. More basic statistics are shown in Appendix A.1

Dataset	Language	Table source	#Tables	#Questions	Avg. Q len	Implicit	Table type				Task	
							Ver	Hor	Hie	Com	CTC	TQA
NL2SQL (Sun et al., 2020)	Chinese	Reports, Spreadsheets	6,029	64,891	11	-	✓	-	-	-	-	✓
Cspider (Min et al., 2019)	Chinese	Spider	876	9,691	11.9	-	✓	-	-	-	-	✓
DuSQL (Wang et al., 2020b)	Chinese	Baidu Baike, Reports, Forums	813	28,762	19.3	-	✓	-	-	-	-	✓
CHASE (Guo et al., 2021)	Chinese	DuSQL, SParC	1,280	17,940	13	-	✓	-	-	-	-	✓
WTQ (Pasupat and Liang, 2015)	English	Wikipedia	2,108	22,033	10	-	✓	-	-	-	-	✓
WikiSQL (Zhong et al., 2017)	English	Wikipedia	26,521	80,654	11.7	-	✓	-	-	-	-	✓
Spider (Yu et al., 2018)	English	College data,WikiSQL	1,020	10,181	13.2	-	✓	-	-	-	-	✓
ToTTo (Parikh et al., 2020)	English	Wikipedia	83,141	-	-	-	✓	-	✓	-	-	-
AIT-QA (Katsis et al., 2022)	English	Annual reports	116	515	12.9	-	✓	-	✓	-	-	✓
HiTab (Cheng et al., 2022)	English	Stat. reports, Wiki	3,597	10,672	16.5	-	✓	-	✓	-	-	✓
IM-TQA(ours)	**Chinese**	**Reports, Baidu Encyclopedia, Specification, Record Sheets**	**1200**	**5000**	**13.1**	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison of IM-TQA to related TQA datasets. Implicit represents implicit table structures. Ver, Hor, Hie and Com denotes vertical, horizontal, hierarchical and complex table respectively.

	Train	Valid	Split	Total
# table structures	466	40	80	586
# tables	936	111	153	1,200
# questions	3,909	464	627	5,000
# vertical tables	224	31	45	300
# horizontal tables	230	34	36	300
# hierarchical tables	231	35	34	300
# complex tables	251	11	38	300

Table 2: Dataset split statistics

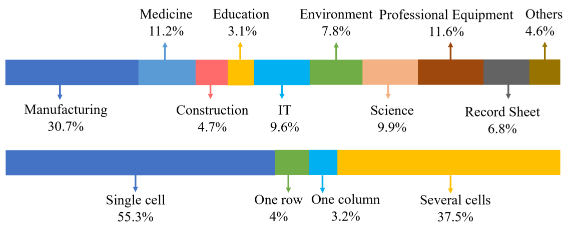


Figure 4: Distribution of domains and question types.

4 Method

We proposed the RGCN-RCI framework for IM-TQA, shown in Figure 5. RGCN-RCI consists of an RGCN-based Cell Type Classification (CTC) module and an RCI-based table question answering (TQA) module. CTC module uses an RGCN to aggregate neighbour cell information and predict cell's functional role. Based on the predictions of CTC module, TQA module adopts an improved RCI model to predict whether a row or column contains the answer or not. The final answer cells are selected from intersections of positive rows and positive columns.

4.1 Cell Type Classification Module

We convert the table into a graph, where nodes are table cells, and neighbouring relations between cells are edges. The resulted graph is processed by an RGCN to predict cell types.

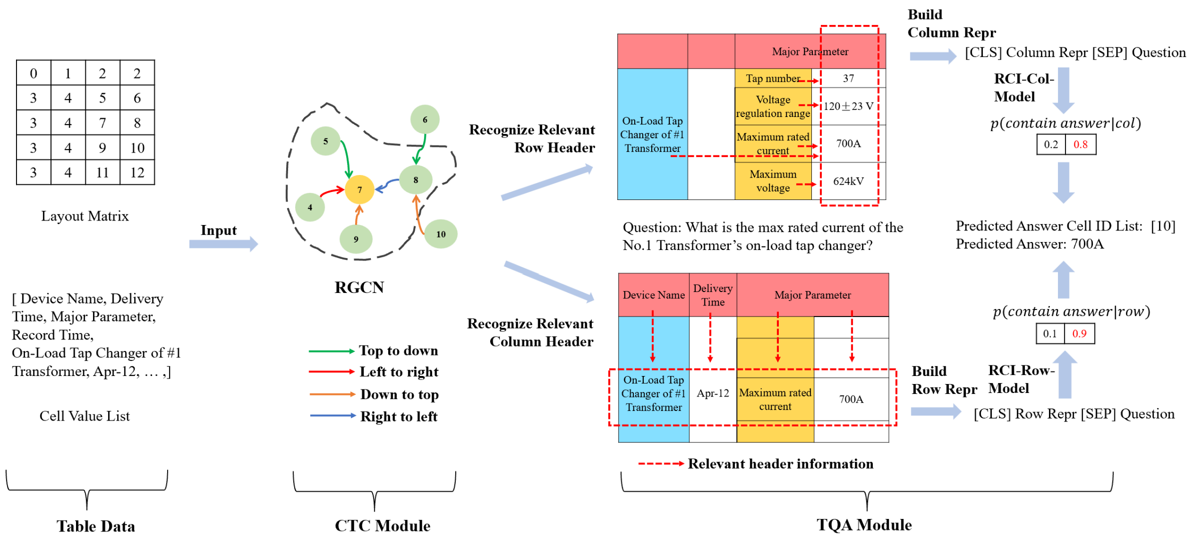
Cell Features The initial node representation vector consists of two parts: hand-crafted feature and semantic feature extracted from the pre-trained language model. We select 24 available manual features from Koci et al. (2016) such as the cell text length (listed in Appendix A.4). The 24-dim integer vectors are transformed into 32-dim continuous numerical vectors by a trained auto encoder. We input cell text to the pre-trained BERT (Devlin et al., 2019) and take the 768-dim output vector of the special [CLS] token as semantic feature of the whole cell. In the end, hand-crafted feature and semantic feature are concatenated to produce 800-dim initial node representation.

Edges We design four directed edges which point from each cell to its surrounding neighbour cells: *top to down*, *left to right*, *down to top*, *right to left*. We argue that neighbour cell information is important for predicting cell functional roles. For example, most data cells are surrounded by other data cells. As the table is converted to a heterogeneous graph, a relational graph convolutional network (RGCN) (Schlichtkrull et al., 2017) is used to aggregate neighbour cell information in different orientations and update node representations. The updated node representations are input to the final linear layer to predict cell types.

4.2 Table Question Answering Module

RCI (Glass et al., 2021) is a state-of-the-art TQA model for *Lookup* questions. It concatenates a text sequence representation of each row (or column) to the question text, and uses a pre-trained language model like ALBERT (Lan et al., 2020) to predict whether the row/column contains the answer or not. Cells on the intersection of positive rows and positive columns are final answers. When constructing the textual representation of each row (or column), RCI incorporates the structure of vertical table. The row textual representation is the concatenation of "column header : cell text", and the column textual representation is the concatenation of column header and all cell texts in this column.

However, this representation method does not fit other types of tables. Due to the inability to understand implicit and diverse table structures, this method may include irrelevant headers or miss



Layout Matrix
Cell Value List
[Device Name, Delivery Time, Major Parameter, Record Time, On-Load Tap Changer of #1 Transformer, Apr-12, … ,]
Input
RGCN
Top to down
Left to right
Down to top
Right to left
Recognize Relevant Row Header
Recognize Relevant Column Header
Question: What is the max rated current of the No.1 Transformer's on-load tap changer?
Build Column Repr
[CLS] Column Repr [SEP] Question
RCI-Col-Model
p(contain answer|col)
Predicted Answer Cell ID List: [10]
Predicted Answer: 700A
p(contain answer|row)
Build Row Repr
RCI-Row-Model
[CLS] Row Repr [SEP] Question
Relevant header information
Table Data
CTC Module
TQA Module

Figure 5: Overview of the proposed framework, with the table of Figure 1 (d) as a running example.

useful header information. For example, when constructing the representation of the third row in Figure 1 (b), RCI treats all cells in the first row as column headers and gives wrong textual representation:

> Name : Listed height | Tim Duncan : 6 ft 11 in (2.11 m) |

where ":" is a delimiter token between column header and cell text, and "|" is a delimiter token between different cells. Any distinctive tokens in the model vocabulary can serve as delimiters. When building the representation of the fourth column in Figure 1 (d), it will lose relevant headers in the third column and output representation without necessary information:

> Major Parameter | 37 | 120±23 V | 700A | 624kV |

To overcome the defect of RCI's original textual representation method, we propose a new representation method with the help of predicted cell types from CTC module and table layout matrix. Specifically, when constructing row textual representation, we locate the nearest column attribute and column index for every data cell in this row. These column headers are regarded as relevant headers and will be concatenated with the corresponding data cell. Similarly, when constructing column textual representation, the nearest row attribute and row index will be concatenated with the corresponding data cell in this column. Again, let's take the third row in Figure 1 (b) as an example. Based on CTC module's predicted results and layout matrix, "Tim Duncan" is the nearest column index to the data cell "6 ft 11 in (2.11 m)", and "Name" is a row attribute irrelevant to cells in the third row. The new row textual representation is:

> Listed height | Tim Duncan : 6 ft 11 in (2.11 m) |

For the fourth column in Figure 1 (d), its new column representation is:

> Major Parameter | On-Load Tap Changer of #1 Transformer - Tap number : 37 | On-Load Tap Changer of #1 Transformer - Voltage regulation range : 120±23 V |. . . |

where "-" is a delimiter token between index cell and attribute cell. Compared with original textual representation method, our proposed method helps RCI to exclude irrelevant headers and include useful header, which contributes to final predictions.

5 Experiments

5.1 Cell Type Classification

5.1.1 Experimental Setup

Considered baselines are as follows. **Random Forest** (Koci et al., 2016): A random forest classifier is used to conduct CTC task based on manual features. **CNN-BERT** (Dong et al., 2019): A method using BERT to extract semantic features and a CNN to learn spatial correlations between cells. **Bi-LSTM** (Ghasemi Gol et al., 2019): Two bidirectional LSTM are used to capture dependencies between different cells, one observing the sequence of cells in each row, and the other observing the sequence of cells in each column. **MLP**: Directly applying a two-layer neural network to all cell features to predict cell types without neighbour information. **RAT** (Shaw et al., 2018): The Relation-Aware-Transformer whose self-attention mechanism is extended to consider the edge label between each pair of nodes. In this setting, the table can be seen as a directed complete graph where

Model	All tables							Complex tables							Gap(%)↓
	Per-class F1(%)↑					Macro F1(%)↑	Macro F1-header(%)↑	Per-class F1(%)↑					Macro F1(%)↑	Macro F1-header(%)↑	
	PD	CA	RA	CI	RI			PD	CA	RA	CI	RI			
RF	95.0	80.4	69.4	78.6	65.2	77.7±0.2	73.4	92.0	67.2	64.4	29.8	53.6	61.4±1.4	53.8	16.3
MLP	95.8	79.8	76.2	78.3	72.3	80.5±0.4	76.7	91.8	64.8	80.0	43.8	54.3	66.9±0.5	60.7	13.6
CNN-BERT†	96.6	87.4	84.4	71.0	75.8	83.0±0.8	79.7	93.8	81.0	86.6	33.4	58.8	70.7±1.2	65.0	12.3
Bi-LSTM	97.2	91.4	87.4	79.0	79.4	86.9±0.3	84.3	93.8	86.6	90.2	60.4	72.2	80.6±0.7	77.4	6.30
RAT	96.0	85.3	82.5	82.0	78.3	84.8±0.3	82.0	92.3	73.0	85.0	69.5	61.0	76.2±0.5	72.1	8.60
RGCN **(ours)**	**98.8**	**92.4**	**89.6**	**85.6**	**85.4**	**90.4±0.5**	**88.3**	**97.2**	**89.4**	**93.0**	**69.6**	**79.4**	**85.7±0.6**	**82.9**	**4.70**

Table 3: CTC results on all tables and complex tables. PD, CA, RA, CI and RI are acronyms of five cell types, e.g., PD denotes Pure Data. † represents our implementation. RF stands for random forest. ± stands for standard deviation over 5 repeated experiments. CTC results on vertical, horizontal and hierarchical tables are shown in Table 9, Table 10 and Table 11 in Appendix B.1 due to space limitation.

Model	Exact Match Acc(%)				
	All	Ver	Hor	Hie	Com
RAT†	18.5±0.9	34.5	33.6	5.03	4.07
TAPAS	33.2±0.8	58.0	31.1	26.4	15.7
RCI	47.2±0.3	68.4	45.1	56.0	19.2
RCI-AIT	49.6±0.4	69.5	43.4	60.4	23.8
RGCN-RCI (ours)	53.4±0.4	70.7	45.9	62.9	32.0
RGCN-RCI (ours) + Oracle headers	**55.3±0.3**	**73.0**	**46.7**	**66.7**	**33.1**

Table 4: TQA results on our dataset. † represents our implementation. Ver, Hor, Hie and Com denotes four table types, respectively. ± stands for standard deviation over 5 repeated experiments.

each table cell can communicate with all the other cells. We replace the RGCN with RAT to update node representations. Implementation details can be found in Appendix A.2.

5.1.2 Experimental Results

Table 3 shows CTC results on all tables and complex tables, averaged over 5 repeated experiments. Models are evaluated with F1 score. Macro F1-header represents the macro F1 on four header types. Gap represents the Macro F1 difference between all tables and complex tables.

From the results shown in Table 3, we can observe that: (1) RGCN achieves the best macro F1 on all tables and complex tables, beating the best-performing baseline Bi-LSTM by 3.5% and 5.1% respectively, which demonstrates the effectiveness of our proposed graph-based CTC model over various table types, especially complex table. (2) CNN-BERT, Bi-LSTM, RAT and RGCN aggregates information from different neighbour cells. The best performance of RGCN indicates that neighbour information in a local area is most important for CTC task. This is consistent with the intuition that human can determine cell types based on a small part of the table and do not need to read the whole table. (3) Performance on complex tables is worse than overall performance, showing that CTC models struggle in comprehending diverse and complex table structures with flexible header locations.

5.2 Table Question Answering

5.2.1 Experimental Setup

We include following TQA baselines. **RCI** (Glass et al., 2021): A state-of-the-art TQA model with its original textual representation method. **RCI-AIT** (Katsis et al., 2022): A variant of RCI for AIT-QA dataset with the textual representation method designed for hierarchical tables. **TAPAS** (Herzig et al., 2020): A table pre-training based TQA model which takes the linearization of question and table cells as input and outputs token representations. We pre-train a TAPAS model on 6 million Chinese tables collected from Baidu Encyclopedia. In fine-tuning stage, we concatenate output representations of the first character and last character in the cell text as cell representation and use a linear layer to predict the probability of selecting the cell as answer. **RAT**: Table cells and question are converted to a graph. We follow Mueller et al. (2019) to use cell-level relations such as question-node to column-header, and use RAT to update cell representations. The final cell representations are used to predict answers. Implementation details are shown in Appendix A.3.

5.2.2 Experimental Results

Table 4 shows performance on IM-TQA dataset, averaged over 5 repeated experiments. We evaluate model's performance with exact match accuracy. "RGCN-RCI" denotes our proposed framework. "RGCN-RCI + Oracle headers" represents we use annotated headers to build new textual representation of each row or column.

From the results shown in Table 4, we can find that: (1) Existing TQA models struggle on our dataset. For example, the best baseline RCI-AIT achieves an overall accuracy of 49.6%, and its accuracy on complex tables is only 23.8%. This shows that previous TQA models cannot achieve a great generalization on multi-type tables, especially complex tables whose structure is most complicated. (2) Compared with baselines which lack the ability to comprehend diverse table structures, RGCN-RCI framework achieves better performance and improves the accuracy of RCI-AIT on all tables and complex tables by 3.8% and 8.2% respectively, which demonstrates the effectiveness of our proposed textual representation method. RGCN-RCI + Oracle headers can achieve more performance boost. This further proves that table structure understanding is beneficial to question answering over implicit and multi-type tables. A case study is shown in Appendix B.3. (3) As a table pre-training model, TAPAS still cannot generalize well on all table types. We suppose that this is because TAPAS is originally designed and pre-trained on vertical tables. Considering more diverse table types in pre-training stage may improve its performance on multi-type tables. (4) Even with an oracle providing correct headers, RGCN-RCI only achieves 33.1% accuracy on complex tables. This shows that there is a long way to go for TQA models to find correct answers in complicated table structures.

We randomly select 100 error cases of RGCN-RCI framework to conduct error analysis. Error cases fall into four categories: (1) Row mistakes(40%): model fails to select the correct row which contains answer. (2) Column mistakes(28%): model fails to select the correct column. This shows that predicting the correct column is usually easier than predicting the correct row. (3) Row and column mistakes(14%): model selects wrong row and wrong column at the same time, which is common in complex tables(57%). (4) Missing rows or columns(18%): model mistakenly predicted that none of rows or columns contains answer, which often results from the paraphrased question expression such as synonyms.

6 Related Work

Table QA methods can be categorized into two types: semantic-parsing-based methods and non-semantic-parsing methods. The semantic-parsing-based methods first transform the natural language question into a logical form such as SQL and then execute the logical form against tables to output the final answer (Wang et al., 2020a; Hui et al., 2022; Guo et al., 2019). Non-semantic-parsing methods often adopt heuristic designs to extract final answer from tables without generating logical forms (Yang et al., 2022; Zayats et al., 2021; Glass et al., 2021). Researchers also proposed various TQA datasets (Zhong et al., 2017; Iyyer et al., 2017; Chen et al., 2020). Most of previous datasets consist of vertical tables with regular structure except HiTab (Cheng et al., 2022) and AIT-QA (Katsis et al., 2022), which also considers hierarchical tables. By contrast, our proposed IM-TQA benchmark is the first TQA dataset that include tables with implicit and multi-type structures.

Cell type classification is a crucial task for table structure understanding, which aims at recognizing table cells' functional roles. Previous work proposed different taxonomies of cell types, which mainly focus on spreadsheet tables (Dong et al., 2019; Zhang et al., 2021; Koci et al., 2016). Coarse-grained taxonomies classify table cells according to their roles in basic table structure (Sun et al., 2021), e.g., header, attribute, metadata, data. Fine-grained taxonomies will subdivide header cells or data cells according to more specific functions (Zhang et al., 2021), e.g., dividing headers into value name, index, index name, aggregation and others. Our proposed cell type taxonomy belongs to coarse-grained taxonomies and we focus on header cells that are useful for TQA models to locate answers.

7 Limitations

Our proposed dataset is in Chinese. We plan to construct the corresponding English dataset and enlarge the dataset scale in future work. As an exploration of table question answering over implicit and multi-type structures, this paper focuses on *Lookup* questions and we leave annotations of *Aggregation* questions in the future work. In addition, our proposed RGCN-RCI framework is not an end-to-end model. We look forward to seeing more end-to-end models which can simultaneously learn table structures and question answering with the help of our dataset.

8 Conclusion

We propose a new problem, table question answering over implicit and multi-type table structures, and construct the corresponding dataset named IM-TQA. Our dataset includes tables of four types, which are collected from various domains. Besides QA pairs, we also annotate functional roles of table cells to promote understanding of implicit table structures. In experiments, we benchmark recent methods on CTC and TQA tasks on our dataset, and propose a framework that outperforms existing methods. Experimental results shows that IM-TQA can provide a challenging and comprehensive testbed for future research.

9 Ethical Considerations

Our proposed benchmark is a free and open resource for the community to study table question answering over implicit and multi-type tables. Tables are collected from the public organization China Securities Regulatory Commission and Baidu Encyclopedia, which allow sharing and redistribution for non-commercial use. Tables in these web pages are open to public so there is no privacy risk. In the annotation process, we asked annotators to check if there exist any offensive content such as insulting or discriminatory speech. They did not find any such content in our benchmark. We also checked for identifiers and replaced identifying information with mono-directional hashes. We recruit 10 professional annotators and pay them at a price of 6.5 dollars per hour (above the average local payment of similar jobs). The total time cost for annotation is 1,200 working hours. Annotators were informed that these labeled data would be used as a table question answering dataset. Main experiments in this paper can be run on a single NVIDIA GeForce RTX 3090 GPU. We will release our benchmark and code for future research along with the paper. Our dataset follows the Computational Use of Data Agreement v1.0.

References

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, Online. Association for Computational Linguistics.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Haoyu Dong, Shijie Liu, Zhouyu Fu, Shi Han, and Dongmei Zhang. 2019. Semantic structure extraction for spreadsheet tables with a multi-task learning architecture. In *Workshop on Document Intelligence at NeurIPS 2019*.

Haoyu Dong, Jiong Yang, Shi Han, and Dongmei Zhang. 2020. Learning formatting style transfer and structure extraction for spreadsheet tables with a hybrid neural network architecture. In *Proceedings of the 29th ACM International Conference on Information amp; Knowledge Management*, CIKM '20, page 2389–2396, New York, NY, USA. Association for Computing Machinery.

Majid Ghasemi Gol, Jay Pujara, and Pedro Szekely. 2019. Tabular cell classification using pre-trained cell embeddings. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 230–239.

Majid Ghasemi-Gol and Pedro Szekely. 2018. Tabvec: Table vectors for classification of web tables.

Michael Glass, Mustafa Canim, Alfio Gliozzo, Saneem Chemmengath, Vishwajeet Kumar, Rishav Chakravarti, Avi Sil, Feifei Pan, Samarth Bharadwaj, and Nicolas Rodolfo Fauceglia. 2021. Capturing row and column semantics in transformer based question answering over tables.

Jiaqi Guo, Ziliang Si, Yu Wang, Qian Liu, Ming Fan, Jian-Guang Lou, Zijiang Yang, and Ting Liu. 2021. Chase: A large-scale and pragmatic Chinese dataset for cross-database context-dependent text-to-SQL. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2316–2331, Online. Association for Computational Linguistics.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Yanyang Li, Bowen Li, Jian Sun, and Yongbin Li. 2022. S^2SQL: Injecting syntax to question-schema interaction graph encoder for text-to-SQL parsers. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1254–1262, Dublin, Ireland. Association for Computational Linguistics.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics.

Yannis Katsis, Saneem Chemmengath, Vishwajeet Kumar, Samarth Bharadwaj, Mustafa Canim, Michael Glass, Alfio Gliozzo, Feifei Pan, Jaydeep Sen, Karthik Sankaranarayanan, and Soumen Chakrabarti. 2022. AIT-QA: Question answering dataset over complex tables in the airline industry. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 305–314, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Elvis Koci, Maik Thiele, Oscar Romero, and Wolfgang Lehner. 2016. A machine learning approach for layout inference in spreadsheets. IC3K 2016, page 77–88, Setubal, PRT. SCITEPRESS - Science and Technology Publications, Lda.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Qingkai Min, Yuefeng Shi, and Yue Zhang. 2019. A pilot study for chinese SQL semantic parsing. *CoRR*, abs/1909.13293.

Thomas Mueller, Francesco Piccinno, Peter Shaw, Massimo Nicosia, and Yasemin Altun. 2019. Answering conversational questions on structured data without logical forms. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5902–5910, Hong Kong, China. Association for Computational Linguistics.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Kexuan Sun, Harsha Rayudu, and Jay Pujara. 2021. A hybrid probabilistic approach for table understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4366–4374.

Ningyuan Sun, Xuefeng Yang, and Yunfeng Liu. 2020. Tableqa: a large-scale chinese text-to-sql dataset for table-aware sql generation.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020a. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Lijie Wang, Ao Zhang, Kun Wu, Ke Sun, Zhenghua Li, Hua Wu, Min Zhang, and Haifeng Wang. 2020b. DuSQL: A large-scale and pragmatic Chinese text-to-SQL dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6923–6935, Online. Association for Computational Linguistics.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery amp; Data Mining*, KDD '21, page 1780–1790, New York, NY, USA. Association for Computing Machinery.

Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. Tableformer: Robust transformer modeling for table-text encoding.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018*

Conference on Empirical Methods in Natural Language Processing, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Vicky Zayats, Kristina Toutanova, and Mari Ostendorf. 2021. Representations for question answering from documents with tables and text. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2895–2906, Online. Association for Computational Linguistics.

Yakun Zhang, Xiao Lv, Haoyu Dong, Wensheng Dou, Shi Han, Dongmei Zhang, Jun Wei, and Dan Ye. 2021. Semantic table structure identification in spreadsheets. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2021, page 283–295, New York, NY, USA. Association for Computing Machinery.

Yanzhao Zheng, Haibin Wang, Baohua Dong, Xingjun Wang, and Changshan Li. 2022. HIE-SQL: History information enhanced network for context-dependent text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2997–3007, Dublin, Ireland. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

A Dataset Statistics and Implementation Details

A.1 Dataset Statistics

Table 5 shows IM-TQA's basic statistical information such as average question number, row number, column number per table. We count the number of distinct table structures. If header cell locations of two tables are exactly same, we consider they share the same table structure. As expected, structure number of complex tables (241) and hierarchical tables (153) are larger than that of vertical tables (103) and horizontal tables (89). Table 6 shows number of different table headers.

Characteristic	Value
Avg. question number per table	4.1
Row number per table (median/mean)	6/7.3
Column number per table (median/mean)	5/5.2
Cell number per table (median/mean)	23/31.6
Header cell number per table (median/mean)	10/12.7
Avg. answer num per question	1.68

Table 5: Dataset basic statistics

	Ver	Hor	Hie	Com	Total
# table structures	103	89	153	241	586
# Column Attribute	1,158	99	2,185	1,434	4,876
# Row Attribute	145	1,445	831	3,649	6,070
# Column Index	0	1,092	16	268	1,376
# Row Index	1,355	0	872	781	3,008

Table 6: Different header number.

A.2 Implementation Details for CTC Experiments

In this paper, we use PaddlePaddle[6] to implement our model. We use valid set for model selection and hyper-parameter tuning, and then evaluate the best model on test set. We use bert-base-chinese to extract cell semantic features. For the RGCN model, the GNN layer number is 4, the dimension of hidden layers is 800, and the ReLu activation is used between adjacent GNN layers. We train RGCN for 20 epochs. The Adam optimizer is adopted with learning rate of 1e-4. To mitigate the class-imbalance in the CTC task, we follow Ghasemi Gol et al. (2019) and use a weighted *Cross Entropy Loss* as our loss function. We set the weight w_k of cell type k to be inversely proportional to the number of cells with class type k in train set n_k^{train}, i.e., $w_k = 1 - \frac{n_k^{train}}{\sum_{k'=1}^{5} n_{k'}^{train}}$.

[6]https://www.paddlepaddle.org.cn/

Model	Number of Parameters	Training Time
Pretrained BERT	110M	-
Auto Encoder	0.01M	10.5 minutes
Random Forest	n_estimators=100	2 minutes
MLP	6M	1.5 minutes
CNN-BERT	0.1M	3 minutes
Bi-LSTM	1.5M	10 minutes
RAT	80M	1 hour
RGCN	30M	7 minutes
TAPAS	110 M	pre-training: 1 week finetune: 1 hour
RCI	110 M×2	1 hour

Table 7: Model parameter number and training time.

For RAT baseline, we follow the implementation in Wang et al. (2020a). The RAT layer number and the attention head number is 8. The probability for dropout layers is set to 0.1. The AdamW optimizer is adopted with learning rate of 2e-5, warmup fraction of 0.1 and weight decay of 0.01. We train RAT for 50 epochs with batch size of 16. For MLP baseline, we adopt a multilayer perceptron of one hidden layer with hidden size 800 and ReLu activation. For CNN-BERT baseline, we adopt a text CNN (Kim, 2014) on cell feature matrix along the row direction. For other baselines, we follow the original experimental setup in their papers. PSL^{RF} (Sun et al., 2021) is a very recent CTC model, but the code has not been publicly available. Thus, we do not compare with this method.

A.3 Implementation Details for TQA Experiments

We follow the original paper to train RCI model with our proposed textual representation method. We use bert-base-chinese as the sequence-pair classifier, which is trained for 3 epochs with batch size of 64. The AdamW optimizer is adopted with learning rate of 2e-5, warmup fraction of 0.1 and weight decay of 0.01. The probability of dropout layers is set to 0.1. The max grad norm is set to 1.0. To achieve better performance in multi-turn TQA task, Mueller et al. (2019) also introduced a node representing the answer of last question. We do not use these unnecessary designs. All model parameter number and training time are listed in Table 7. Main experiments in this paper can be run on a single NVIDIA GeForce RTX 3090 GPU. The pretraining of TAPAS was ran on four Tesla V100 GPU for about one week. Note that out of resource limitation, we only pretrain TAPAS for once.

A.4 Complete Manual Features

We list all manual features in Table 8. We consider 15 content features and 9 spatial features. We follow Ghasemi Gol et al. (2019) and train an auto encoder to transform 24-dim integer vectors into 32-dim continuous numerical vectors. We train the auto encoder by an vector reconstruct task. The auto encoder tries to reconstruct the input integer vector at its output, and generates continuous vector representations at the output of the encoder layer. Mean square error is used as loss function. At test time, we feed the 24-dim integer manual vectors into the trained auto encoder and gain the 32-dim continuous vector from the encoder output.

Manual Cell Features
LENGTH#(character-level)
NUM OF TOKENS#(word-level)
LEADING SPACES#
IS NUMERIC?
STARTS WITH NUMBER?
STARTS WITH SPECIAL?
IS CAPITALIZED?
IS UPPER CASE?
IS ALPHABETIC?
CONTAINS SPECIAL CHARS?
CONTAINS PUNCTUATIONS?
CONTAINS COLON?
WORDS LIKE TOTAL?
WORDS LIKE TABLE?
IN YEAR RANGE?
ROW NUMBER#
COL NUMBER#
NEIGHBOUR CELL NUM#
HAS 0 NEIGHBORS?
HAS 1 NEIGHBORS?
HAS 2 NEIGHBORS?
HAS 3 NEIGHBORS?
HAS 4 NEIGHBORS?
IS MERGED CELL?

Table 8: Manual Feature List

B More Experimental Analysis

B.1 More CTC Experimental Analysis

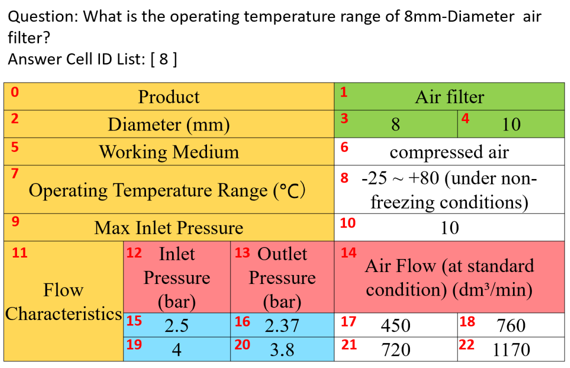
(4) The performance of MLP baseline is the worst among methods based on neural networks, which proves that neighbour information is essential for distinguishing divergent types of cells. (5) F1 scores on the column index (CI) and row index (RI) are lower than that on the column attribute (CA) and row attribute (RA). This indicates that, compared with attribute cells that are not data, it is more difficult for CTC models to distinguish index cells from pure data cells. CTC results on vertical, horizontal and hierarchical tables are shown in Table 9, Table 10 and Table 11, which demonstrates

that our graph-based method achieves a better generalization on tables of diffierent types.

B.2 More TQA Experimental Analysis

(5) RCI-AIT achieves better performance than RCI especially on hierarchical tables, which shows the effectiveness of special textual representation method designed for hierarchical tables. However, it cannot help RCI-AIT to generalize well on multi-type tables as it essentially relies on explicit table structures with header annotations and are unable to comprehend unknown table structures. (6) The performance of RAT is the worst among baselines. We suppose the reason is that RAT adopts cell-level relations for vertical tables (Mueller et al., 2019) and is also not pre-trained. (7) As a non-table-pretrained model, RCI beats the TAPAS on all model types, which is consistent with the results in Glass et al. (2021). This shows that selecting answer cells based on column and row information is better than the way that directly predicts whether a table cell is the answer or not.

B.3 TQA Case Study


Question: What is the operating temperature range of 8mm-Diameter air filter?
Answer Cell ID List: [8]

0 Product			1 Air filter	
2 Diameter (mm)			3 8	4 10
5 Working Medium			6 compressed air	
7 Operating Temperature Range (°C)			8 -25 ~ +80 (under non-freezing conditions)	
9 Max Inlet Pressure			10 10	
11 Flow Characteristics	12 Inlet Pressure (bar)	13 Outlet Pressure (bar)	14 Air Flow (at standard condition) (dm^3/min)	
	15 2.5	16 2.37	17 450	18 760
	19 4	20 3.8	21 720	22 1170

Figure 6: A real case (translated to English) where RCI fails but RGCN-RCI gives the correct answer.

Figure 6 shows an QA sample on a complex table containing parameters of an air filter. In this case, original RCI cannot correctly determine which column contains the answer as its representation of the fourth column lacks necessary row header information, such as "Operating Temperature Range". By contrast, RGCN-RCI with our new textual representation method could incorporate useful row header information into its column representation and finally locates the correct answer.

C Instructions for Annotators

Screenshots of original instructions for annotators have been saved and shown in this section. The instruction for CTC annotation task is shown in Figure 9. And the instruction for TQA annotation task is shown in Figure 10. A real CTC annotation case of complex table is shown in Figure 11. Besides instructions, we also provided annotators with sufficient QAs to ensure that they fully comprehended the annotation requirements.

Model	Vertical tables						
	per-class F1(%)					Macro F1(%)	Macro F1 -Header(%)
	PD	CA	RA	CI	RI		
RF	93.5	92.3	35.2	-	78.7	74.9	68.7
MLP	98.5	87.5	40.5	-	86.3	78.2	71.4
CNN-BERT	98.0	93.8	56.6	-	87.8	84.1	79.4
Bi-LSTM	98.8	**98.4**	61.8	-	88.8	87.0	83.0
RAT	97.1	95.2	55.3	-	87.4	83.8	79.3
RGCN	**99.3**	96.3	**62.7**	-	**94.6**	**88.2**	**84.5**

Table 9: CTC results on vertical tables.

Model	Horizontal tables						
	per-class F1(%)					Macro F1(%)	Macro F1 -Header(%)
	PD	CA	RA	CI	RI		
RF	94.0	8.20	79.3	90.1	-	67.9	59.2
MLP	95.0	16.0	88.5	81.5	-	70.3	62.0
CNN-BERT	97.2	18.0	94.8	81.4	-	72.9	64.7
Bi-LSTM	98.6	18.8	94.2	79.6	-	72.8	64.2
RAT	96.3	15.0	90.4	88.2	-	72.5	64.5
RGCN	**99.4**	**20.4**	**95.6**	**92.4**	-	**77.0**	**69.5**

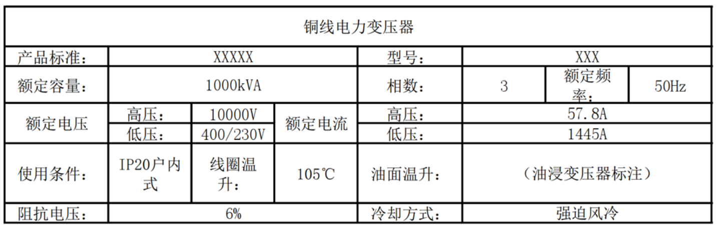
Table 10: CTC results on horizontal tables.

Model	Hierarchical tables						
	per-class F1(%)					Macro F1(%)	Macro F1 -Header(%)
	PD	CA	RA	CI	RI		
RF	93.3	92.6	50.7	-	64.4	75.3	69.2
MLP	98.6	95.1	51.6	-	73.0	79.6	73.2
CNN-BERT	98.8	95.8	55.6	-	75.6	81.4	75.7
Bi-LSTM	98.4	**98.4**	67.6	-	75.2	84.9	80.4
RAT	99.2	98.1	64.5	-	**83.7**	86.4	82.1
RGCN	**99.6**	98.0	**68.4**	-	82.0	**87.0**	**82.8**

Table 11: CTC results on hierarchical tables.

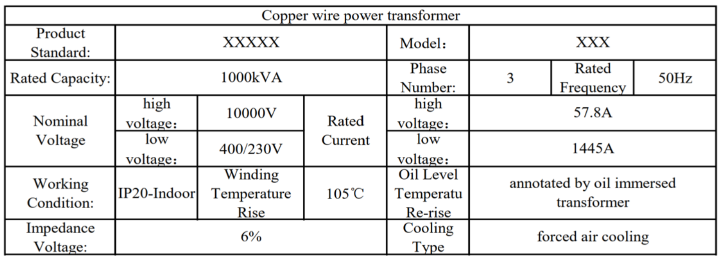
D More Complex Table Examples

Figure 7 and Figure 8 depict two real complex tables in Chinese and we translated them into English for reading convenience. As demonstrated in these examples, header cells in complex table can appear at any position and be mixed with other data cells. Such table structures have not been thoroughly investigated and challenges existing TQA methods.



铜线电力变压器							
产品标准:	XXXXX			型号:	XXX		
额定容量:	1000kVA			相数:	3	额定频率:	50Hz
额定电压	高压:	10000V	额定电流	高压:	57.8A		
	低压:	400/230V		低压:	1445A		
使用条件:	IP20户内式	线圈温升:	105℃	油面温升:	（油浸变压器标注）		
阻抗电压:	6%			冷却方式:	强迫风冷		

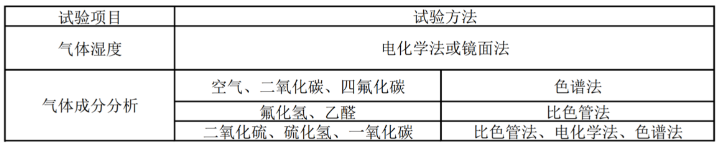
(a) Original Table



Copper wire power transformer							
Product Standard:	XXXXX			Model:	XXX		
Rated Capacity:	1000kVA			Phase Number:	3	Rated Frequency	50Hz
Nominal Voltage	high voltage:	10000V	Rated Current	high voltage:	57.8A		
	low voltage:	400/230V		low voltage:	1445A		
Working Condition:	IP20-Indoor	Winding Temperature Rise	105℃	Oil Level Temperatu Re-rise	annotated by oil immersed transformer		
Impedance Voltage:	6%			Cooling Type	forced air cooling		

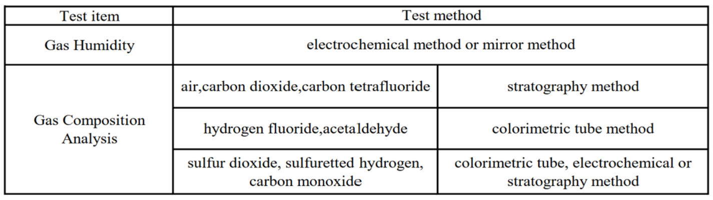
(b) Translated to English

Figure 7: A complex table about an transformer where many row attributes are mixed with data cells.



试验项目	试验方法	
气体湿度	电化学法或镜面法	
气体成分分析	空气、二氧化碳、四氟化碳	色谱法
	氟化氢、乙醛	比色管法
	二氧化硫、硫化氢、一氧化碳	比色管法、电化学法、色谱法

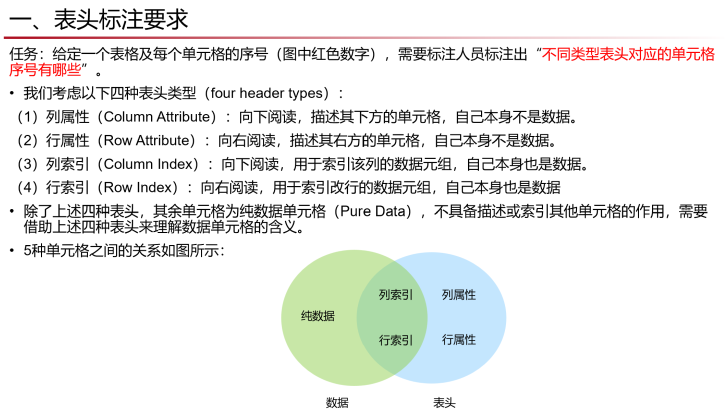
(a) Original Table



Test item	Test method	
Gas Humidity	electrochemical method or mirror method	
Gas Composition Analysis	air,carbon dioxide,carbon tetrafluoride	stratography method
	hydrogen fluoride,acetaldehyde	colorimetric tube method
	sulfur dioxide, sulfuretted hydrogen, carbon monoxide	colorimetric tube, electrochemical or stratography method

(b) Translated to English

Figure 8: A complex table about gas test requirements. A col attribute cell, a pure data cell, and three row indices appear in sequence in the second column, and such mixed data arrangement increases difficulty for TQA models to find the correct answer.

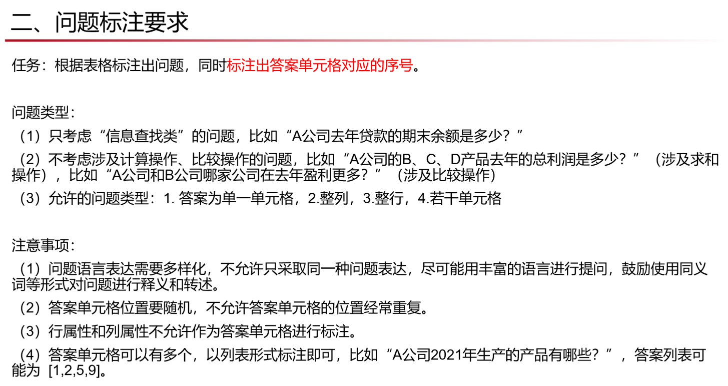


一、表头标注要求

任务：给定一个表格及每个单元格的序号（图中红色数字），需要标注人员标注出“不同类型表头对应的单元格序号有哪些”。

- 我们考虑以下四种表头类型（four header types）：
  （1）列属性（Column Attribute）：向下阅读，描述其下方的单元格，自己本身不是数据。
  （2）行属性（Row Attribute）：向右阅读，描述其右方的单元格，自己本身不是数据。
  （3）列索引（Column Index）：向下阅读，用于索引该列的数据元组，自己本身也是数据。
  （4）行索引（Row Index）：向右阅读，用于索引改行的数据元组，自己本身也是数据
- 除了上述四种表头，其余单元格为纯数据单元格（Pure Data），不具备描述或索引其他单元格的作用，需要借助上述四种表头来理解数据单元格的含义。
- 5种单元格之间的关系如图所示：

Figure 9: Instructions(in screenshot) for CTC annotation tasks.



二、问题标注要求

任务：根据表格标注出问题，同时标注出答案单元格对应的序号。

问题类型：

（1）只考虑“信息查找类”的问题，比如“A公司去年贷款的期末余额是多少？”

（2）不考虑涉及计算操作、比较操作的问题，比如“A公司的B、C、D产品去年的总利润是多少？”（涉及求和操作），比如“A公司和B公司哪家公司在去年盈利更多？”（涉及比较操作）

（3）允许的问题类型：1. 答案为单一单元格，2.整列，3.整行，4.若干单元格

注意事项：

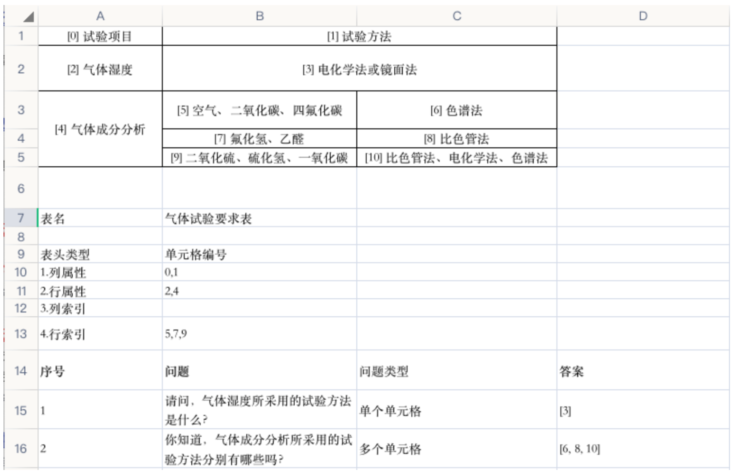
（1）问题语言表达需要多样化，不允许只采取同一种问题表达，尽可能用丰富的语言进行提问，鼓励使用同义词等形式对问题进行释义和转述。

（2）答案单元格位置要随机，不允许答案单元格的位置经常重复。

（3）行属性和列属性不允许作为答案单元格进行标注。

（4）答案单元格可以有多个，以列表形式标注即可，比如“A公司2021年生产的产品有哪些？”，答案列表可能为 [1,2,5,9]。

Figure 10: Instructions(in screenshot) for TQA annotation tasks.



	A	B	C	D
1	[0] 试验项目	[1] 试验方法		
2	[2] 气体湿度	[3] 电化学法或镜面法		
3	[4] 气体成分分析	[5] 空气、二氧化碳、四氟化碳	[6] 色谱法	
4		[7] 氟化氢、乙醛	[8] 比色管法	
5		[9] 二氧化硫、硫化氢、一氧化碳	[10] 比色管法、电化学法、色谱法	
6				
7	表名	气体试验要求表		
8				
9	表头类型	单元格编号		
10	1.列属性	0,1		
11	2.行属性	2,4		
12	3.列索引			
13	4.行索引	5,7,9		
14	序号	问题	问题类型	答案
15	1	请问，气体湿度所采用的试验方法是什么?	单个单元格	[3]
16	2	你知道，气体成分分析所采用的试验方法分别有哪些吗?	多个单元格	[6, 8, 10]

Figure 11: A real annotation case(in screenshot).