HyperG: Hypergraph-Enhanced LLMs for Structured Knowledge

Sirui Huang*
University of Technology Sydney
Sydney, Australia
sirui.huang@student.uts.edu.au
Hong Kong Polytechnic University
Hong Kong SAR, China
sirui.huang@connect.polyu.hk

Hanqian Li*
Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China

Yanggan Gu
Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China

Xuming Hu†
Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China
xuminghu@hkust-gz.edu.cn

Qing Li
Hong Kong Polytechnic University
Hong Kong SAR, China

Guandong Xu‡
University of Technology Sydney
Sydney, Australia
guandong.xu@uts.edu.au
Education University of Hong Kong
Hong Kong SAR, China
gdxu@eduhk.hk

Abstract

Given that substantial amounts of domain-specific knowledge are stored in structured formats, such as web data organized through HTML, Large Language Models (LLMs) are expected to fully comprehend this structured information to broaden their applications in various real-world downstream tasks. Current approaches for applying LLMs to structured data fall into two main categories: serialization-based and operation-based methods. Both approaches, whether relying on serialization or using SQL-like operations as an intermediary, encounter difficulties in fully capturing structural relationships and effectively handling sparse data. To address these unique characteristics of structured data, we propose *HyperG*, a hypergraph-based generation framework aimed at enhancing LLMs' ability to process structured knowledge. Specifically, *HyperG* first augment sparse data with contextual information, leveraging the generative power of LLMs, and incorporate a prompt-attentive hypergraph learning (PHL) network to encode both the augmented information and the intricate structural relationships within the data. To validate the effectiveness and generalization of *HyperG*, we conduct extensive experiments across two different downstream tasks requiring structured knowledge. Our code is publicly available at: https://github.com/s1ruihuang/HyperG.

CCS Concepts

• **Mathematics of computing** → **Hypergraphs**; • **Information systems** → **Language models**.

*Both authors contributed equally to this research.
†corresponding author: xuminghu@hkust-gz.edu.cn
‡corresponding author: gdxu@eduhk.hk

SIGIR '25, Padua, Italy

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3730002

Keywords

Hypergraph, Large Language Models, Table Understanding

ACM Reference Format:
Sirui Huang, Hanqian Li, Yanggan Gu, Xuming Hu, Qing Li, and Guandong Xu. 2025. *HyperG*: Hypergraph-Enhanced LLMs for Structured Knowledge. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3726302.3730002

1 Introduction

With the advancement of digitalization across various industries, substantial amounts of structured knowledge are stored in tabular formats. This structured knowledge, often containing domain-specific information closely tied to different downstream tasks, complements the general knowledge acquired by Large Language Models (LLMs) during pre-training, thereby enhancing their capability to support downstream queries and reasoning [8, 43].

LLMs, leveraging their sophisticated linguistic capabilities and extensive knowledge base, have been widely utilized as one-/few-shot learners in various structured tasks [14, 15, 24, 59]. Currently, the approaches for applying LLMs on structured knowledge, including tables, fall into two primary categories: serialization-based [15, 21, 33] and operation-based methods [22, 32, 47, 52]. Serialization-based methods convert structured knowledge into sequences of tokens, enabling the model to process the structured data in conjunction with task descriptions [33, 41]. For example, Hegselmann et al. [15] utilize a Table-to-Text model or a LLM as the serializer to convert tables into natural language strings, which are then fed into the LLM along with task descriptions. However, serializing structured data can undermine the inherent structural relationships, especially in larger tables, potentially leading to serve knowledge forgetting and diminished logical coherence during reasoning [28, 56]. Additionally, the serialized formats critically influenced the performance of LLMs [42]. The operation-based methods extract relevant information from structured data using SQL-like operations based on task requirements, and then incorporate this knowledge into LLMs

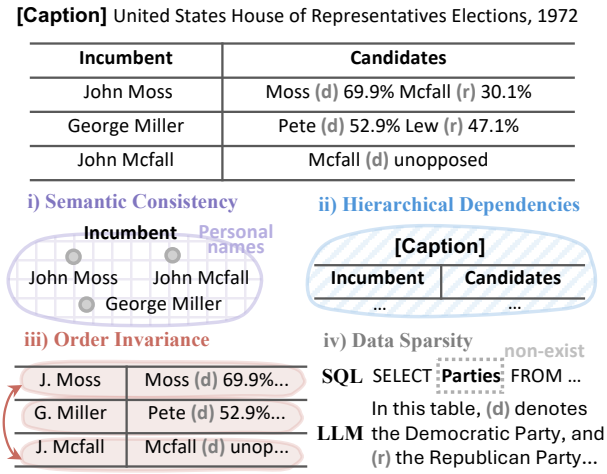


Figure 1: An example illustrates the three aspects of the structural relationships in tables: i) Semantic Consistency, ii) Hierarchical Dependencies, and iii) Order Invariance. Additionally, it highlights the data sparsity issue iv), where incomplete data affects SQL queries over the table .

to generate responses. While the SQL-like operations account for structural relationships, these methods fail to fully harness the extensive knowledge base of LLMs for effective reasoning [59]. As shown in Figure 1 d), the "(d)" and "(r)" represent the Democratic and Republican parties, respectively. Since the parties are not explicitly listed in a separate column, retrieving information about the political affiliations of incumbents via SQL queries is challenging. However, LLMs can easily interpret this information due to their advanced in-context learning abilities and knowledge base. Therefore, **structural relationships** and **data sparsity** are two critical challenges that current methods are not fully account for when reasoning over structured knowledge, which differs fundamentally from the unstructured text inputs LLMs typically handle [12].

Graphs are structure-aware, making them a natural choice for modeling structural relationships. However, traditional graphs remain insufficient in effectively capturing the group relationships between rows and columns. Unlike traditional graphs, where an edge connects only two nodes, a hyperedge in a hypergraph can connect multiple cells nodes in an unordered manner. Hypergraphs consider the **structural relationships** within tabular data from three aspects: i) Semantic Consistency. Data in the cells of the same row or column in a table generally correspond to a consistent semantic category, allowing LLMs to identify and infer implicit semantic relationships. As illustrated in Figure 1 i), the cells in the "Incumbent" column are all personal names. ii) Hierarchical Dependencies. Hyperedges are capable of capturing intricate, higher-order dependencies within structured knowledge, such as the dependencies of the captions, headers, and cells. iii) Order Invariance. Changing word order in natural language can alter meaning, but rearranging rows or columns in a table, swapping the Moss and McFall rows in Figure 1 iii), does not affect the overall semantics. To address the **sparsity** issue such as the incomplete parties in Figure 1 iv), hypergraphs facilitate high-order information propagation between nodes and hyperedges, thereby supplementing the representations of incomplete cells with information from their neighbors. In addition, the extensive general knowledge embedded in LLMs can be leveraged to address sparse data issues.

To enhance LLMs' capabilities on structured knowledge, we propose a novel ***Hyper***graph-based ***G***eneration framework, namely ***HyperG***, to facilitate seamless integration of knowledge from structure learning with hypergraph neural networks into LLMs, without losing focus on task-specific requirements. Specifically, *HyperG* explicitly guide the LLMs to augment sparse table cells with contextual information. We then construct semantics hypergraphs with the augmented table and introduce a novel Prompt-Attentive Hypergraph Learning (PHL) module that propagates task-specific inquiries in prompts along with embedded semantic knowledge across structures, and train this module jointly with the LLM. Our contributions are concluded as follows:

- **Towards structural relationship.** We propose *HyperG*, which uses hypergraphs to capture the semantic consistency, order invariance, and hierarchical dependencies within structured knowledge, thereby enhancing the LLM's capability to understand and reason over structured knowledge.
- **Towards data sparsity.** We design a novel hypergraph neural network to tackle the sparsity issue in tabular knowledge by utilizing the generative abilities of LLMs and then facilitates information propagation through hyperedges.
- **Experiments.** We conduct extensive experiments on various downstream tasks involving structured data to validate the effectiveness of our proposed *HyperG* framework.

2 Problem Definition

Aiming to enhance the capability of LLMs in handling knowledge stored in structured data with hypergraphs, in this paper, we consider tables as the structured data sources to illustrate our *HyperG* framework. We construct a hypergraph with the structured knowledge in table $\mathcal{T}$. Each table is formally represented as $\mathcal{T} = \{o, h_i, v_{m,n} | 0 \leq i \leq N, 0 \leq m \leq M, 0 \leq n \geq N\}$, where o is the table caption, h_i represents the header for the i^{th} column, $v_{m,n}$ represents the cell at the m^{th} row (denoted as $r_m \in \mathcal{R}$), and the n^{th} column (denoted as $c_n \in C$). As depicted in the upper left of Figure 2, the very upper-left cell is denoted as $v_{0,0}$. The task description prompt x, provided in natural language to the LLMs, includes a textual representation of the table $\mathcal{T}$ (in markdown format) and the essential inquiry ω regarding this table, following a specific template, $\omega \subset x$. Specifically, the essential inquiry ω can be claims in fact verification or questions in question answering. For tasks requiring the knowledge stored in $\mathcal{T}$, we aim to help pretrained LLMs (denoted as $LLM(\cdot)$) to understand and extract the structured knowledge relevant to the inquiry ω stored in $\mathcal{T}$, thereby improving the effectiveness of LLM's final generations.

3 Methodology

Figure 2 provides an overview of our proposed *HyperG* framework, which is designed to enhance the ability of LLMs to handle tasks that require knowledge embedded in structured data. This section details the workflow of *HyperG*, first augmenting the structured data with contextual information, followed by learning and integrating task-relevant structured knowledge into the LLMs to generate answers.

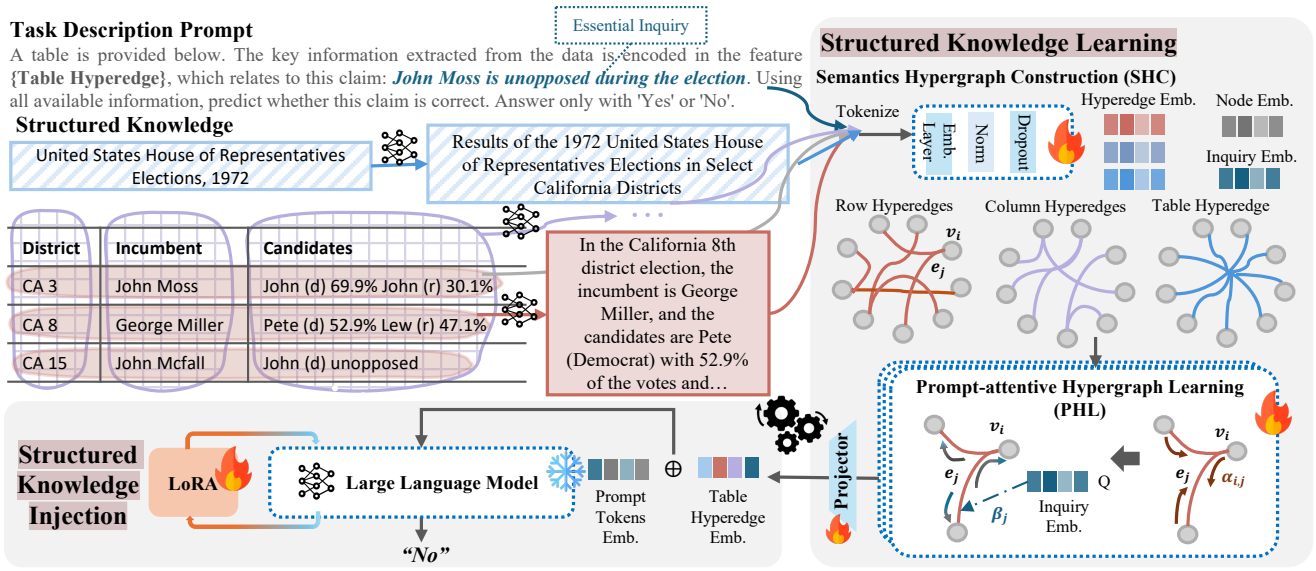


Figure 2: An overview of our proprosed *HyperG* framework.

3.1 Contextual Augmentation

To address the **data sparsity** issue caused by missing or incomplete information in cells, N/A, none, or incompleted descriptions such as the example in Figure 1. For each table $\mathcal{T}$, we augment its caption o, columns $c \in \mathcal{C}$, and rows $r \in \mathcal{R}$ with contextual information, leveraging the semantics understanding and generative ability of the large language model $LLM(\cdot)$.

Specifically, as shown in Figure 2, the caption o "*United States House of Representatives Elections, 1972*" is vague, as it does not specify where the elections occurred. After being supplemented with the contextual information from the table, the augmented caption, "*Results of the 1972 United States House of Representatives Elections in Select California Districts*", denoted as $\bar{o}$, more clearly illustrates the table's content. As for the sparse cells which contain missing or incomplete data, we utilize the LLM to generate descriptions for each row and column. Formally, for the m^{th} row,

$$\bar{r}_m = LLM(P_0(o, h_:, v_{m,:}) \tag{1}$$

where $\bar{r}_m$ represents the augmented description for the m^{th} row containing cells $v_{m,:} = (v_{m,1}, ..., v_{m,N})$, $h_: = (h_1, ..., h_N)$ denotes the N headers of table $\mathcal{T}$, and $P_0(\cdot)$ refers to the template used to prompt the LLM in generating the corresponding augmented summary. Specifically, in this paper, we define the augmentation prompt P_0 as: "*You will be given with the table caption and headers. Please enhance the caption/describe the given row/column corresponding to the table content.*" The descriptions for the columns c_n in table $\mathcal{T}$ are generated in a similar manner, yeilding $\bar{c}_n$.

3.2 Structured Knowledge Learning

After augmenting the sparse data with contextual information, *HyperG* learns **structural relationships** over knowledge that is structurally stored through two steps: first, by constructing hypergraphs that aligns with the semantics (SHC), and second, by utilizing a novel prompt-attentive neural network for hypergraph learning (PHL). This section will elaborate on how our proposed *HyperG* conducts these two steps in detail.

3.2.1 Semantics Hypergraph Construction (SHC). This step embeds the semantics of table $\mathcal{T}$ into a hypergraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{..., v_i, ...\}$ represents the set of node(vertex), and $\mathcal{E} = \{..., e_j, ...\}$ represents the set of hyperedges. Each hyperedge connects multiple nodes, $v_i \in \mathcal{N}_{e_j}$ denotes that the node v_i is included in the set of nodes connected by hyperedge e_j, while $e_j \in \mathcal{N}_{v_i}$ represents the hyperedge e_j is included in the set of hyperedges which connects node v_i. Each cell $v_{m,n}$ in the table $\mathcal{T}$ is treated as a node, $v_i \in \mathcal{V}$, $|\mathcal{V}| = M \times N$. The rows $r_m \in \mathcal{R}$, columns $c_n \in \mathcal{C}$, and the entire table $\mathcal{T}$ act as hyperedges, leading to three types: row hyperedges $e_{\mathcal{R}} = \{..., e_{r_m}, ...\} \subseteq \mathcal{E}$, column hyperedges $e_{\mathcal{C}} = \{..., e_{c_n}, ...\} \subseteq \mathcal{E}$, and table hyperedge $e_{\mathcal{T}} \subseteq \mathcal{E}$, $|\mathcal{E}| = M + N + 1$. The connections between the nodes $v \in \mathcal{V}$ and hyperedges $e \in \mathcal{E}$ within the hypergraph $\mathcal{G}$ are represented by an incidence matrix $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$, where each element $h_{i,j} = 1$ if node v_i is connected by hyperedge e_j, and $h_{i,j} = 0$ otherwise.

Specifically, we first tokenize the textual contents of cells, rows, columns, and captions using the BERT [10] tokenizer. For example, the augmented table caption is transformed into O number of tokens represented by $(\mathbf{t}_{\bar{o},1}, \mathbf{t}_{\bar{o},2}, ..., \mathbf{t}_{\bar{o},O}) = Tok_{BERT}(\bar{o})$. The tokens are subsequently passed to an embedding layer, represented as $Emb(\cdot)$, which has an output hidden dimension of d for semantics learning. Layer normalization and dropout layers are implemented in the embedding process to ensure robust generalization capabilities.

$$\mathbf{h}_{e^{\mathcal{T}}} = Dropout(LN(Emb(\mathbf{t}_{\bar{o},1}, \mathbf{t}_{\bar{o},2}, ..., \mathbf{t}_{\bar{o},O}))) \tag{2}$$

where $\mathbf{h}_{e^{\mathcal{T}}} \in \mathbb{R}^d$ is the hidden embedding of table hyperedge, $LN(\cdot)$ represents the layer normalization, and $Dropout(\cdot)$ refers to a dropout layer with a dropout rate 0.1. By representing the semantics of each cell content $v \in \mathcal{V}$ as node embedding $\mathbf{h}_v$ with Equation (2), representing the row and column descriptions $\bar{r}_m$ and $\bar{c}_n$ as row/column hyperedges $\mathbf{h}_{e^{\mathcal{R}}}$ and $\mathbf{h}_{e^{\mathcal{C}}}$, and the table hyperedge $\mathbf{h}_{e^{\mathcal{T}}}$, we construct the semantic hypergraph $\mathcal{G}$. Additionally, we

also calculate the semantic embedding for the essential inquiry (as highlighted in teal-blue in the task description prompt in Figure 2), denoted as $\mathbf{h}_\omega$ for further learning.

3.2.2 Prompt-attentive Hypergraph Learning (PHL). Provided with the hypergraph $\mathcal{G}$, we design a prompt-attentive hypergraph neural network to further learn structured knowledge from $\mathcal{G}$. In traditional hypergraph learning [2, 11, 13, 50], hyperedge embeddings typically do not directly participate in the propagation process; instead, hyperedges primarily serve to connect related nodes, with the focus on node embeddings. In *HyperG*, we aim to integrate the semantic embeddings of both nodes and hyperedges during propagation. Since the table cells contain diverse content, while the augmented hyperedge descriptions ($\bar{o}$, $\bar{r}$, and $\bar{c}$) are generated by the same LLM and maintain a consistent linguistic style, we apply node-to-edge and edge-to-node propagation using attention scores denoted by α and β with distinct designs. Specifically, inspired by [7], each PHL layer comprises two-step graph attention: first conducts *semantic-aware propagation* from nodes to their connected hyperedges, then *attentively integrate the embedding of the inquiry in the prompt* and propagates from edges to nodes.

Semantic-aware propagating. Nodes are embedded from the original table cells content with Equation (2), denoted as $\mathbf{h}_v$. We first propagate the original semantics embedded in $\mathbf{h}_v$ to each connected hyperedge e with the K-head hypergraph attention mechanism. The attention score $\alpha_v^{(k)}$ in node-to-edge propagation for node v at the k^{th} head is calculated as follows.

$$\alpha_v^{(k)} = \text{L-ReLU}(\sum \mathbf{Q}^{(k)} \cdot \mathbf{K}_v^{(k)}) \tag{3}$$

where L-ReLU$(\cdot)$ denotes the LeakyReLU activation function, the query representation $\mathbf{Q}^{(k)} = \mathbf{W}_{k,:} \in \mathbb{R}^{1\times d}$ is the k^{th} vector of a learnable weight $\mathbf{W} \in \mathbb{R}^{K\times d}$. We use another multi-layer perceptron to learn the key representation of the target node v, $\mathbf{K}_v^{(k)} = MLP_K^{(k)}(\mathbf{h}_v^l) \in \mathbb{R}^{1\times d}$.

Next, the information is propagated from the nodes to their connected hyperedges, formally represented as below.

$$\mathbf{h}_e^{l,(k)} = \sum_{v\in\mathcal{N}_e} \sigma(\alpha_v^{(k)})\mathbf{V}_v^{(k)} \tag{4}$$

where $\mathbf{V}_v^{(k)} = MLP_V^{(k)}(\mathbf{h}_v^l) \in \mathbb{R}^{1\times d}$ denotes a multilayer perceptron used to transform the node embedding $\mathbf{h}_v^l$ to its value representation, $\mathbf{h}_e^l$ and $\mathbf{h}_v^l$ represents the hyperedge e and its connected node v, respectively, as input to the l^{th} layer. The h_v^l is initialized by the semantics embedding of v, $\mathbf{h}_v^0 = \mathbf{h}_v \in \mathbb{R}^{1\times d}$. The softmax function $\sigma(\alpha_{v_i}^{(k)}) = \frac{exp(\alpha_{v_i}^{(k)})}{\sum_{v\in\mathcal{N}_e} exp(\alpha_v^{(k)})}$ computes the normalized attention score for each node $v_i \in \mathcal{N}_e$.The aggregation $\sum_{v\in\mathcal{N}_e}$ for the embedding of each node $\mathbf{h}_v^l$ can be performed using any aggregation function, such as summation.

As shown in Figure 3, to increase the representation and generation capability to be compatible with the LLM, the aggregated embedding of hyperedges $\mathbf{h}_e$ are processed using residual connections, normalization, and feed forward layers, following the architecture of transformer [46].

$$\hat{\mathbf{h}}_e^l = LN(FF(LN({}_k\mathbf{h}_e^{l,(k)} + \mathbf{W})) +_k \mathbf{h}_e^{l,(k)}) \tag{5}$$

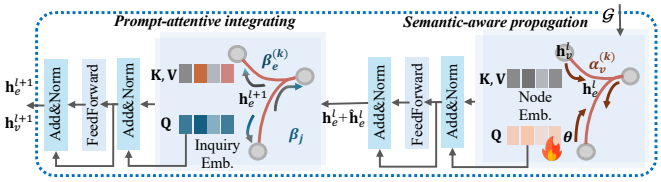


Figure 3: The detailed architecture of PHL.

where ${}_k\mathbf{h}_e^{l,(k)} \in \mathbb{R}^{K\times d}$ represents the concatenation of the outputs of K heads of the multi-head attention mechanism described in Equation (3)(4). The $\hat{\mathbf{h}}_e^{l,(k)} \in \mathbb{R}^{1\times d}$ denotes the hyperedge embedding incorporating information propagated from nodes, and is then concatenated to the original semantics embedding of hyperedge e, $\mathbf{h}_e^{l+1} = \hat{\mathbf{h}}_e^l + \mathbf{h}_e$, $\mathbf{h}_e^{l+1} \in \mathbb{R}^{1\times 2d}$.

Prompt-attentive integrating. The original semantics embedding of hyperedge $\mathbf{h}_e$ are embedded from the LLM-augmented descriptions obtained with Equations (1)(2). To integrate the task requirements described in prompts into hypergraph learning, we adopt the embedding of the essential inquiry $\mathbf{h}_\omega$ in the prompt to calculate the attention score $\beta_e^{(k)}$ for edge-to-node propagation, similar to the process in Equation (3) (2). The essential inquiry can be replaced by any textual information that is of particular relevance or concern to downstream tasks in the prompts.

$$\beta_v^{(k)} = \text{L-ReLU}(\sum MLP_Q^{(k)}(\mathbf{h}_\omega \cdot MLP_K^{(k)}(\mathbf{h}_e^{l+1}))) \tag{6}$$

$$\mathbf{h}_v^{l,(k)} = \sum_{e\in\mathcal{N}_v} \sigma(\beta_v^{(k)} MLP_V^{(k)}(\mathbf{h}_e^{l+1})) \tag{7}$$

where $e \in \mathcal{N}_v$ denotes that the set of all the hyperedge e which connects the target node v, $MLP_Q^{(k)}(\mathbf{h}_\omega) \in \mathbb{R}^{1\times d}$ represents the k^{th} vector of the query representation $MLP_Q(\mathbf{h}_\omega) \in \mathbb{R}^{K\times d}$, computed by a multilayer perceptron based on $\mathbf{h}_\omega$. This operation utilizes task requirements to attentively propagate information from hyperedges to nodes. This operation intuitively aligns with the human reasoning process, where relevant rows or columns are identified first, followed by a detailed examination of individual cells when handling tasks that require structured knowledge.

Similarly, the node embedding $\mathbf{h}_v^l$ learned from multi-head attentions are further processed with the residual connections, normalization, and feed-forward layers, formally as below.

$$\mathbf{h}_v^{l+1} = LN(FF(LN({}_k\mathbf{h}_v^{l,(k)} + MLP_Q^{(k)}(\mathbf{h}_\omega))) +_k \mathbf{h}_v^{l,(k)})) \tag{8}$$

where ${}_k\mathbf{h}_v^{l,(k)}$ denotes the concatenation of the outputs of K-head attention mechanism described in Equation (6)(7).

Through the adoptions of semantic-aware propagation and inquiry-attentive integration in each PHL layer, the hypergraph neural network attains a comprehensive understanding of the hierarchical semantics embedded within structured data. This approach ensures semantic consistency, comprehensively captures hierarchical dependencies, and preserves the order invariance property of structural relationships within the knowledge structure.

3.3 Structured Knowledge Integration

After completing the hypergraph learning process, the knowledge linked to each cell, along with the columns, rows, and caption,

is embedded in the representations of the nodes and hyperedges, which are further integrated into the generation process of LLMs.

3.3.1 Encoding Structured Knowledge. By connecting each cell node to the table hyperedge formed from the caption, as detailed in Section 3.2.1, the hidden embedding $\mathbf{h}_{e_\mathcal{T}}$ effectively captures the task-relevant structured knowledge of the entire table $\mathcal{T}$. Therefore, $\mathbf{h}^L_{e_\mathcal{T}}$ is mapped to the token space of LLM using a projector π.

$$\mathbf{e}_\mathcal{T} = \pi(\mathbf{h}^L_{e_\mathcal{T}}) \tag{9}$$

Here, the projected table embedding is denoted as $\mathbf{e}_\mathcal{T} \in \mathbb{R}^{d'}$, where d' denotes the dimension of the input tokens for the LLM. In *HyperG*, the projector π is implemented using two linear layers, with a ReLU activation function in between. The table embedding $\mathbf{e}_\mathcal{T}$ is then integrated into the token embeddings $\mathbf{e}_{x,:} = Tok_{LLM}(P_2(x))$ of the task description prompt x at the designated placeholder position labeled "Table Hyperedge" in natural language, as highlighted by the bold text in the upper left corner of Figure 2.

$$\hat{\mathbf{e}}_x = (\mathbf{e}_{x,:ph-start})(\mathbf{e}_\mathcal{T})(\mathbf{e}_{x,ph-end:}) \tag{10}$$

where $\mathbf{e}_{x,:ph-start}$ denotes all the prompt tokens preceding the placeholder, $\mathbf{e}_{x,ph-end:}$ denotes all the prompt tokens following the placeholder. Additionally, $\hat{\mathbf{e}}_x$ represents the tokens that integrate structured knowledge for further inference in the LLMs.

3.3.2 Training. Given the task description, markdown table, and the inquiry ω in prompt x, structured table $\mathcal{T}$, our *HyperG* jointly train the prompt-attentive hypergraph learning network with the LoRA [16]. The supervised fine-tuning process can be expressed in terms of the log likelihood loss. Given the input task description prompt x and target output y from the training set $\mathcal{D}$, there is,

$$\mathbb{E}_{(x,y,\mathcal{T})\in\mathcal{D}}\left[\sum_{t=1}^{T} log p_\theta(y_t|y_{1:t-1}, x, \mathcal{T})\right] \tag{11}$$

Here, the conditional probability distribution of the target generation output sentence y given prompt x is represented as $p_\theta(y|x) = \prod_{t=1}^{T} x_\theta(y_t|y_{<t}, x, \mathcal{T})$, where θ denotes the model parameters and T is the length of the generated sequence.

4 Experiments

To validate the effectiveness of *HyperG*, we have conduct extensive experiments to answer the following research questions.

- **RQ1**: How does the proposed *HyperG* perform compared to state-of-the-art (SoTA) methods when using various LLMs as backbones across different downstream tasks?
- **RQ2**: How does the proposed *HyperG* retain the *Order Invariance* of structural relationships?
- **RQ3**: How does *HyperG* retain the *Semantic Consistency* and *Hierarchical Dependencies* of structural relationships?
- **RQ4**: How do the different components influence *HyperG*?
- **RQ5**: Is *HyperG* scalable to tables of different sizes?

4.1 Experimental Setups

4.1.1 Tasks. We validate *HyperG* on two levels of downstream tasks as follows, with the statistics of the training data in Table 1.

- **Table Fact Verification (TFV).** This task aims at assessing *HyperG* in fact-checking. Specifically, we conduct experiments on TabFact [48], which contains 16k Wikipedia tables used as evidence for 118k human-annotated claims to explore fact verification with semi-structured knowledge.
- **Table Question Answering (TQA).** To validate *HyperG* on facilitating LLMs to reason over structured knowledge and provide better answers to user input questions, we test on the WiKiTableQuestions [36] dataset, which includes 14,152 examples of open question-answer pairs for training and 4,344 examples for testing.

Table 1: The statistics of training data.

Tasks	Answer Type	#Graphs	Avg. #Nodes	Avg. #Edges	Inquiry Avg. len
TFV	yes/no	1849	78.65	20.39	67.37
TQA	open answer	10141	125.11	28.94	65.05

4.1.2 Baselines. We compare our proposed *HyperG* against 12 baseline methods, categorized by their different ways of handling tables: operation-based methods [39, 47, 53] that use external operations like SQL queries and serialization-based methods [1, 45, 55] that transform information in structures into sequences then prompt into the LLMs. In terms parameter sizes, our comparison covers a range of model sizes range from 2 billion to 70 billion parameters. For a fair comparison, we evaluate the operation-based baseline methods [39, 47, 53] using the same backbone LLMs (**LLaMA3-8B-Instruct**, **LLaMA3.2-3B-Instruct**, **Gemma-2-9B-It**, and **Gemma-2-2B-It**) as those used for our *HyperG*. To reduce the impact of varying instruction-following abilities among different LLMs, we adopt the instruction-tuned verions of all the selected backbone LLMs in our experiments.

- **GPT Family** [34, 35] We use GPT-3.5-turbo and GPT-4o-mini models from the GPT family by OpenAI. GPT-3.5-turbo offers excellent cost-performance balance with fast inference, while GPT-4o-mini provides strong performance at lower computational costs, ideal for resource-constrained scenarios.
- **LLaMA3 Family** [45] We select LLaMA3.1-70B-Instruct, LLaMA3-8B-Instruct, and LLaMA3.2-3B-Instruct from the LLaMA3 series. The 70B variant excels at complex and long-context reasoning, the 8B variant is optimized for instruction understanding and generation, and the 3B variant offers rapid responses suitable for resource-limited scenarios.
- **Gemma-2-It** [1] is fine-tuned on Gemma-2 with user interactions, focusing on task-specific adaptability while ensuring efficiency through knowledge distillation from the very large model. In this paper, we use the 2B, 9B, 27B variants of Gemma-2-It.
- **TableLlama** [55] adopts LongLoRA to finetune on a dataset that includes a diverse range of serialized tables and the corresponding natural language task instructions.
- **Text-to-SQL** [39] designs in-context samples to instruct LLMs in generating SQL queries for answering questions.
- **Dater** [53] leverages LLMs to decompose the task into multiple sub-tasks, utilizing SQL queries to address each sub-task.
- **CHAIN-OF-TABLE** [47] prompts LLMs through in-context learning to iteratively produce operations and update the table, thereby constructing a reasoning chain in a structured format.
- **LoRA** [16] is a widely used technique for efficiently fine-tuning LLMs by updating a small number of low-rank weights.

Table 2: Comparison of the performance of our *HyperG* and 13 baseline methods based on varying parameter sizes. The first group of methods prompts LLMs with serialized tables, while the methods in each of the last four groups use the same backbone LLMs. The best and second-best results are marked with bold and underline, respectively.

Methods	Backbones	TFV			TQA			
		Acc.	Prec.	F1	Denot. Acc.	ROUGE-1	ROUGE-2	ROUGE-L
Gemma-2-2B-It [1]	-	59.80	60.55	58.54	31.88	39.68	17.81	39.60
LLaMA3.2-3B-Instruct [45]	-	54.90	56.66	54.89	24.77	35.11	16.62	35.07
TableLlama [55]	LLaMA2-7B	70.04	71.27	69.39	24.63	28.07	13.95	27.98
LLaMA3-8B-Instruct [45]	-	66.29	66.29	66.28	37.85	47.58	21.42	47.49
Gemma-2-9B-It [1]	-	75.00	75.20	74.99	46.85	55.73	24.99	55.73
Gemma-2-27B-It [1]	-	76.50	76.29	75.94	53.96	61.39	27.88	61.31
LLaMA-3.1-70B-Instruct [45]	-	**79.16**	**79.67**	**79.12**	**55.71**	**64.71**	**29.14**	**64.70**
GPT-4o-mini [35]	-	71.09	75.58	70.05	21.22	36.42	19.73	36.43
GPT-3.5-Turbo [34]	-	62.03	70.86	58.27	19.96	33.69	18.67	33.63
Text-to-SQL [39]	LLaMA3.2-3B-Instruct	57.80	58.33	56.33	28.84	35.22	12.89	34.79
Dater [53]	LLaMA3.2-3B-Instruct	60.03	58.39	59.10	33.93	39.18	13.58	39.05
CHAIN-OF-TABLE [47]	LLaMA3.2-3B-Instruct	61.09	60.49	60.49	17.14	26.81	12.97	26.67
LoRA [16]	LLaMA3.2-3B-Instruct	55.21	57.84	57.34	36.33	42.51	19.71	42.51
***HyperG* (Ours)**	LLaMA3.2-3B-Instruct	**61.95**	**61.95**	**61.93**	**48.50**	**54.50**	**25.80**	**54.47**
Text-to-SQL [39]	LLaMA3-8B-Instruct	69.72	67.20	69.63	39.24	48.28	20.07	47.79
Dater [53]	LLaMA3-8B-Instruct	73.37	72.42	73.59	48.30	51.74	18.37	51.54
CHAIN-OF-TABLE [47]	LLaMA3-8B-Instruct	78.06	78.08	78.06	36.97	46.09	19.39	46.1
LoRA [16]	LLaMA3-8B-Instruct	66.32	67.16	63.48	49.65	56.78	25.76	56.76
***HyperG* (Ours)**	LLaMA3-8B-Instruct	**79.14**	**80.59**	**78.95**	**55.39**	**61.45**	**27.61**	**61.37**
Text-to-SQL [39]	Gemma-2-2B-It	51.28	51.15	52.81	34.62	45.89	18.39	44.71
Dater [53]	Gemma-2-2B-It	55.68	54.82	57.55	40.95	**55.30**	19.13	**55.11**
CHAIN-OF-TABLE [47]	Gemma-2-2B-It	57.66	61.49	57.56	38.23	45.94	18.52	45.81
LoRA [16]	Gemma-2-2B-It	59.24	59.81	58.02	25.15	33.04	15.59	32.97
***HyperG* (Ours)**	Gemma-2-2B-It	**60.64**	**61.78**	**60.64**	**41.80**	47.70	**22.01**	47.70
Text-to-SQL [39]	Gemma-2-9B-It	70.18	71.21	72.03	50.88	54.46	19.71	51.53
Dater [53]	Gemma-2-9B-It	72.88	71.60	73.31	57.94	61.95	22.91	61.64
CHAIN-OF-TABLE [47]	Gemma-2-9B-It	61.55	**79.59**	71.77	50.83	61.92	**28.16**	61.74
LoRA [16]	Gemma-2-9B-It	75.56	75.69	75.56	31.72	53.42	24.30	53.39
***HyperG* (Ours)**	Gemma-2-9B-It	**79.14**	79.20	**79.13**	**58.54**	**62.60**	28.07	**62.56**

4.1.3 Evaluation Protocol. We evaluate the generation of LLMs enhanced by our proposed *HyperG* framework with respect to the different tasks. For the TFV task, where the answers are either "yes" or "no", we employ **accuracy**, **precision**, **recall**, and **F1 score** as the evaluation metrics. To mitigate the impact of option bias [38, 58] in LLMs, we use a weighted version of all these metrics. For the TQA task, where responses may take the form of sentences or phrases, we adopt the following natural language evaluating metrics.

- **Denotation Accuracy (Denot. Acc.)** [37], following [22, 47], measures how closely a response matches the ground truth answer, regardless of the order of phrases in the answers.
- **ROUGE-N** measures the similarity between the LLM-generated responses and the ground truth answers by comparing overlapping n-grams, used to evaluate text summaries or translations by quantifying shared word sequences. In this paper, we report both ROUGE-1 and ROUGE-2 scores.
- **ROUGE-L** evaluates the similarity between the LLM-generated responses and the ground truth answers by identifying the longest common subsequence (LCS) and is used to assess the fluency and coherence of the generated text.

4.1.4 Implementations Details. For *HyperG*, we explore the learning rates for the LoRA module within the range of {5e-5, 1e-5, 5e-6}, while applying scaling factors for the learning rate in the PHL module (the novel hypergraph neural networks) and the projector from {1, 10, 20}. We search batch sizes from {8, 16, 32}, and conduct experiments over 1 to 4 epochs, utilizing an early stopping strategy. Specifically, for the LoRA module, we fine-tune the Query, Key, and Value projectors with a rank of 8, a LoRA alpha of 32, and a dropout rate of 0.1. For the selected baseline models, we adopt the optimal configurations from the HuggingFace[1] and accelerate inference with vllm 0.5.4[2]. All experiments in this paper are conducted on two NVIDIA A800-SXM4-80GB GPUs. For further details, please refer to our publicly released code linked in the Abstract Section.

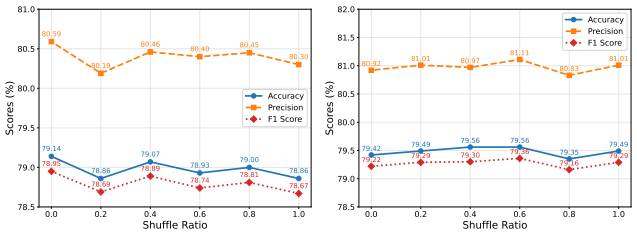
4.2 Task Performance (RQ1)

Table 2 presents a comparison of the performance of our *HyperG* with 13 baseline methods, encompassing both serialization-based methods [1, 16, 34, 35, 45, 55] and operation-based methods [39, 47,

[1] https://huggingface.co/models
[2] https://github.com/vllm-project/vllm

53]. We evaluate their capabilities in structured knowledge using the TFV task on the TabFact dataset [48] and the TQA task on the WiKiTableQuestion dataset [36]. In Table 2, the first group consists of serialized-based methods utilizing various LLMs, while the last four groups compare the performance of our *HyperG* with both state-of-the-art operation-based and serialized-based methods across four backbone LLMs. The following observations can be drawn from the performance results in Table 2.

- **Our *HyperG* outperforms both the operation-based and serialization-based methods based on LLMs.** It can be found in Table 2 that our *HyperG* consistently achieve competing performances across both the TFV and TQA tasks. In general, operations-based methods [39, 47, 53] achieves better outperform the methods [1, 16, 34, 35, 45, 55] of merely prompting LLMs with serialized information, even when the models have been fine-tuned on structural data [55]. This highlights the importance of maintaining structures when reasoning about questions related to structured data. Our proposed *HyperG* utilizes hypergraphs to encode structural information, complementing the powerful natural language capabilities of LLMs. It demonstrates an average improvements of 1.73% and 2.43% in accuracy on TFV and TQA, respectively, when compared to the second-best performances in each group. Upon reviewing the response examples, we found that CHAIN-OF-TABLE [47] encounters difficulties in TQA due to the loss of the question while reasoning over extended chains.
- **Our *HyperG* narrows the performance gap between large and small LLMs, requiring only a modest number of additional parameters.** Table 2 shows that instruction-tuned LLMs [1, 45, 55] with larger parameter sizes achieve better performance with serialization compared to their smaller counterparts. For instance, LLaMA-3.1-70B-Instruct achieves an accuracy of 79.16%, whereas LLaMA-3-8B-Instruct attains only 66.29%. Nevertheless, our *HyperG* intergrated with LLaMA-3-8B-Instruct, which adds approximately **189M** parameters (roughly one-tenth of the parameter difference between LLaMA-3-8B-Instruct and LLaMA-3.1-70B-Instruct), achieves performance comparable to the larger model across both tasks. *HyperG* provides average improved accuracy of 6.22% and 15.72% on the four backbone models regarding the two tasks, respectively. Similarly, *HyperG* based on the Gemma-2-9B-It surpasses its 27B variant by 2.64% and 2.58% in the TFV and TQA tasks with respect to accuracy, respectively. This superiority is generalizable from the TQA task to the TFV task, and is attributed to *HyperG*'s ability in encoding enriched structured knowledge, enabling LLMs to produce more accurate answers. Additionally, the results in Table 2 further demonstrate that our *HyperG* enhances LLMs across parameters sizes ranging from 2B to 9B.
- **Our *HyperG* delivers improvements in performance and training efficiency compared to other SFT methods.** As Supervised Fine-Tuning (SFT) is required in our *HyperG*, we compare it to the other two SFT baseline methods: LoRA [16] and TableLlama [55]. First, *HyperG* demonstrates significant improvements, achieving an average enhancement of 6.13% in accuracy and 15.22% in denotation accuracy [37] over LoRA for the TFV and TQA tasks, respectively. While LoRA is widely recognized as an efficient tool for instruction tuning, it proves less effective when applied to smaller-scale LLMs, such as Gemma-2-2B-It, particularly for reasoning over serialized structured data. This limitation highlights the challenges of adapting LoRA to tasks requiring nuanced structural understanding. Furthermore, when compared to TableLlama [55], which is fine-tuned on a benchmark involving serialized structured knowledge, *HyperG* provides a more efficient solution by fine-tuning on 189M additional parameters with very limited training data (see Table 1). These observations reinforce our earlier assertion that serialization-based methods can undermine the preservation of structures, further highlighting the necessity of *HyperG* in enhancing LLMs to fully utilize such knowledge for improved reasoning.



(a) Shuffle Rows (b) Shuffle Columns

Figure 4: Performances of *HyperG* under different variances of order simulated by shuffling.

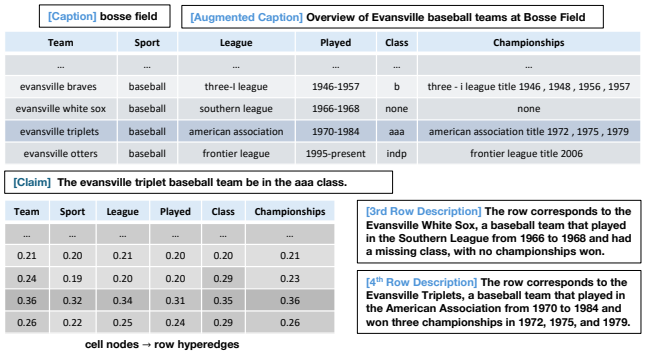
4.3 Order Invariance (RQ2)

In contrast to natural language, where changes in word order can modify the meaning of a sentence, rearranging rows or columns in a table does not affect its meaning. In *HyperG*, this invariance is handled with hyperedges, which represent rows and columns, are inherently unordered within the structure of hypergraphs. To assess how our proposed *HyperG* framework helps the LLM maintain the *Order Invariance* of structural relationships, we shuffle the rows in the test data to evaluate *HyperG*'s robustness to order variations.

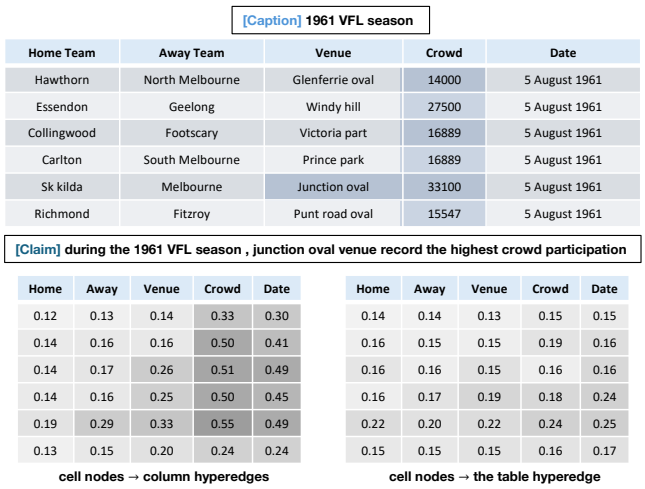
Specifically, we randomly sampled a subset of tables from the TFV testing set and performed shuffling of the rows and columns respectively within each sampled table to introduce variability and evaluate the performance of our proposed *HyperG*. Figure 4 displays the performances of *HyperG* which uses LLaMA3-8B-Instruct as the backbone model, across different shuffle ratios. The x-axis represents the sampling ratio, while the y-axis indicates performance scores with respect to accuracy, precision, and F1 score. As decipted in Figure 4, *HyperG* framework demonstrates stable performance despite variations in row and column order. The accuracy variance of 0.0109 for row shuffling and 0.00043 for columns shuffling, respectively. This stability underscores the robustness of *HyperG* in maintaining structural representation integrity from the perspective of order invariance, thereby validating our previously stated rationale for employing hypergraphs.

4.4 Semantic Consistency and Hierarchical Dependencies (RQ3)

Beyond quantitative metrics, we also conduct qualitative evaluations to investigate whether *HyperG* retains semantic consistency and hierarchical dependencies of structural relationships during



(a) Case 1. Information about the baseball teams at Bosse Field.



(b) Case 2. The results of the 1961 Victorian Football League (VFL).

Figure 5: Visualization of the weights between cell nodes and different hyperedges in two random cases.

reasoning. In Figure 5, we randomly selected two cases and visualized the attention weights between each cell node and the hyperedges associated with the claim's content. This visualization provides insights into how *HyperG* prioritizes and propagates information between table elements and their relevance to the given queries/claims, specifically demonstrating its ability to maintain semantic consistency and hierarchical dependencies.

The semantic consistency in structural realtionships suggests that cells in the same column are similar in semantics. In Case 1, the claim pretains to the class of a team named "evansville triplets". *HyperG* first augmented each row with easy-to-understand natural language descriptions based on the row context, as shown in the bottom right of Figure 5. This augmentation enables *HyperG* to better interpret cells with specific missing values ("none" in the "Class" columns), leading to similar weights for these cells and their counterparts within the same column. In Case 2, the claim queries about the highest crowd participation, which requires examining the "Crowd" column to identify the largest number. Though the crowd numbers for these teams vary, the weights assigned to the column hyperedges are similar thanks to the augmented column descriptions (omitted here) and the Semantics Hypergraph Construction (SHC) in *HyperG*. This is more evident in the weight matrix associated with the table hyperedge shown in the right bottom of 5 (b). Even though the venue names are quite different, the cells in the "Venue" column share similar weights.

The Hierarchical Dependencies refers to the hierarchy across cells, columns, rows, and the whole table. As depicted in Figure 5, the attention of *HyperG* is primarily focused on the cells and the rows/columns related to the claim. This focus extends from cell nodes to row/colum hyperedges, and then the table hyperedges, gradually diminishing in intensity. For example, in Figure 5 (b), the weights assigned to the queried cell "Junction oval", the evidence cell "33100", and the relevant column "Crowd" exceed the average weights of 0.27 and 0.16 in the weight matrices corresponding to the column hyperedges and the table hyperedge, respectively. This demonstrates how the attention mechanism spans the hierarchical structure, emphasizing specific elements within the table.

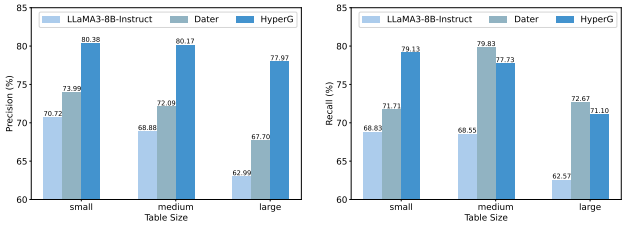
Table 3: The ablation study results of *HyperG* using LLaMA3-8B-Instruct as the base model on the TFV task. Red signifies degradation in percentage.

Methods	TFV			
	%Acc.	%F1	%Prec.	%Recall
***HyperG* (Ours)**	77.20 0.00	77.46 0.00	79.98 0.00	77.20 0.00
w/o PHL	70.96 ↓6.24	70.86 ↓6.60	71.35 ↓8.63	70.96 ↓6.24
w/o PHL, w/ HGNN	72.70 ↓4.50	72.54 ↓4.92	73.03 ↓6.95	72.70 ↓4.50
w/o Inquiry Emb.	72.63 ↓4.57	73.39 ↓4.07	74.22 ↓5.76	74.22 ↓2.98

4.5 Ablation Study (RQ4)

We are also curious about the contribution of each component in *HyperG* contributes to the enhancements of *HyperG*. As shown in Table 3, we successively removed the proposed prompt-attentive hypergraph learning (PHL) module, substituted the PHL module with HGNN [13], and removed the LLM-based argumentation. Note that this ablation study is conducted under hyperparameters setting different from those used for the results in Table 2.

It can be observed from the experimental results in Table 3 that the original framework of our proposed *HyperG* delivers the best performance on verifying the factual knowledge stored in structured data. Firstly, removing the PHL module and directing the semantic embeddings directly to the projector results in a 6.24% reduction in accuracy. Furthermore, to examine the role of hypergraph neural networks in enhancing LLMs' comprehension of structured knowledge, we replace our proposed PHL module with the classical HGNN [13], leading to a 4.50% decrease in performance compared to *HyperG*, as shown in the third row of Table 3. This performance degradation highlights the effectiveness of hypergraphs in representing structured knowledge. Specifically, We attribute this decline to the inability of HGNN to adequately leverage the information encoded in hyperedges for node updates during propagation. Additionally, we explore the impact of incorporating the inquiry embedding in the PHL module. As demonstrated in the last row of Table 3, removing the inquiry embedding causes a substantial 5.76% drop in precision and a more moderate 2.98% decline in recall. This suggests that incorporating inquiry embeddings helps LLMs mitigate the bias toward over-generating positive responses, fostering more cautious reasoning by integrating the essential inquiry within prompts when processing structured data.



(a) Precision (b) Recall

Figure 6: Performances of *HyperG* on tables of different sizes.

4.6 Scalability (RQ5)

Scalability is a critical concern, as large tables pose significant challenges for LLMs, which often struggle to interpret and integrate context from lengthy prompts [30, 54]. To evaluate the performances of *HyperG* on tables of varying sizes, we divided the testing tables in [48] into three classes: **small** (#rows $\leq$ 5 and #columns$\leq$ 5), **medium** (6$\leq$#rows$\leq$10 and 6$\leq$#columns$\leq$ 10), and **large** (#rows$\geq$ 10 and #columns$\geq$ 10). We compare the performance of LLaMA3-8B-Instruct [45], Dater [53], and *HyperG*, with LLaMA3-8B-Instruct serving as the backbone model for all.

As illustrated in Figure 6, with the table sizes increases there are generally declines in all of the models. *HyperG* demonstrates relatively stable performance in precision with a variance of 8.84, surpasses LLaMA3-8B-Instruct of 10.87. For small tables, *HyperG* surpasses Dater in precision and recall by 6.39% and 7.42%, respectively. However, Dater demonstrates superior recall performance compared to *HyperG* for medium and large tables. Upon careful examination and analysis of the true positive cases, we found that this is primarily due to the LLMs in Dater being inclined to generate positive answers. This is also benefits from its effective approach of decomposing queries and tables into sub- questions and sub-tables. The declines of *HyperG* in larger tables are primarily observed in recall. We attribute this to the limited number of hypergraph learning layers and aim to address this issue in the future through more sophisticated graph techniques.

5 Related Works

Large Language Models (LLMs) excel in a broad spectrum of generative tasks [4, 9, 20, 29, 49] but face challenges when processing structured knowledge, such as tabular data [17, 18, 57]. Significant efforts have been made to enhance the capabilities of generative models [26, 27]. Specifically, prior efforts to enhance LLMs' capabilities in handling structured knowledge can be broadly categorized into two main approaches: serialization-based methods [15, 21, 33] and operation-based methods [22, 25, 32, 47, 52]. Serialization-based methods convert structured data into a linear sequence of tokens, similar to how unstructured textual data is formatted for input into LLMs. TableLlama [55], a pioneering approach to enhancing LLMs' performance on tabular data, is fine-tuned on the proposed Table-Instruct dataset, which comprises serialized tables and task-specific instructions for several representative tabular tasks. However, when dealing with highly complex tables or graphs, inquiry-relevant knowledge may be overlooked within the excessively long serialized token sequences [28, 56]. The second category of methods resort to one or a series of operations such as SQL queries to help LLMs reason over structured data [22, 25, 32, 47, 52]. For example, Chain-of-Table [47] iteratively samples operations to select specific portions of the table that are tailored to the inquiry. Dater [52] transforms the sub-questions generated by CodeX [5] into SQL queries, enabling step-by-step multi-hop reasoning. Although these operation-based methods effectively locate the inquiry-relevant knowledge from structured data, they struggle when the target cell or neighboring cells contain missing or incomplete information.

As messages propagate through the structures in Graph Neural Networks (GNNs), efforts have been made to integrate GNNs with LLMs to address structured knowledge more effectively [3, 19, 31, 40, 44, 51]. For example, Chai et al. [3] uses a transformer module to encode the structured knowledge in graphs as the prefix of inputs to the LLMs. Additionally, graphs serve as powerful tools for representing tabular data [6, 23]. HGT [23] explicitly models tables as graphs by connecting various components within the tables to enhance LLM capabilities. Furthermore, HYTREL [6] is particularly relevant to our *HyperG* as it also empolys hypergraphs to represent tabular data, but it overlooks incorporating the semantics of task within prompts during message propagation. To the best of our knowledge, we are the first to leverage hypergraphs to enhance the capabilities of LLMs in handling structured knowledge.

6 Conclusion

In this paper, we present a novel hypergraph-based generation framework, *HyperG*, designed to enhance the understanding and reasoning capabilities of Large Language Models (LLMs) when dealing with knowledge structurally stored. The primary objective of *HyperG* is to tackle the challenges arising from complex structural relationships and data sparsity, such as incomplete cell information, within structured data. By employing a novel prompt-attentive hypergraph learning (PHL) module, *HyperG* effectively propagates information across high-order group dependencies, capturing intricate connections within the data. Comprehensive experiments across three distinct tabular tasks consistently demonstrate the impact of *HyperG* on enhancing the performance of LLMs with different parameter scales. We envision *HyperG* as a solution for enhancing LLMs in a broader range of applications which requires nuanced understanding of structured information.

Acknowledgement

This project has been supported by the Hong Kong Research Grants Council under the General Research Fund (project no. 15200023); Research Impact Fund (project no. R1015-23), the National Natural Science Foundation of China under Grants No.62072257; the Australian Research Council Grants DP22010371, LE220100078; the Guangdong Provincial Department of Education Project (Grant No.2024KQNCX028); CAAI-Ant Group Research Fund, Scientific Research Projects for the Higher-educational Institutions (Grant No.2024312096), Education Bureau of Guangzhou Municipality; Guangzhou-HKUST(GZ) Joint Funding Program (No.2025A03J3957), Education Bureau of Guangzhou Municipality.

References

[1] 2024. Gemma: Open Models Based on Gemini Research and Technology. arXiv:2403.08295 [cs.CL] https://arxiv.org/abs/2403.08295

[2] Song Bai, Feihu Zhang, and Philip HS Torr. 2021. Hypergraph convolution and hypergraph attention. *Pattern Recognition* 110 (2021), 107637.

[3] Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. 2023. Graphllm: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845* (2023).

[4] Junzhe Chen, Xuming Hu, Shuodi Liu, Shiyu Huang, Wei-Wei Tu, Zhaofeng He, and Lijie Wen. 2024. LLMArena: Assessing Capabilities of Large Language Models in Dynamic Multi-Agent Environments. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 13055–13077.

[5] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[6] Pei Chen, Soumajyoti Sarkar, Leonard Lausen, Balasubramaniam Srinivasan, Sheng Zha, Ruihong Huang, and George Karypis. 2024. HYTREL: Hypergraph-enhanced tabular data representation learning. *Advances in Neural Information Processing Systems* 36 (2024).

[7] Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. [n. d.]. You are AllSet: A Multiset Function Framework for Hypergraph Neural Networks. In *International Conference on Learning Representations*.

[8] Lingxi Cui, Huan Li, Ke Chen, Lidan Shou, and Gang Chen. 2024. Tabular Data Augmentation for Machine Learning: Progress and Prospects of Embracing Generative AI. *arXiv preprint arXiv:2407.21523* (2024).

[9] Yunkai Dang, Kaichen Huang, Jiahao Huo, Yibo Yan, Sirui Huang, Dongrui Liu, Mengxi Gao, Jie Zhang, Chen Qian, Kun Wang, et al. 2024. Explainable and interpretable multimodal large language models: A comprehensive survey. *arXiv preprint arXiv:2412.02104* (2024).

[10] 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)*. 4171–4186.

[11] Yihe Dong, Will Sawin, and Yoshua Bengio. 2020. Hnhn: Hypergraph networks with hyperedge neurons. *arXiv preprint arXiv:2006.12278* (2020).

[12] Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Jane Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, Christos Faloutsos, et al. 2024. Large language models (LLMs) on tabular data: Prediction, generation, and understanding-a survey. (2024).

[13] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 3558–3565.

[14] Sungwon Han, Jinsung Yoon, Sercan O Arik, and Tomas Pfister. [n. d.]. Large Language Models Can Automatically Engineer Features for Few-Shot Tabular Learning. In *Forty-first International Conference on Machine Learning*.

[15] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. 2023. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 5549–5581.

[16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=nZeVKeeFYf9

[17] Xuming Hu, Junzhe Chen, Xiaochuan Li, Yufei Guo, Lijie Wen, Philip S. Yu, and Zhijiang Guo. 2024. Towards Understanding Factual Knowledge of Large Language Models. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=9OevMUdods

[18] Sirui Huang, Yanggan Gu, Xuming Hu, Zhonghao Li, Qing Li, and Guandong Xu. 2024. Reasoning Factual Knowledge in Structured Data with Large Language Models. *arXiv preprint arXiv:2408.12188* (2024).

[19] Xuanwen Huang, Kaiqiao Han, Yang Yang, Dezheng Bao, Quanjin Tao, Ziwei Chai, and Qi Zhu. 2024. Can GNN be Good Adapter for LLMs?. In *Proceedings of the ACM Web Conference 2024* (Singapore, Singapore) *(WWW '24)*. Association for Computing Machinery, New York, NY, USA, 893–904. doi:10.1145/3589334.3645627

[20] Jiahao Huo, Yibo Yan, Xu Zheng, Yuanhuiyi Lyu, Xin Zou, Zhihua Wei, and Xuming Hu. 2025. Mmunlearner: Reformulating multimodal machine unlearning in the era of multimodal large language models. *arXiv preprint arXiv:2502.11051* (2025).

[21] Sukriti Jaitly, Tanay Shah, Ashish Shugani, and Razik Singh Grewal. 2023. Towards Better Serialization of Tabular Data for Few-shot Classification. *arXiv preprint arXiv:2312.12464* (2023).

[22] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A General Framework for Large Language Model to Reason over Structured Data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 9237–9251. doi:10.18653/v1/2023.emnlp-main.574

[23] Rihui Jin, Yu Li, Guilin Qi, Nan Hu, Yuan-Fang Li, Jiaoyan Chen, Jianan Wang, Yongrui Chen, and Dehai Min. 2024. HGT: Leveraging Heterogeneous Graph-enhanced Large Language Models for Few-shot Complex Table Understanding. *arXiv preprint arXiv:2403.19723* (2024).

[24] Kezhi Kong, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Chuan Lei, Christos Faloutsos, Huzefa Rangwala, and George Karypis. [n. d.]. OpenTab: Advancing Large Language Models as Open-domain Table Reasoners. In *The Twelfth International Conference on Learning Representations*.

[25] Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. 2023. SheetCopilot: bringing software productivity to the next level through large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*. 4952–4984.

[26] Qian Li, Tri Dung Duong, Zhichao Wang, Shaowu Liu, Dingxian Wang, and Guandong Xu. 2021. Causal-aware generative imputation for automated underwriting. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3916–3924.

[27] Qian Li, Zhichao Wang, Haiyang Xia, Gang Li, Yanan Cao, Lina Yao, and Guandong Xu. 2024. HOT-GAN: Hilbert Optimal Transport for Generative Adversarial Network. *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[28] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469* (2024).

[29] Zhonghao Li, Xuming Hu, Aiwei Liu, Kening Zheng, Sirui Huang, and Hui Xiong. 2024. Refiner: Restructure retrieval content efficiently to advance question-answering capabilities. *arXiv preprint arXiv:2406.11357* (2024).

[30] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics* 12 (2024), 157–173. doi:10.1162/tacl_a_00638

[31] Pengfei Liu, Yiming Ren, Jun Tao, and Zhixiang Ren. 2024. Git-mol: A multi-modal large language model for molecular science with graph, image, and text. *Computers in biology and medicine* 171 (2024), 108073.

[32] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-Play Compositional Reasoning with Large Language Models. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=HtqnVSCj3q

[33] Dehai Min, Nan Hu, Rihui Jin, Nuo Lin, Jiaoyan Chen, Yongrui Chen, Yu Li, Guilin Qi, Yun Li, Nijun Li, et al. 2024. Exploring the impact of table-to-text methods on augmenting llm-based question answering with domain hybrid data. *arXiv preprint arXiv:2402.12869* (2024).

[34] OpenAI. 2023. OpenAI's GPT-3.5 Turbo. https://platform.openai.com/docs/models/gpt-3-5-turbo Accessed: October 6, 2024.

[35] OpenAI. 2024. GPT-4o mini: advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence

[36] Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305* (2015).

[37] 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)*, Chengqing Zong and Michael Strube (Eds.). Association for Computational Linguistics, Beijing, China, 1470–1480. doi:10.3115/v1/P15-1142

[38] Pouya Pezeshkpour and Estevam Hruschka. 2024. Large Language Models Sensitivity to The Order of Options in Multiple-Choice Questions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 2006–2017. doi:10.18653/v1/2024.findings-naacl.130

[39] Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498* (2022).

[40] Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh Chawla, and Chao Huang. 2024. A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6616–6626.

[41] Yutong Shao and Ndapandula Nakashole. 2024. On Linearizing Structured Data in Encoder-Decoder Language Models: Insights from Text-to-SQL. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 131–156.

[42] Ananya Singha, José Cambronero, Sumit Gulwani, Vu Le, and Chris Parnin. [n. d.]. Tabular Representation, Noisy Operators, and Impacts on Table Structure Understanding Tasks in LLMs. In *NeurIPS 2023 Second Table Representation Learning Workshop*.

[43] Xiaoyu Tan, Haoyu Wang, Xihe Qiu, Yuan Cheng, Yinghui Xu, Wei Chu, and Yuan Qi. 2024. Struct-X: Enhancing Large Language Models Reasoning with Structured Data. *arXiv preprint arXiv:2407.12522* (2024).

[44] Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V Chawla, and Panpan Xu. 2024. Graph neural prompting with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 19080–19088.

[45] Hugo Touvron et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.CL] https://arxiv.org/abs/2407.21783

[46] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[47] Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. Chain-of-Table: Evolving Tables in the Reasoning Chain for Table Understanding. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=4L0xnS4GQM

[48] Jianshu Chen Yunkai Zhang Hong Wang Shiyang Li Xiyou Zhou Wenhu Chen, Hongmin Wang and William Yang Wang. 2020. TabFact : A Large-scale Dataset for Table-based Fact Verification. In *International Conference on Learning Representations (ICLR)*. Addis Ababa, Ethiopia.

[49] Yang Wu, Yao Wan, Hongyu Zhang, Yulei Sui, Wucai Wei, Wei Zhao, Guandong Xu, and Hai Jin. 2024. Automated Data Visualization from Natural Language via Large Language Models: An Exploratory Study. *Proc. ACM Manag. Data* 2, 3, Article 115 (May 2024), 28 pages. doi:10.1145/3654992

[50] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. 2019. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems* 32 (2019).

[51] Haoran Yang, Xiangyu Zhao, Sirui Huang, Qing Li, and Guandong Xu. 2024. Latex-gcl: Large language models (llms)-based data augmentation for text-attributed graph contrastive learning. *arXiv preprint arXiv:2409.01145* (2024).

[52] Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 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* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 174–184. doi:10.1145/3539618.3591708

[53] Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 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* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 174–184. doi:10.1145/3539618.3591708

[54] Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 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* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 174–184. doi:10.1145/3539618.3591708

[55] Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2024. TableLlama: Towards Open Large Generalist Models for Tables. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 6024–6044.

[56] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=RkRrPp7GKO

[57] Yilun Zhao, Haowei Zhang, Shengyun Si, Linyong Nan, Xiangru Tang, and Arman Cohan. 2023. Investigating Table-to-Text Generation Capabilities of Large Language Models in Real-World Information Seeking Scenarios. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, Mingxuan Wang and Imed Zitouni (Eds.). Association for Computational Linguistics, Singapore, 160–175. doi:10.18653/v1/2023.emnlp-industry.17

[58] Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2024. Large Language Models Are Not Robust Multiple Choice Selectors. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=shr9PXz7T0

[59] Max Zhu, Siniša Stanivuk, Andrija Petrovic, Mladen Nikolic, and Pietro Lio. [n. d.]. Incorporating LLM Priors into Tabular Learners. In *NeurIPS 2023 Second Table Representation Learning Workshop*.