

# HeGTa: Leveraging Heterogeneous Graph-enhanced Large Language Models for Few-shot Complex Table Understanding

Rihui Jin<sup>1,2</sup>, Yu Li<sup>1,2</sup>, Guilin Qi<sup>1,7\*</sup>, Nan Hu<sup>1,2</sup>, Yuan-Fang Li<sup>3</sup>, Jiaoyan Chen<sup>4</sup>, Jianan Wang<sup>5</sup>,  
Yongrui Chen<sup>1,2</sup>, Dehai Min<sup>1,2</sup>, Sheng Bi<sup>6</sup>

<sup>1</sup>School of Computer Science and Engineering, Southeast University, China

<sup>2</sup>Key Laboratory of New Generation Artificial Intelligence Technology and its Interdisciplinary Applications (Southeast University), Ministry of Education, China

<sup>3</sup>Monash University <sup>4</sup>The University of Manchester <sup>5</sup>Alibaba Group

<sup>6</sup>Law and Innovation Lab, Law School, Southeast University, China

<sup>7</sup>State Key Laboratory for Novel Software Technology, Nanjing University, China

{ari\_king, yuli\_11, gqi, nanhu, yrchen, zhishanq, bisheng}@seu.edu.cn, yuanfang.li@monash.edu, jiaoyan.chen@manchester.ac.uk, jn\_wang2@outlook.com

## Abstract

Table Understanding (TU) has achieved promising advancements, but it faces the challenges of the scarcity of manually labeled tables and the presence of complex table structures. To address these challenges, we propose HeGTa, a heterogeneous graph (HG)-enhanced large language model (LLM) designed for few-shot TU tasks. This framework aligns structural table semantics with the LLM’s parametric knowledge through soft prompts and instruction tuning. It also addresses complex tables with a multi-task pre-training scheme by incorporating novel multi-granularity self-supervised HG pre-text tasks. We empirically demonstrate the effectiveness of HeGTa, showing that it outperforms the SOTA for few-shot complex TU on several benchmarks.

## 1 Introduction

Table Understanding (TU) seeks to learn informative embeddings of tables containing inherently tabular semantics, often formatted in ways not easily interpretable by machines, as shown in Fig. 1. This endeavor enhances machine performance across a range of table-related tasks, such as Table QA (Wang et al. 2024), Cell Type Classification (Jia et al. 2023) and Table Type Classification (Lu et al. 2024).

Yet, in real-world scenarios, TU faces the challenges of a **lack of sufficient human annotations** and the presence of **complex table structures**, which diminishes the effectiveness and applicability of existing frameworks. As for the first challenge, the data-hungry nature of existing frameworks results in diminished performance in few-shot TU scenarios, where only several samples are annotated. Although some studies (Dong et al. 2022; Herzig et al. 2020) have adopted pre-training with encoder-only architectures (Devlin et al. 2019) to alleviate the annotation burden, these solutions still require considerable amounts of labeled data for task-specific fine-tuning. Regarding the second challenge, while

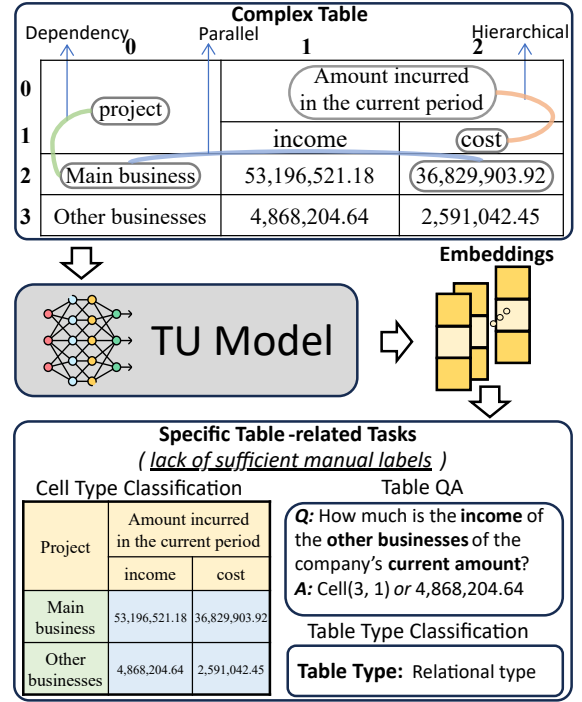


Figure 1: Few-shot complex Table Understanding. Complex tables contain intricate cell-to-cell relationships, including dependency, hierarchical, and parallel ones.

current frameworks attempt to capture structural information in tables through the use of position embeddings (Wang et al. 2020c) or by modeling tables as graphs (Jia et al. 2023; Wang et al. 2021), these methods are effective for simple tables, but wane with complex tables. This shortfall arises because the cell-to-cell relationships in complex tables are more intricate than those in simple tables, as shown in Fig. 1.

Fortunately, recent advancements have introduced techniques that can be integrated into the TU framework to

\*Corresponding author.

tackle the challenges: 1) Large Language Models (LLMs) have shown remarkable effectiveness in managing few-shot tasks involving data in other modalities, such as vision (Liu et al. 2023) and time series (Jin et al. 2023a). This is achieved through instruction tuning and soft prompts, which seek to align the semantic spaces of the other modality encoder with that of the LLM, a method that can also be applied to tabular data. 2) Self-supervised heterogeneous graph (HG) pre-training (Ma, Liu, and Zuo 2023; Yang et al. 2022) enables models to navigate multifaceted relationships within data, such as complex intercell relationships in Fig. 1, using substantial volumes of unlabeled data.

We propose HeGTa, an HG-enhanced LLM framework for the few-shot complex TU. To enable HeGTa to model intricate relationships within complex tables, HeGTa begins with modeling the tabular data with the HG and then processing it through a tabular graph encoder to generate vectors imbued with structural information. Additionally, to ensure that the LLM achieves comparable performance in table tasks to its performance in natural language (NL) tasks, especially in few-shot scenarios, we align the representation spaces of the tabular graph encoder and the LLM via instruction tuning. Specifically, we integrate table vectors containing structural information as soft prompts within the LLM inputs and innovatively design three multi-granularity self-supervised tasks tailored for both tables and LLMs for pre-training. After pre-training specific parameters of HeGTa via these self-supervised tasks, the HeGTa can capitalize on the LLM’s exceptional generalization capability to adapt to downstream tasks with minimal data samples. To validate the performance of HeGTa in few-shot complex TU, we conduct extensive comparative experiments with existing powerful baselines on nine publicly available datasets for three specific table-related tasks in few-shot scenarios. The experimental results show that HeGTa exceeds the current SOTA for the few-shot complex TU across multiple benchmark datasets. In summary, our main contributions are as follows.

- To the best of our knowledge, we are the first to propose a framework to align table semantics with NLs’ to empower the LLM to perform on tables with the same few-shot proficiency as it does with NLs.
- To improve the framework’s ability to complex TU, we propose a refined way to convert tables into HGs and design three novel multi-granularity self-supervised HG pre-training tasks tailored for tabular data and LLMs.
- We conduct extensive experiments on nine publicly available datasets, and the experimental results show that HeGTa exceeds the current SOTA for few-shot complex TU across multiple benchmark datasets.

## 2 Related Work

### 2.1 Graph-based Table Understanding

Many studies (Du et al. 2021; Jin et al. 2023b) have converted tables into graphs and utilized graph encoders to capture tables’ inherent topological information. These frameworks include employing homogeneous graphs or utilizing a basic node-linking strategy that connects cells exclusively to

their adjacent counterparts. Consequently, such frameworks underperform when dealing with complex table structures.

### 2.2 LLM-based Textual Table Understanding

Following the success of LLMs in NL tasks, some efforts (Zhang et al. 2023a; Chen 2022; Sui et al. 2024) have extended their application to table-related tasks. Although these frameworks leverage the exceptional generalization capacity of LLMs to achieve SOTA performance in some few-shot tasks, they resort to simply converting the table into a row-by-row NL format as the input. This process leads to a loss of the tables’ intrinsic topological information.

### 2.3 LLM-based Symbolic Table Understanding

Inspired by Program-of-Thoughts (Chen et al. 2022), several studies (Cao et al. 2023; Cheng et al. 2022; Wang et al. 2024; Zhang et al. 2023b; Ye et al. 2023) have adapted semantic parsing techniques, traditionally applied to relational database tables, to general Table QA tasks. This involves transforming the table into a format interpretable by query languages, such as SQL, and subsequently utilizing LLMs to generate a symbolic program to retrieve the answer. This methodology represents the current SOTA in Table QA. However, its architectural limitations restrict its applicability to other table-related tasks.

### 2.4 Bert-like Encoder-only Table Understanding

Since the rise in popularity of pre-training models like BERT (Devlin et al. 2019), there has been considerable effort devoted to designing specialized encoding methods for tabular data and unique pre-text objectives for pre-training (Yin et al. 2020; Cheng et al. 2021a; Jia et al. 2023). Despite their utilization of self-supervised training, these methods still require a substantial amount of labeled data during fine-tuning for downstream tasks. Additionally, while they incorporate positional embedding into serialized tabular data, they do not effectively capture topological information.

## 3 Task Definition

Given a table  $T = \{c_{i,j} | 0 \leq i < N, 0 \leq j < M\}$  where  $N$  is the number of rows,  $M$  is the number of columns, and  $c_{i,j}$  is the cell located in the  $i^{th}$  row and  $j^{th}$  column. Merged cells, characterized by a row span or column span greater than 1, are prevalent in complex tables. For instance, the cell labeled "project" in the top-left corner of the table shown in Fig. 3 has a row span of 2. We assign the coordinates of such merged cells based on the position of the top-left cell before merging. Hence, the coordinate of the "project" cell is designated as (0,0).

### Specific sub-TU tasks for Evaluation

Cell Type Classification (CTC) involves identifying the type  $y_c$  of each cell  $c_{i,j}$  within a table  $T$ , where  $y_c$  can belong to a basic taxonomy  $Y_c = \{header\ cell, data\ cell\}$  or a more complex one, varying across datasets.

Table Type Classification (TTC) is a table-level categorization task that requires models to classify the table  $T$  according to a specific taxonomy  $Y_t$ .

Table QA (TQA) demands that the model produce an answer  $y_a$  in response to a natural language question  $q$ , with table  $T$  serving as the reference for deriving the answer  $y_a$ .

## 4 METHODOLOGY

This section explains three phases of HeGTa: tabular HG construction and two tuning stages, as shown in Fig. 2.

### 4.1 Tabular Heterogeneous Graph Construction

Given that HGs are more proficient at capturing diverse relationships compared to homogeneous graphs, we employ HGs with heuristic node-linking rules to model the structure of complex tables effectively. The subsequent subsections detail the process of converting tabular data into HGs, including the creation of nodes and the heuristic rules for establishing edges between nodes. The process of creating a tabular HG is shown in Fig. 3.

**Four Tabular Node Types** TABLE, ROW, Header CELL, and Data CELL nodes are denoted as green, red, yellow, and blue nodes in Fig. 3. 1) The TABLE node represents the content described by the table, facilitating the table-level tasks. 2) The ROW node signifies the information contained within a row, aiding in the prediction of the row’s type during self-supervised training. 3) The Header CELL node denotes cells located in the header row, identifying column schemas or categories. 4) The Data CELL node represents cells in data rows, meaning the actual data entries of the table.

**Creating Tabular Nodes** First, CELL nodes are created for each cell in the table, with each node denoted as  $c_{i,j}$ , where  $i$  and  $j$  represent the cell’s coordinates in the original table. If a CELL node is located in the table’s header section or is part of a merged cell spanning the entire table width, it is viewed as a Header Cell; otherwise, it is identified as a Data Cell. The algorithm for determining the header section is elaborated in Appendix. Subsequently, ROW nodes are created to match the number of rows in the table, along with a single TABLE node.

**Initializing Tabular Node Embeddings** An initialization vector is required for each node in the HG. For Header and Data CELL nodes, we employ the output of S-BERT (Reimers and Gurevych 2019) applied to the text within each cell to obtain their initialization vectors. In the case of ROW nodes  $r_i$ , we initialize a vector by concatenating the text from the cells in the  $i^{th}$  row and inputting this concatenated text into the S-BERT. TABLE nodes serve to represent the table’s content itself. In a human’s view, a table’s content can be inferred by examining the cells in the headers. Consequently, we opt to concatenate the text from the cells in the headers and to obtain the embedding.

**Adding Edges** To enhance the machine’s understanding of the table’s semantics, we develop the following heuristic rules to link nodes: 1) A TABLE node, representing the whole table, should be linked to all CELL nodes to encapsulate the global semantics. 2) A ROW node derives information from cells within the same row, so the ROW node  $r_i$

should be linked to each CELL node  $c_{i*}$ . 3) A strong correlation exists between the semantics of Data CELLS and their corresponding Header CELLS within the same column, so they should be linked. 4) Data CELLS located in the same column exhibit a stronger relational bond compared to those in different columns. Consequently, adjacent Data CELLS within a column are interconnected. 5) All Header CELLS should be interconnected because the interpretation of relationships between Data CELLS across columns necessitates the semantic understanding of their respective Header CELLS. For instance, as demonstrated in Fig. 3, determining the relationship between cell  $c_{2,0}$  and cell  $c_{2,1}$  requires an examination of their Header CELLS. This examination reveals that “53,196,521.18” represents the income from the “main business” project.

Edge types are categorized based on the nodes they link, as follows: Table-Header, Table-Data, Header-Row, Data-Row, Header-Data, Data-Data, and Header-Header edges.

### 4.2 Stage 1: Self-supervised Instruction Tuning

Illustrated in Fig. 2, HeGTa utilizes the tabular output of the HG encoder as soft prompts (Li and Liang 2021), which are part of the LLM input. The weights of both modules are tuned through self-supervised instruction tuning to align the vector representation spaces of the two modules. This subsection provides details of the pre-training process, which involves three different granularity self-supervised tasks.

**Tabular HG Encoder** Following the conversion of the table into a tabular HG, HeGTa introduces a heterogeneous GNN (HGNN) (Hu et al. 2020) as the encoder for tabular HG data. The encoder takes a tabular HG as input and generates vector representations for the tabular nodes as output. The HGNN employs a message-passing mechanism to collect semantic and topological information from neighboring cell nodes within each layer, considering various edge types.

#### Multi-granularity Self-supervised Instruction Tasks

Drawing inspiration from the methodologies in LLaVA (Liu et al. 2023), Time-LLM (Jin et al. 2023a), and GraphGPT (Tang et al. 2023), we develop a method to effectively align the vector spaces of two distinct modalities of data: tables and NLs. We introduce the soft prompt technique as a bridge between the two encoders and focus on fine-tuning the tabular graph encoder and the LoRA (Hu et al. 2021) module within the LLM. This process, designed to be lightweight, enables the LLM to grasp the topological information of tables through semantic instruction tuning.

Instruction tuning (Wang et al. 2022; Ouyang et al. 2022), a technique that merges fine-tuning with prompt learning, significantly enhances the generalization capabilities of LLMs. Therefore, before the model-tuning process, we pre-format the self-supervised training data into an instruction-based QA format. Examples of training data for instruction tuning are illustrated in Fig. 4.

To align the LLM with the HG encoder, the HeGTa incorporates additional tokens into the vocabulary: `<tabular_node>`, `<table_start>` and `<table_end>`. The token `<tabular_node>` serves as a placeholder for table tasks within the instruction text, allowing for the substitution with

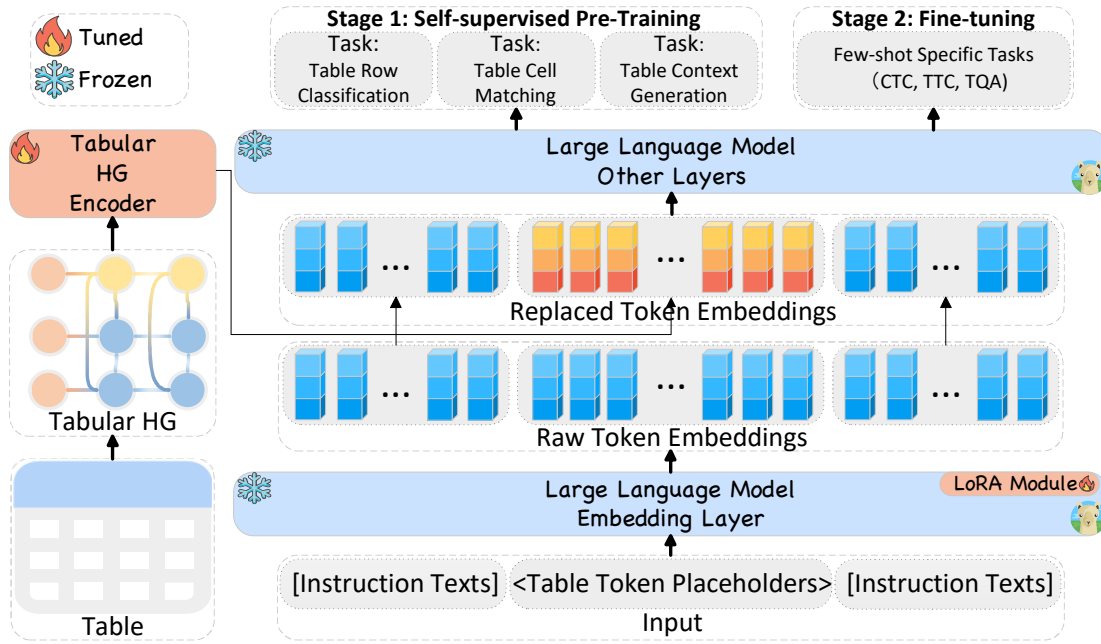


Figure 2: An overview of HeGTa framework. HeGTa processes  $\langle \text{table}, \text{instruction} \rangle$  as an input. First, the table is converted into an HG and processed by a Tabular HG encoder to generate a vector for each tabular node, while the LLM transforms instruction texts into initial token embeddings. Subsequently, the HG encoder’s outputs serve as soft prompts for the LLM, enabling the replacement of placeholder embeddings with actual tabular node vectors. The modified embedding sequence is then processed by the remaining LLM layers. Throughout Stage 1 and Stage 2, only the weights of red components are tuned.

actual table node vectors post-processing by the LLM’s Embedding layer. The quantity of these placeholders matches the number of table nodes relevant to the current task. The tokens  $\langle \text{table\_start} \rangle$  and  $\langle \text{table\_end} \rangle$  signify the beginning and ending delimiters of the table placeholders.

The forward propagation in HeGTa begins with the input of a  $\langle \text{Table}, \text{Instruction} \rangle$  pair, where the instruction text passes through the LLM’s embedding layer, and the table is transformed into a Tabular HG before being processed by the HGNN. The LLM embedding layer assigns embeddings to each token, creating a sequence, while the HGNN’s output provides aggregated vector representations for each table node. Subsequently, HeGTa replaces the embeddings corresponding to the table token placeholders within the sequence with the actual node vectors. This adjusted embedding sequence is then fed into the LLM’s remaining layers.

To enhance the HeGTa’s understanding of tables, we introduce three tasks tailored for the graph-enhanced LLM, each varying in granularity: Table Row Classification, Table Cell Matching, and Table Context Generation. These tasks are designed to integrate table representations into the LLM’s semantic space. The complexity of these tasks increases progressively, aiming to incrementally improve the model’s ability to comprehend semantic information. Detailed descriptions of each task are provided below.

**Table Row Classification (TRC).** The objective of this task is to train the model to accurately identify the category of each table row. Utilizing the vector representations of Row nodes provided by the HG encoder, the model dis-

cerns whether a node corresponds to a Header Row or a Data Row. The categorization of Row nodes is automated by an algorithm elaborated in Appendix, eliminating the need for manual intervention and thus qualifying as a self-supervised training process. In the dataset we utilize, the labeling accuracy of the algorithm reaches 97.6%. This task facilitates the model’s initial grasp of coarse-grained information about the structure of table rows.

**Table Cell Matching (TCM).** The task involves supplying the model with vector representations of each cell node alongside a list of shuffled cell contents. The model’s objective is to correctly pair each cell node vector with its corresponding original text. For instance, referring to the table depicted in Fig. 3, the model needs to align the node vector of  $c_{1,1}$  with the string “income” within the list correctly. This training task enables the model to discern the semantic information encapsulated within the cell contents based on the graph node vectors. Essentially, it aligns the semantic space of the HG encoder and the LLM.

**Table Context Generation (TCG).** This task aims to enable the model to generate context surrounding table data, utilizing the vector representations from the Table node. This task facilitates the model’s learning of the table’s global semantic information, proving beneficial for tasks requiring a comprehensive understanding of the table as an entity.

Examples of datasets in instruction format for the three tasks are provided in Fig. 4.

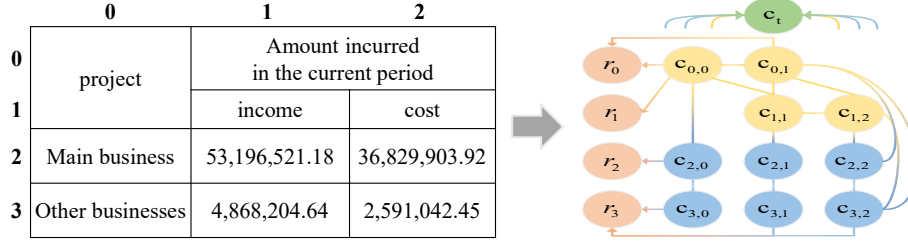


Figure 3: Table-to-heterogeneous graph conversion. Node types are color-coded: Table (green), Row (red), Data Cell (blue), and Header Cell (yellow). Edge types are similarly color-coded, with bidirectional edges shown as undirected lines. Some edges are omitted for clarity.

<p>Q: Given a sequence of table ROW tokens <code>&lt;table_start&gt;</code> <code>&lt;tabular_node&gt;</code>...<code>&lt;tabular_node&gt;</code> <code>&lt;table_end&gt;</code> that constitute a table. Each table has two ROW types: Header-Row and Data-Row. Please <b>classify each table ROW</b> according to the table ROW tokens.</p> <p>A: Based on the given 4 table ROW tokens, we can classify them as follows: Table ROW token 0 corresponds to Header-Row. Table ROW token 1 corresponds to Header-Row. Table ROW token 2 corresponds to Data-Row. Table ROW token 3 corresponds to Data-Row.</p>
<p>Q: Given a sequence of table CELL tokens <code>&lt;table_start&gt;</code> <code>&lt;tabular_node&gt;</code>...<code>&lt;tabular_node&gt;</code> <code>&lt;table_end&gt;</code> that constitute a table. Each table CELL token contains the content within the CELL. The following is a list of CELL contents:</p> <p>0) "income", 1) "Float Value: 36,829,903.92", 2) "Other businesses", ... , 9) "cost". Please <b>reorder the list of CELLS</b> according to the order of table CELL tokens (i.e., complete the matching of table CELL tokens and CELL contents).</p> <p>A: Based on the given 8 table CELL tokens and the list of table CELL contents, we obtain the matching of table CELL tokens and contents as follows: Table CELL token 0 corresponds to "project" ... Table CELL token 9 corresponds to "Float Value: 2,591,042.45".</p>
<p>Q: Given a tabular TABLE token <code>&lt;table_start&gt;</code> <code>&lt;tabular_node&gt;</code> <code>&lt;table_end&gt;</code> that represents a table. The TABLE token contains the content within the table. Please <b>generate the context</b> according to the tabular TABLE token.</p> <p>A: Based on the given TABLE CELL token, the context about the table is: The income and cost of the main business of the company's current amount is "53,196,521.18", "36,829,903.92"</p>

Figure 4: Examples of three self-supervised instruction tuning datasets, each tailored for distinct tasks: Table Row Classification, Table Cell Matching, and Table Context Generation.

### 4.3 Stage 2: Task-specific Instruction Fine-tuning

Following the completion of the self-supervised pre-training tasks in Stage 1, the HeGTa has successfully aligned the representation spaces of the HG encoder and the LLM. This alignment, combined with the intrinsic capabilities of the LLM, empowers HeGTa to effectively comprehend the topological nuances of complex tables. Consequently, when applied to a specific downstream task (i.e., CTC, TTC, TQA), HeGTa requires only a small number of training samples to grasp the expected answer format for the downstream task and reorganize its pre-existing knowledge into the suitable output format. Example prompts of downstream tasks can be found in Appendix. When fine-tuning in Stage 2, the model adopts the same tuning strategy as in Stage 1, with all parameters frozen except for those of the HG Encoder. To enhance the reasoning capabilities of HeGTa, we annotated the reasoning process with Chain of Thought (CoT) (Wei et al. 2022) for  $k$ -shot TQA samples. Inspired by Auto-CoT (Zhang et al. 2022), we first clustered the table samples in the test set, and then clustered the corresponding questions within each table cluster. Finally, we selected the central TQA samples from each question cluster for annotation. This approach ensures the diversity of contextual examples with a limited number of annotations.

## 5 Experiments

### 5.1 Datasets

To evaluate HeGTa, we selected a variety of datasets that are widely studied and easy to parse, including *WCC* (Ghasemi-

Gol and Szekely 2018), *IM-TQA* (Zheng et al. 2023), *HiTab* (Cheng et al. 2021b), *WTQ (Raw)* along with its header-flatten version *WTQ (Flatten)* (Pasupat and Liang 2015) and *TabFact* (Chen et al. 2020), each pertinent to CTC, TTC, or TQA, respectively.

Statistics for these datasets are presented in Table 1, illustrating the types of annotations, the primary domains covered, and the proportion of complex tables. Given the focus on few-shot TU scenarios, we only list the size of test sets. Comprehensive details of the dataset annotations and pre-processing are available in Appendix.

### 5.2 Baselines

We compare HeGTa with several strong baselines to verify its effectiveness. These baselines can be categorized into four groups according to their frameworks. ForTap (Cheng et al. 2021a) and GetPt (Jia et al. 2023) emulate BERT, devising specific pretext tasks tailored for tabular data to pre-train the Transformer encoder (Devlin et al. 2019). TabularNet (Du et al. 2021) and TabPrompt (Jin et al. 2023b) incorporate a graph encoder. Binder (Cheng et al. 2022), AugCodex (Cao et al. 2023) and COTable (Wang et al. 2024) leverage LLMs to generate symbolic programs, which are then executed to obtain the final answer. Compared to symbolic methods, TableLlama (Zhang et al. 2023a) and TableCoT (Chen 2022) leverage the textual understanding capabilities of LLMs to directly address tabular tasks.

Datasets	Annotation Type			Table Info			
	CTC	TTC	TQA	# Test Tables	% Complex Tables	# QA pairs	Main Domains
IM-TQA	✓	✓	✓	153	47.71	627	Manufacturing
WCC		✓		371	–	–	General
HiTab	✓		✓	3597	92.88	1584	Crime, Health
WTQ (Flatten)			✓	2108	0.00	4344	General
WTQ (Raw)			✓	2108	14.80	4344	General
TabFact			✓	1695	0.00	12779	General

Table 1: Dataset Statistics. “✓” indicates the type of annotation in the dataset that has this task in it. “# Test Tables”, “% Complex Tables” and “# QA pairs” columns show the number of tables in the test set, the percentage of complex tables and the numbers of QA pairs, respectively.

Model Type	Models	CTC		TTC		TQA				
		HiTab	IM-TQA	WCC	IM-TQA	WTQ (Flat)	WTQ (Raw)	TabFact	HiTab	IM-TQA
Bert-like Encoder-based	ForTap	56.33	47.34	46.14	53.40	27.33	26.52	28.67	24.88	29.08
	GetPt	58.45	49.67	48.62	56.45	23.69	23.34	25.03	23.34	22.07
Graph-based	TabularNet	54.14	45.62	46.01	56.41	19.18	18.58	20.81	22.34	20.05
	TabPrompt	<u>63.56</u>	46.77	50.31	54.34	17.71	14.77	19.56	20.20	14.14
LLM-based (Symbolic)	Binder	–	–	–	–	43.16	40.05	59.04	44.11	41.25
	Aug-Codex	–	–	–	–	33.32	32.28	57.07	45.87	42.16
	COTable	–	–	–	–	<u>44.38</u>	<u>42.97</u>	<u>59.76</u>	<u>49.68</u>	43.06
LLM-based (Textual)	TableCoT	53.12	49.76	45.57	53.80	31.42	32.12	52.01	47.55	45.21
	TableLlama	60.15	<u>53.40</u>	<u>51.56</u>	<u>57.55</u>	38.88	39.50	–	–	<u>48.82</u>
HeGTa		<b>66.39</b>	<b>59.58</b>	<b>55.32</b>	<b>58.46</b>	<b>45.12</b>	<b>45.96</b>	<b>60.61</b>	<b>51.56</b>	<b>50.73</b>

Table 2: Overall evaluation results on three TU tasks with best **bolded** and runner-up underlined. ‘–’ indicates that the current framework cannot handle the current task due to some limitations.<sup>1</sup>

### 5.3 Implementation Details

**General Setup Across All Tasks.** We employ Vicuna-7B-v1.5 (Chiang et al. 2023) as the base LLM and a 2-layer Heterogeneous Graph Transformer (HGT) (Wang et al. 2020a) as our tabular HG encoder whose hidden dimension is set as 1024. The initial vectors of nodes in HG are obtained by S-BERT (Reimers and Gurevych 2019), whose dimension is 768. We integrate a LoRA (Hu et al. 2021) module to the embedding layer of the LLM. At any tuning stage, HeGTa exclusively tunes the parameters of the HGT and the LoRA module. This lightweight configuration allows our model to be trained on a single 4090 GPU. We assemble the tables from training and validation sets of the datasets mentioned above along with a selection of *TURL* (Deng et al. 2020) corpus as a comprehensive dataset consisting of 200k tables for use in the pre-training stage.

**General Setup across Evaluation Tasks.** Following a typical  $k$ -shot setup (Liu et al. 2021; Wang et al. 2020b) to validate the HeGTa’s performance on few-shot TU, we randomly generate five 8-shot tasks for both training and validation on each dataset (i.e., 8 train/dev tables for CTC, 8

train/dev tables per class for TTC and 8 train/dev QA pairs for TQA). Each training task, paired with a corresponding validation task, is utilized to fine-tune the models optimally for subsequent testing. Table 2 presents the average performance on the test set across the five tasks.

**Evaluation Metrics.** We adopt Macro-F1, Macro-F1 and accuracy as the evaluation metric for CTC, TTC and TQA, respectively. Given the generative structure of LLMs, relying solely on exact matches might inaccurately categorize some correct responses (e.g., adding an answer with an additional period at the end). Therefore, we incorporate the semantic-match evaluator (Cheng et al. 2022) for TQA, which preprocesses the outputs prior to evaluation.

### 5.4 Results and Analysis

Table 2 compares the performance of different methods on all datasets described in § 5.1. From the results, We have the following observations: 1) HeGTa outperformed all baselines across 9 datasets, demonstrating its superiority over both non-LLM methods and LLMs with equivalent parameters. Additionally, as shown in Appendix, HeGTa demonstrates competitive performance in embedding-related tasks (i.e., CTC and TTC) even compared to SOTA GPT-4. 2) LLM-based symbolic methods excel in TQA but are inher-

<sup>1</sup>For a fair comparison, all LLM-based methods use the open-source LLaMA2-7B (Touvron et al. 2023) as the underlying model. More details of baselines are shown in Appendix.



Module	CTC		TTC		TQA				
	HiTab	IM-TQA	WCC	IM-TQA	WTQ (Flat)	WTQ (Raw)	TabFact	HiTab	IM-TQA
<i>w/o</i> TRC	-2.65	-1.53	+1.02	-1.40	-1.10	-1.34	-0.89	-0.78	+1.19
<i>w/o</i> TCM	-4.50	-4.88	-3.22	-2.44	-2.31	-2.71	-2.10	-1.67	-2.45
<i>w/o</i> TCG	+0.11	+1.19	-1.61	-1.73	-2.21	-2.54	-2.57	-0.30	-2.58
<i>w/o</i> HG	-3.28	-1.61	-0.24	-0.60	-1.80	-1.75	-1.68	-1.01	-1.31
<i>w/o</i> hl	-2.25	+0.90	+0.42	-0.75	-0.74	-0.85	-0.79	-0.20	+0.56
<i>w/o</i> CoT	–	–	–	–	-2.31	-2.35	-2.57	-1.98	-1.31

Table 3: Ablation results on all datasets. “*w/o* TRC”, “*w/o* TCM”, and “*w/o* TCG”: HeGTa pre-trained without the specified objective. “*w/o* HG”: HeGTa pre-trained with homogeneous graphs. “*w/o* hl”: HeGTa pre-trained with linking all neighboring cells directly instead of the heuristic linking strategy.

ently incapable of handling other table tasks due to their architectural design. Moreover, methods with 7B parameters struggle with reasoning and generating precise, executable code compared to those based on ChatGPT. For further details, see Appendix. 3) WTQ (Raw) represents the unprocessed version of WTQ (Flat), preserving the complexity of hierarchical table headers. The significant performance drop of other methods on WTQ (Raw) highlights HeGTa’s ability to effectively handle relational tables with complex structures. 4) Compared to HeGTa, other LLM-based frameworks underperform due to the loss of table topological information, even with enhancements like TableLlama’s extensive training on linearized tables or TableCoT’s use of HTML, a format better suited for LLMs (Sui et al. 2024). This underscores the effectiveness of introducing GNNs in overcoming the limitations of linearized table representations, thereby unlocking the full potential of LLMs.

**Strengths and Weaknesses of LLM-Based Models** All QA samples of HiTab can be classified into lookup-based reasoning (L-R) and computation-based reasoning (C-R) (e.g., sum, max, count). Table 4 presents the performance of three LLM-based methods on two types of QA samples. TableCoT shows superior performance in L-R compared to C-R, aligning with the conclusion (Chen et al. 2022) that LLMs struggle with numerical calculations. COTable leverages SQL to mitigate the shortcomings of LLMs in numerical reasoning, leading to the highest accuracy in C-R. However, the hierarchical structure causes a decline in SQL performance. HeGTa addresses the issue of structural information loss of linearized tables. As a result, HeGTa achieves notable improvements, particularly in L-R samples. Additionally, without restricting to the few-shot setting, i.e., tuning the model with the entire training set, HeGTa consistently maintains top performance, as shown in Appendix.

## 5.5 Ablation Study

We conduct ablation experiments to evaluate each component’s effectiveness, as shown in Table 3.

We begin by examining the individual improvements contributed by each of the three self-training tasks. To evaluate their specific impact, we pre-train three models, each omitting one of the objectives: TRC, TCM, or TCG. These three pre-training objectives improve the LLM’s understanding

	L-R		C-R		Total	
	#	%	#	%	#	%
# Test set	1133	100.00	451	100.00	1584	100.00
TableCoT	612	54.02	195	43.24	806	47.55
COTable	578	51.02	265	58.76	842	49.68
HeGTa	669	59.05	206	45.68	874	51.56

Table 4: Further results of the three methods on HiTab. “#” represents the number of correctly answered samples, and “%” denotes the proportion of correctly answered samples.

of tabular structure. The TCM task most effectively aligns the tabular modality with the LLM’s semantic space. The TRC task enhances performance on relational tables, while the TCG task boosts TQA performance but slightly reduces CTC performance.

We conducted ablation studies on CoT and our tabular graph construction method by removing the CoT text, converting tables into homogeneous rather than heterogeneous graphs, and replacing the heuristic linking (hl) strategy with a naive rule that links all neighboring cells. The results clearly demonstrate the significant impact of CoT on TQA tasks. The introduction of HGs benefits all tasks, with particularly notable improvements in TU with hierarchical structures. However, compared to the benefits of HGs, the hl strategy shows less pronounced gains for non-relational tables, possibly because the naive rule is more effective for aggregating and transmitting information in these tables.

## 6 Conclusion

In this paper, we introduce a novel framework, HeGTa, tailored for few-shot complex TU. The effectiveness of HeGTa is validated across multiple tabular datasets, accompanied by an in-depth ablation study to examine the impact of each component. In future work, we plan to expand HeGTa’s applicability to tables featuring more diverse layouts and to further improve HeGTa’s performance in TQA tasks by integrating the program-aided paradigm into our approach.

## 7 Acknowledgments

This work is partially supported by National Nature Science Foundation of China under No. 62476058, by the project "Key Laboratory of rich-media Digital Publishing Content Organization and Knowledge Service Open Fund-Research on Knowledge-enhanced Training Techniques of Large Language Model" No. ZD2024-04/01. We thank the Big Data Computing Center of Southeast University for providing the facility support on the numerical calculations in this paper.

## References

- Cao, Y.; Chen, S.; Liu, R.; Wang, Z.; and Fried, D. 2023. API-Assisted Code Generation for Question Answering on Varied Table Structures. *ArXiv*, abs/2310.14687.
- Chen, W. 2022. Large Language Models are few(1)-shot Table Reasoners. *ArXiv*, abs/2210.06710.
- Chen, W.; Ma, X.; Wang, X.; and Cohen, W. W. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.
- Chen, W.; Wang, H.; Chen, J.; Zhang, Y.; Wang, H.; Li, S.; Zhou, X.; and Wang, W. Y. 2020. TabFact: A Large-scale Dataset for Table-based Fact Verification. In *International Conference on Learning Representations*.
- Cheng, Z.; Dong, H.; Cheng, F.; Jia, R.; Wu, P.; Han, S.; and Zhang, D. 2021a. FORTAP: Using Formulas for Numerical-Reasoning-Aware Table Pretraining. In *Annual Meeting of the Association for Computational Linguistics*.
- Cheng, Z.; Dong, H.; Wang, Z.; Jia, R.; Guo, J.; Gao, Y.; Han, S.; Lou, J.-G.; and Zhang, D. 2021b. HiTab: A Hierarchical Table Dataset for Question Answering and Natural Language Generation. *ArXiv*, abs/2108.06712.
- Cheng, Z.; Xie, T.; Shi, P.; Li, C.; Nadkarni, R.; Hu, Y.; Xiong, C.; Radev, D. R.; Ostendorf, M.; Zettlemoyer, L.; Smith, N. A.; and Yu, T. 2022. Binding Language Models in Symbolic Languages. *ArXiv*, abs/2210.02875.
- Chiang, W.-L.; Li, Z.; Lin, Z.; Sheng, Y.; Wu, Z.; Zhang, H.; Zheng, L.; Zhuang, S.; Zhuang, Y.; Gonzalez, J. E.; et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%\* chatgpt quality. See <https://vicuna.lmsys.org> (accessed 14 April 2023).
- Deng, X.; Sun, H.; Lees, A.; Wu, Y.; and Yu, C. 2020. TURL: Table Understanding through Representation Learning. *SIGMOD Rec.*, 51: 33–40.
- Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv*, abs/1810.04805.
- Dong, H.; Cheng, Z.; He, X.; Zhou, M.; Zhou, A.; Zhou, F.; Liu, A.; Han, S.; and Zhang, D. 2022. Table pre-training: A survey on model architectures, pre-training objectives, and downstream tasks. *arXiv preprint arXiv:2201.09745*.
- Du, L.; Gao, F.; Chen, X.; Jia, R.; Wang, J.; Han, S.; and Zhang, D. 2021. TabularNet: A Neural Network Architecture for Understanding Semantic Structures of Tabular Data. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.
- Ghasemi-Gol, M.; and Szekely, P. A. 2018. TabVec: Table Vectors for Classification of Web Tables. *ArXiv*, abs/1802.06290.
- Herzig, J.; Nowak, P. K.; Müller, T.; Piccinno, F.; and Eisen-schlos, J. M. 2020. TaPas: Weakly Supervised Table Parsing via Pre-training. *ArXiv*, abs/2004.02349.
- Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.
- Hu, Z.; Dong, Y.; Wang, K.; and Sun, Y. 2020. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, 2704–2710.
- Jia, R.; Guo, H.; Jin, X.; Yan, C.; Du, L.; Ma, X.; Stankovic, T.; Lozajic, M.; Zoranovic, G.; Ilic, I.; Han, S.; and Zhang, D. 2023. GetPt: Graph-enhanced General Table Pre-training with Alternate Attention Network. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.
- Jin, M.; Wang, S.; Ma, L.; Chu, Z.; Zhang, J. Y.; Shi, X. L.; Chen, P.-Y.; Liang, Y.; Li, Y.-F.; Pan, S.; and Wen, Q. 2023a. Time-LLM: Time Series Forecasting by Reprogramming Large Language Models. *ArXiv*, abs/2310.01728.
- Jin, R.; Wang, J.; Tan, W.; Chen, Y.; Qi, G.; and Hao, W. 2023b. TabPrompt: Graph-based Pre-training and Prompting for Few-shot Table Understanding. In *Conference on Empirical Methods in Natural Language Processing*.
- Li, X. L.; and Liang, P. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *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)*, abs/2101.00190.
- Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023. Visual Instruction Tuning. *ArXiv*, abs/2304.08485.
- Liu, Z.; Fang, Y.; Liu, C.; and Hoi, S. C. 2021. Relative and absolute location embedding for few-shot node classification on graph. In *Proceedings of the AAAI conference on artificial intelligence*, 4267–4275.
- Lu, W.; Zhang, J.; Zhang, J.; and Chen, Y. 2024. Large language model for table processing: A survey. *arXiv preprint arXiv:2402.05121*.
- Ma, S.; Liu, J.-w.; and Zuo, X. 2023. Self-supervised learning for heterogeneous graph via structure information based on metapath. *Applied Soft Computing*, 143: 110388.
- Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L. E.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P. F.; Leike, J.; and Lowe, R. J. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.
- Pasupat, P.; and Liang, P. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *Annual Meeting of the Association for Computational Linguistics*.
- Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in*



- Natural Language Processing*. Association for Computational Linguistics.
- Sui, Y.; Zhou, M.; Zhou, M.; Han, S.; and Zhang, D. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 645–654.
- Tang, J.; Yang, Y.; Wei, W.; Shi, L.; Su, L.; Cheng, S.; Yin, D.; and Huang, C. 2023. GraphGPT: Graph Instruction Tuning for Large Language Models. *ArXiv*, abs/2310.13023.
- Touvron, H.; Martin, L.; Stone, K. R.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D. M.; Blecher, L.; Ferrer, C. C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A. S.; Hosseini, S.; Hou, R.; Inan, H.; Kardaş, M.; Kerkez, V.; Khabsa, M.; Kloumann, I. M.; Korenev, A. V.; Koura, P. S.; Lachaux, M.-A.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kam-badur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *ArXiv*, abs/2307.09288.
- Wang, F.; Sun, K.; Chen, M.; Pujara, J.; and Szekely, P. 2021. Retrieving complex tables with multi-granular graph representation learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1472–1482.
- Wang, K.; Shen, W.; Yang, Y.; Quan, X.; and Wang, R. 2020a. Relational Graph Attention Network for Aspect-based Sentiment Analysis. In *ACL*.
- Wang, N.; Luo, M.; Ding, K.; Zhang, L.; Li, J.; and Zheng, Q. 2020b. Graph few-shot learning with attribute matching. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 1545–1554.
- Wang, Y.; Kordi, Y.; Mishra, S.; Liu, A.; Smith, N. A.; Khoshabi, D.; and Hajishirzi, H. 2022. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *Annual Meeting of the Association for Computational Linguistics*.
- Wang, Z.; Dong, H.; Jia, R.; Li, J.; Fu, Z.; Han, S.; and Zhang, D. 2020c. TUTA: Tree-based Transformers for Generally Structured Table Pre-training. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.
- Wang, Z.; Zhang, H.; Li, C.-L.; Eisenschlos, J. M.; Perot, V.; Wang, Z.; Miculicich, L.; Fujii, Y.; Shang, J.; Lee, C.-Y.; et al. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*.
- Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.
- Yang, Y.; Guan, Z.; Wang, Z.; Zhao, W.; Xu, C.; Lu, W.; and Huang, J. 2022. Self-supervised heterogeneous graph pre-training based on structural clustering. *Advances in Neural Information Processing Systems*, 35: 16962–16974.
- Ye, Y.; Hui, B.; Yang, M.; Li, B.; Huang, F.; and Li, Y. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 174–184.
- Yin, P.; Neubig, G.; tau Yih, W.; and Riedel, S. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. *ArXiv*, abs/2005.08314.
- Zhang, T.; Yue, X.; Li, Y.; and Sun, H. 2023a. TableLlama: Towards Open Large Generalist Models for Tables. *ArXiv*, abs/2311.09206.
- Zhang, Y.; Henkel, J.; Floratou, A.; Cahoon, J.; Deep, S.; and Patel, J. M. 2023b. ReAcTable: Enhancing ReAct for Table Question Answering. *ArXiv*, abs/2310.00815.
- Zhang, Z.; Zhang, A.; Li, M.; and Smola, A. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.
- Zheng, M.; Hao, Y.; Jiang, W.-J.; Lin, Z.; Lyu, Y.; She, Q.; and Wang, W. 2023. IM-TQA: A Chinese Table Question Answering Dataset with Implicit and Multi-type Table Structures. In *Annual Meeting of the Association for Computational Linguistics*.