Improving Table Understanding with LLMs and Entity-Oriented Search

Thi-Nhung Nguyen[1], Hoang Ngo[2], Dinh Phung[1], Thuy-Trang Vu[1], Dat Quoc Nguyen[2]
[1]Monash University, [2]Qualcomm AI Research*
{nhung.thinguyen,dinh.phung,trang.vu1}@monash.edu, {hoangngo,datnq}@qti.qualcomm.com

Abstract

Our work addresses the challenges of understanding tables. Existing methods often struggle with the unpredictable nature of table content, leading to a reliance on preprocessing and keyword matching. They also face limitations due to the lack of contextual information, which complicates the reasoning processes of large language models (LLMs). To overcome these challenges, we introduce an entity-oriented search method to improve table understanding with LLMs. This approach effectively leverages the semantic similarities between questions and table data, as well as the implicit relationships between table cells, minimizing the need for data preprocessing and keyword matching. Additionally, it focuses on table entities, ensuring that table cells are semantically tightly bound, thereby enhancing contextual clarity. Furthermore, we pioneer the use of a graph query language for table understanding, establishing a new research direction. Experiments show that our approach achieves new state-of-the-art performances on standard benchmarks WikiTableQuestions and TabFact.

1 Introduction

Tables are widely used to systematically organize and present data. Understanding tables is key to addressing various downstream tasks, such as table-based question answering (Wang et al., 2023a; Lin et al., 2023). As illustrated in Figure 1, the goal is to extract the relevant information from the table to provide accurate answers to users' questions. Recent research has explored using Large Language Models (LLMs) to solve tabular data problems by leveraging their strong performance with prompting (Yang et al., 2023; Nguyen et al., 2023; Xie et al., 2023). One common approach is to convert a natural language question into a structured query (e.g., SQL) and then execute the query on tables to retrieve the final answer (Lin et al., 2023; Gemmell & Dalton, 2023; Wang et al., 2024; Nahid & Rafiei, 2024; Liu et al., 2024; Kong et al., 2024).

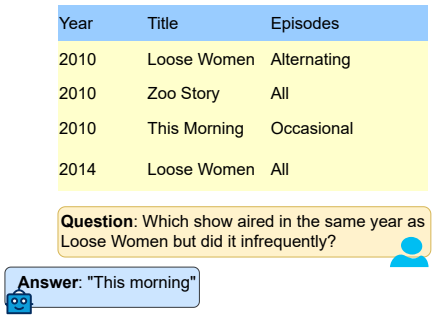

Figure 1: A table question answering example on a "show" table.

Although tabular data allows users to organize and display information logically in real-life scenarios, it presents unique challenges for LLM-based methods. One of the major challenges is the unpredictable nature of the content and formatting within table cells. For instance, in a column labeled "*address*", one cell may contain a full address with the street, city, and zip code (e.g., "*123 Main St, Springfield, 12345*"), while another cell may only list the

*Qualcomm Vietnam Company Limited. Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

city name (e.g., "*Springfield*") or even be blank. This inconsistency hinders the performance of query search-based methods. To mitigate this challenge, some works have implemented preprocessing techniques, such as normalizing numbers and dates, inputting missing values, removing outliers, and transforming tables into more suitable formats (Gemmell & Dalton, 2023; Zhao et al., 2023; Liu et al., 2024; Nahid & Rafiei, 2024). However, these preprocessing techniques demand significant effort in data analysis, are difficult to adapt to unseen tables, and can lead to unintended consequences if not applied properly. Improper preprocessing may result in information loss, creation of sparse matrices, and disruption of the table's original structure.

Another challenge with tabular data is the requirement for complex reasoning due to limited contextual information. This issue arises because table cells often contain only keywords or short phrases rather than full sentences, and the relationships between cells are frequently implicit rather than explicitly stated. For instance, consider a table that includes a "*employee*" column containing names like "*Alice Johnson*" and "*Bob Smith*", along with a "*status*" column indicating their employment status (e.g., "*active*," "*on leave*", or "*terminated*"). Without additional context, interpreting the precise meaning of these statuses can be difficult. To address this challenge, some recent studies have focused on chain-of-thought (COT) reasoning, however, this approach requires multi-step reasoning across the entire table, which can be computationally intensive and resource-demanding (Wei et al., 2022; Yao et al., 2023; Chen et al., 2023; Wang et al., 2024). Alternative approaches include rephrasing questions, breaking them into sub-queries, or retrieving related columns or rows (Kong et al., 2024; Patnaik et al., 2024; Ye et al., 2023). However, these methods often rely heavily on keyword matching. As a result, the retrieved table cells may be irrelevant to one another, increasing the effort required for LLMs to extract the final answer.

In this paper, we propose a novel approach based on entity-oriented search to enhance table understanding with LLMs. Our method effectively leverages the semantic similarity between the question and the data stored in the table, along with the implicit relationships between cells. Our approach alleviates the strict data format requirements and the reliance on keyword matching found in related works (Gemmell & Dalton, 2023; Kong et al., 2024; Chen et al., 2024b).

Firstly, we focus on identifying entities stored within the table by prompting the LLM. We assume the table contains entities and relations, which may be organized differently depending on the table's structure. An entity can represent a real-world object, person, place, or concept, each defined by its attributes–data cells that provide detailed information. For example, in Figure 1, each show is an entity with attributes like "*Year*", "*Title*", and "*Episodes*". A relationship example is the relation between each entity and its "*Year*", indicating *when a show aired*. By structuring data this way, we aim to establish stronger relationships and constraints among relevant table cells, thereby clarifying the context of each entity. This approach contrasts with the original table data, where excessive information often introduced noise and confusion, making it difficult to determine context.

Secondly, we propose an entity-oriented search approach to extract relevant entities and attributes. Specifically, we integrate methods from full-text search, semantic search, and graph search. Full-text search allows for keyword-based searching, while our semantic search method focuses on the semantic similarities between the entities and attributes in the questions and those stored in the table. Additionally, we use graph queries to represent questions and to query the graphs formed by entities and relations, effectively leveraging these relations to achieve more accurate search results. We expect that this approach will minimize the need for data preprocessing and keyword matching, adapting effectively to various ways of phrasing questions since the underlying meanings and relationships between entities and attributes remain consistent, even when different terms are used. Furthermore, we are pioneering the use of graph query language (Cypher) for table understanding by providing the LLM with both the graph schema and the question as input, enabling it to transform the question into a Cypher query statement.[1]

[1] `https://neo4j.com/docs/Cypher-manual` (Francis et al., 2018).

Our contributions are summarized as follows: **(I)** We propose an entity-oriented search that effectively leverages the semantic similarities between the entities and attributes in the questions and those in the table, along with the implicit relationships between table cells. This minimizes the need for data preprocessing and keyword matching. **(II)** Our search approach focuses on table entities, ensuring that table cells are semantically tightly bound. This enhances contextual clarity and strengthens relationships between relevant cells. **(III)** We are the first to explore a graph query language (Cypher) to enhance table understanding, introducing a new research direction. **(IV)** Comprehensive experiments show that our approach achieves state-of-the-art performances.

2 Related Works

Table understanding encompasses a wide range of tasks such as question answering (QA). Many early works focus on fine-tuning BERT (Devlin et al., 2019) to serve as a table encoder for these tasks, such as TAPAS (Herzig et al., 2020), Table-BERT (Chen et al., 2020), TABERT (Liu et al., 2022), TURL (Deng et al., 2022), TUTA (Wang et al., 2021), and TABBIE (Iida et al., 2021). Recently, the superior performance of large language models (LLMs) with prompting has shifted research focus towards exploring their potential in processing tabular data. A straightforward approach is to concatenate task descriptions with the serialized table as a string and input them into a LLM to generate a text-based response (Marvin et al., 2023; Cheng et al., 2023; Sui et al., 2024). Additionally, some works have further enhanced performance by utilizing few-shot and curated examples (Cheng et al., 2023; Narayan et al., 2022; Chen, 2023).

To effectively address table-based tasks with large language models (LLMs), recent research increasingly employs external tools instead of relying solely on general text processing. Some works propose generating Python programs and then executing them to extract relevant data (Chen et al., 2023; Gao et al., 2023). Similarly, some works propose using text-to-SQL conversion to extract answers (Rajkumar et al., 2022; Cheng et al., 2023; Ni et al., 2023). However, this approach struggles with complex cases involving intricate tables due to the limitations of the single-pass generation process. In this setup, LLMs cannot modify the table in response to specific questions, necessitating reasoning over a static table. In contrast, a chain-of-thought (CoT)-based approach reasons step by step before providing an answer (Chen et al., 2023; Zhao et al., 2024; Yang et al., 2024). To enhance CoT, several methods have been proposed, such as breaking down the question into sub-problems (Zhou et al., 2023; Khot et al., 2023), employing a table-filling procedure (Ziqi & Lu, 2023), and incorporating preprocessing operations and SQL execution (Wang et al., 2024; Nguyen et al., 2025).

Additionally, self-consistency (SC), proposed by Wang et al. (2023b), is another widely adopted technique in recent state-of-the-art research. SC involves sampling a diverse set of reasoning paths from LLMs and selecting the most consistent answer by marginalizing over these paths (Ye et al., 2023; Cheng et al., 2023; Liu et al., 2024). However, a common limitation of both CoT and SC methods is their requirement for a substantial number of reasoning samples from LLMs. For example, Cheng et al. (2023) generate 50 samples for each question, while Ye et al. (2023); Liu et al. (2024) require over 100 samples. This results in slower performance and higher computational costs, making them almost impractical for real-world implementation.

3 Our Approach

In this section, we introduce a novel framework, **TUNES**, to improve **t**able **un**derstanding with **e**ntity-oriented **s**earch. Given a table and a related question, the task of TUNES is to generate an answer based on relevant information from the table.

Figure 2 illustrates the architecture of our TUNES, which consists of three main components: (1) Entity Identification, (2) Entity-Oriented Search, and (3) LLM-based Answer Generation. Entity Identification: To begin, we focus on extracting entities from the table. We utilize a LLM to analyze the table's structure, such as primary keys, column names, and row names,

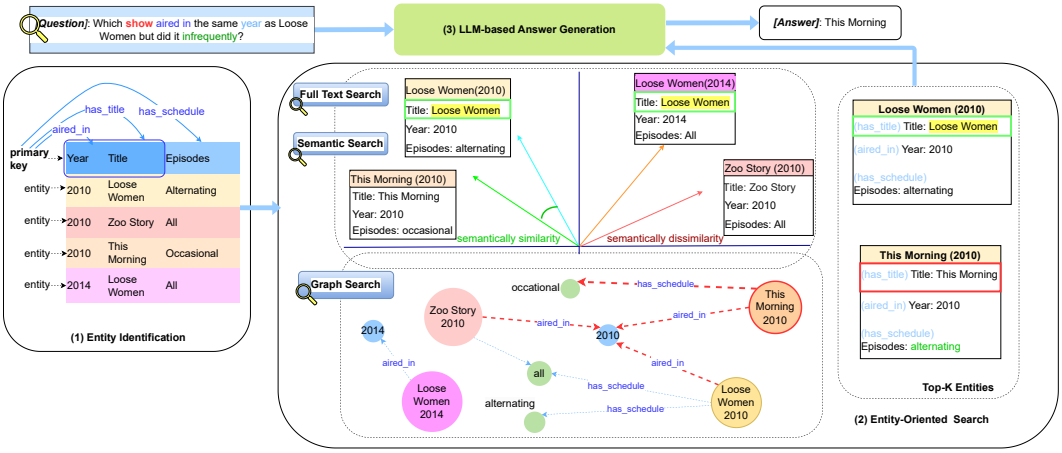

Figure 2: Overview of our proposed framework **TUNES**. The embedding space and the graph are simplified for illustration purposes.

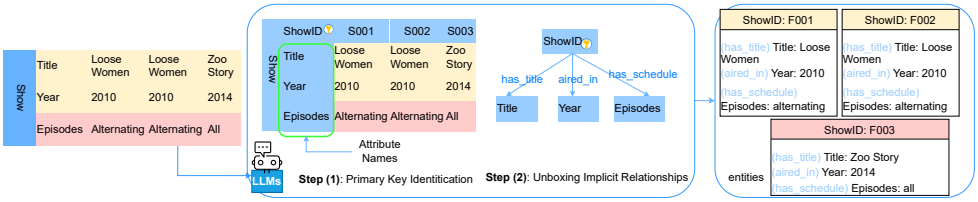

Figure 3: Entity Identification Example.

to identify attributes and relationships, thereby identifying entities. Entity-Oriented Search: We then construct a graph from these entities, attributes, and their relationships. During this process, we remove attributes without values and merge those sharing values to streamline the graph. Simultaneously, we embed the entities, attributes, and user questions into an embedding space, enabling an entity-oriented search that integrates full-text, vector, and graph search. LLM-based Answer Generation: Finally, we extract the top K most relevant entities to the question and input them into the LLM to generate accurate answers.

3.1 Entity Identification

This subsection describes our approach to detecting attributes and relationships, which enables the identification of entities within a data table.

The first step is to find the primary key, which can be a single attribute or a combination of attributes that uniquely identifies each entity, typically found in the column or row names. To locate the primary key, we simply use the first h rows and h columns as input to prompt the LLM. This process enables us to identify the corresponding attribute column or row and subsequently recognize the entities. For example, in Figure 1, combining "*title*" and "*year*" could serve as the primary key to distinguish the entities within the table. In this case, "*year*," "*title*," and "*episodes*" are attributes, and each entity is identified by its attributes across these columns. Note that, if the primary key is absent from the table, the LLM is responsible for generating it and specifying its position (See our prompt in Table 4 in Appendix A).

To unbox the implicit relationships in the table, we aim to explore how each entity is related to its attributes. We achieve this by providing the primary key and attribute names to the LLM, prompting it to generate the relationships (See our prompt in Table 5 in Appendix A). Figure 3 illustrates an example of the entity identification process.

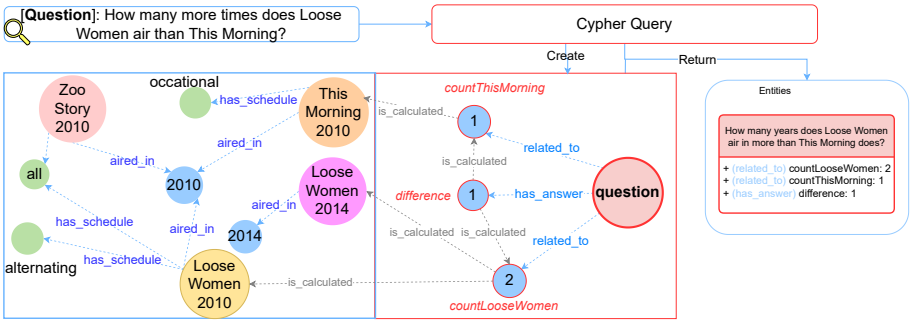

Figure 4: Cypher Query Execution Process. To answer the question "How many more times does Loose Women air than This Morning?", the Cypher query execution process first calculates the number of times "Loose Women" airs, creating a new attribute called "countLooseWomen". It then performs a similar calculation for "This Morning", generating an attribute called "countThisMorning". The process calculates the difference between "countLooseWomen" and "countThisMorning", creating an attribute named "difference". Note that the relationship "is_calculated" is used solely to illustrate the calculation process. Next, the process creates a new entity representing the question and determines how this entity relates to the newly created attributes, using relationships such as "related_to" and "has_answer". Finally, the process returns a complete entity that effectively answers the question by providing a comprehensive analysis of the attributes' impact. See the corresponding Cypher query for the input question in Appendix B. *We pioneer the use of the graph query language Cypher to improve table understanding.*

3.2 Entity-Oriented Search

Graph Search: For each table, we construct a graph $\mathcal{G} = \mathcal{N}, \mathcal{E}$, where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ is the set of edges. $\mathcal{N}$ represents the entities and attributes stored in the table, formed by the union of entity nodes and attribute nodes. To preserve the position of nodes in the table, we store their table addresses as properties of each node. $\mathcal{E}$ represents the relationships within the table, established by the relationships between the entity and its attributes.

Node disambiguation: We exclude attributes that lack values, which results in sparse data in the original table. Also, some attributes may refer to the same value or have synonymous meanings. For example, in Table 1, cells (1,0), (2,0), and (3,0) all correspond to the year "*2010*", and these should be merged into a single node. We accomplish this by filtering based on the semantic similarity score between two node names in an embedding space, as defined in the next paragraph. To query the graph, we provide the LLM with both the graph schema and the question as input, prompting it to convert the question text into a Cypher query statement (See our prompt in Table 6 in Appendix A). We then execute this Cypher query on the graph to search relevant entities and attributes. In addition to searching for relevant entities and verifying the constraints that must be satisfied in questions, the Cypher query execution process enables complex calculations on attributes and entities. It allows for the automatic generation of new entities along with their associated attributes, as illustrated in Figure 4, making graph search a crucial component of TUNES.

Semantic Search: While constructing the graph, we simultaneously map both the entity nodes and the user's question into an embedding space using a text embedding model. This is to calculate the semantic search scores by determining the similarity between the question and the entities. We measure this similarity using the cosine similarity between their representation vectors. Specifically, we obtain the question's representation by directly inputting the question text into the text embedding model. For each entity node, we generate a representation by feeding the embedding model a concatenation of node names from a sub-graph of depth d, with the entity node as the root.

Full-text Search: Full-text search determines relevance by considering the number of keyword matches and their frequency. We use the BM25 algorithm to rank table entities based on how well they align with the given question (Robertson & Zaragoza, 2009). First, we create a search document for each entity by combining its primary key and attributes. Then, we calculate a search score for each entity using BM25, comparing the question to its search document, with higher scores indicating a stronger match to the question.

Entity-Oriented Search Approach: Entity-Oriented search combines full-text search, semantic search, and graph search to provide more relevant search results. Specifically, we extract the top K relevant entities by calculating a weighted sum of their full-text and semantic search scores, while also retrieving entities and attributes output from the Cypher query execution process. Full-text search quickly identifies exact terms or closely related ones, even in large datasets. Semantic search focuses on understanding the underlying context or meaning of the question. Graph search addresses constraints and implicit relationships within the question. This approach ensures that even if a question does not precisely match a relevant entity, the relevant meanings and satisfied relational constraints within the graph will still allow it to be retrieved.

3.3 LLM-based Answer Generation

The top K relevant entities are incorporated into a prompt and input into the LLM to generate a response (See our prompt in Table 7 in Appendix A). Each entity is treated as a paragraph, with the primary key serving as a heading that introduces the main topic, while the attributes and relations provide further details and elaboration. Combining these entities is akin to constructing a document, where each paragraph, representing an entity, can stand alone but may also connect to other paragraphs if two entities share the same attribute. This approach aligns well with LLMs, which are extensively trained on text documents, reducing the need for complex reasoning and resulting in a more accurate answer.

4 Experiment Setup

Datasets & Metric: Following previous works (Wang et al., 2024), we conduct experiments on the benchmark datasets WikiTableQuestions—a question answering dataset over semi-structured tables (Pasupat & Liang, 2015) and TabFact—a dataset for table-based fact verification (Chen et al., 2020). See Appendix C for the statistics of their test sets.

We employ the binary classification accuracy for TabFact and the official denotation accuracy for WikiTableQuestions (Pasupat & Liang, 2015).

Baselines: We compare TUNES with strong baselines, including: **(I)** Methods based on self-consistency (SC) or chain-of-thought (CoT), which require a substantial number of reasoning inferences with LLMs: **Dater** (Khot et al., 2023), **BINDER** (Cheng et al., 2023), **CHAIN-OF-TABLE** (Wang et al., 2024) and **DP&PYAGENT** (Liu et al., 2024). **(II)** Methods without SC and CoT, which only require a few number of LLM inferences: **TEXT2SQL** (Rajkumar et al., 2022), **ChatGPT** (Jiang et al., 2023), **StructGPT** (Jiang et al., 2023), **TableRAG** (Chen et al., 2024b) and **TabSQLify** (Nahid & Rafiei, 2024).

Implementation Details: Following previous works, we use *GPT-3.5-turbo* as the LLM. Additionally, we report our scores with other LLMs, including *GPT-4o-mini* and the open-weight LLMs *Llama-3.1-8B-Instruct* and *Llama-3.1-70B-Instruct* (Dubey et al., 2024). We use the OpenAI API to run inferences with *GPT-3.5-turbo* and *GPT-4o-mini*, while vLLM (Kwon et al., 2023) is used for inference with *Llama-3.1-8B-Instruct* and *Llama-3.1-70B-Instruct*. For all these models, we set the temperature values to 0.4 for Text-to-Cypher generation and 0.0 for answer generation.

To extract primary keys (see Section 3.1), we set h to 5. We use Neo4J to interact with graphs (see Section 3.2).[2] We merge attribute nodes through exact matching and filter them by a

[2] https://neo4j.com/docs/Cypher-manual

	Approach	WikiTQ	TabFact
GPT-3.5-turbo	Dater (Khot et al., 2023) **[SC]** (*)	52.8	78.0
	BINDER (Cheng et al., 2023) **[SC]** (*)	56.7	79.2
	CHAIN-OF-TABLE (Wang et al., 2024) **[CoT]** (*)	59.9	80.2
	DP&PYAGENT (Liu et al., 2024) **[SC]** (*)	65.5	80.0
	Our TUNES$_{\text{GPT-3.5-turbo}}$ **[CoT]**	**68.5**	**81.5**
	StructGPT (Jiang et al., 2023)	48.4	–
	ChatGPT (Jiang et al., 2023)	51.8	–
	TableRAG (Chen et al., 2024b)	57.0	–
	TEXT2SQL (Rajkumar et al., 2022)	52.9	64.7
	TabSQLify (Nahid & Rafiei, 2024)	64.7	79.5
	Our TUNES$_{\text{GPT-3.5-turbo}}$	**64.9**	**79.8**
Others.	CHAIN-OF-TABLE$_{\text{GPT-4o-mini}}$ **[CoT]** (*)	70.4	85.8
	DP&PYAGENT$_{\text{GPT-4o-mini}}$ **[SC]** (*)	74.7	89.9
	Our TUNES$_{\text{GPT-4o-mini}}$	72.3	84.7
	Our TUNES$_{\text{GPT-4o-mini}}$ **[CoT]**	**75.4**	**90.4**
	CHAIN-OF-TABLE$_{\text{Llama-3.1-70B-Instruct}}$ **[CoT]**	70.1	85.6
	DP&PYAGENT$_{\text{Llama-3.1-70B-Instruct}}$ **[SC]**	67.9	85.1
	Our TUNES$_{\text{Llama-3.1-70B-Instruct}}$	75.4	85.6
	Our TUNES$_{\text{Llama-3.1-70B-Instruct}}$ **[CoT]**	**75.7**	**87.5**
	CHAIN-OF-TABLE$_{\text{Llama-3.1-8B-Instruct}}$ **[CoT]**	56.0	49.6
	DP&PYAGENT$_{\text{Llama-3.1-8B-Instruct}}$ **[SC]**	57.3	63.8
	Our TUNES$_{\text{Llama-3.1-8B-Instruct}}$	54.1	68.1
	Our TUNES$_{\text{Llama-3.1-8B-Instruct}}$ **[CoT]**	**57.8**	**71.9**

Table 1: Performance results on the WikiTableQuestions (WikiTQ) and TabFact test sets. [SC] and [CoT] denote approaches based on self-consistency and chain-of-thought, respectively. In rows 2–15, results for previous methods are taken from their respective works, except for Dater, BINDER, CHAIN-OF-TABLE, and DP&PYAGENT, which are marked with (*), are taken from Nguyen et al. (2025). In the last 8 rows, we run the official implementations of CHAIN-OF-TABLE (`https://github.com/google-research/chain-of-table`) and DP&PYAGENT (`https://github.com/Leolty/tablellm`) using Llama-3.1-8B-Instruct and Llama-3.1-70B-Instruct.

cosine similarity greater than 0.95. For text embeddings, we use *bge-m3* (Chen et al., 2024a) as our embedding model, setting $d = 2$ when embedding nodes (see Section 3.2 - Semantic Search). For the entity-oriented search approach (see Section 3.2), the weight of each search component is set to 1, and for the top K most relevant entities, we set K to 50.

TUNES [CoT]: We also develop a variant of **TUNES with CoT**. Specifically, we require the model to answer the question step by step by iteratively executing the search and answer processes. Simultaneously, the LLM-based Answer Generation is required to reason step by step before delivering the final answer. The maximum number of iterations is set to 3.

5 Results

5.1 Main Results

Using GPT-3.5-turbo as the base LLM, as shown in Table 1, when compared to baselines employing SC and CoT, our TUNES$_{\text{GPT-3.5-turbo}}$ [CoT] outperforms the previous state-of-the-art (SOTA) model DP&PYAGENT by 3.0% points on WikiTableQuestions and 1.5% on TabFact. Notably, TUNES$_{\text{GPT-3.5-turbo}}$ [CoT] requires fewer intermediate responses from the LLM (the total is 8, including 2 for table analysis, 3 for searching and 3 for answering), compared to CHAIN-OF-TABLE (25), Binder (50), Dater (100) (Wang et al., 2024), and DP&PYAGENT (50-150) (Liu et al., 2024). Note that the average time for performing both semantic search and full-text search is very small, only at 0.06 seconds per query on a CPU. Thus, in TUNES, almost all of the running time is spent on prompting LLMs. As a result, using entity-oriented search has allowed us to reduce the LLM inference cost per

question by a factor of $25/8 \simeq 3$ to $150/8 \simeq 18$. In addition, our original $\text{TUNES}_{\text{GPT-3.5-turbo}}$ surpasses all baselines not utilizing CoT and SC, outperforming TEXT2SQL by 12% on WikiTableQuestions and 15.1% on TabFact, while achieving a 0.2+% improvement over the previous SOTA model TabSQLify on both datasets.

In the last 12 rows of Table 1, we benchmark TUNES against SOTA methods, including CHAIN-OF-TABLE and DP&PYAGENT, across different LLMs, including Llama-3.1-8B-Instruct, Llama-3.1-70B-Instruct, GPT-4o-mini to further evaluate the adaptability of our approach. The results show that the original TUNES demonstrates competitive performance, while TUNES [CoT] consistently outperforms both CHAIN-OF-TABLE and DP&PYAGENT across all examined LLMs and datasets.

The performance of $\text{TUNES}_{\text{GPT-4o-mini}}$ remains consistent with the results previously observed on GPT-3.5-turbo. In detail, compared to $\text{DP\&PYAGENT}_{\text{GPT-4o-mini}}$ [SC], $\text{TUNES}_{\text{GPT-4o-mini}}$ is 2.4% points lower on WikiTQ and 5.2% points lower on TabFact. However, $\text{TUNES}_{\text{GPT-4o-mini}}$ surpasses $\text{CHAIN-OF-TABLE}_{\text{GPT-4o-mini}}$ [CoT] on WikiTQ by 1.9% points, despite being 1.1% points lower on TabFact. When augmented with CoT reasoning, $\text{TUNES}_{\text{GPT-4o-mini}}$ [CoT] surpasses both baselines, exceeding $\text{CHAIN-OF-TABLE}_{\text{GPT-4o-mini}}$ [CoT] by 5.0% points on WikiTQ and 4.6% points on TabFact, while also outperforming $\text{DP\&PYAGENT}_{\text{GPT-4o-mini}}$ [SC] by 0.7% points on WikiTQ and 0.5% points on TabFact.

For open-source LLMs, TUNES demonstrates better adaptability than the baselines. $\text{TUNES}_{\text{Llama-3.1-8B-Instruct}}$ notably outperforms $\text{CHAIN-OF-TABLE}_{\text{Llama-3.1-8B-Instruct}}$ [CoT] by 18.5% points and $\text{DP\&PYAGENT}_{\text{Llama-3.1-8B-Instruct}}$ [SC] by 4.3% points on TabFact. In addition, $\text{TUNES}_{\text{Llama-3.1-70B-Instruct}}$ even surpass both baselines on both datasets, demonstrating a 5.3% points improvement on WikiTQ over $\text{CHAIN-OF-TABLE}_{\text{Llama-3.1-70B-Instruct}}$ [CoT] and 7.5% points on WikiTQ and 0.5% on Tabfact over $\text{DP\&PYAGENT}_{\text{Llama-3.1-70B-Instruct}}$ [SC]. Meanwhile, $\text{TUNES}_{\text{Llama-3.1-8B-Instruct}}$ [CoT] and $\text{TUNES}_{\text{Llama-3.1-70B-Instruct}}$ [CoT] consistently achieve SOTA performance on both datasets. All obtained results demonstrate the effectiveness of our proposed approach.

Overall, TUNES without CoT requires 6× to 36× fewer LLM calls than the baselines, resulting in significant computational savings. Despite this efficiency, TUNES still achieves competitive performance across four different LLMs, demonstrating its generalizability and effectiveness. Meanwhile, when combined with CoT, TUNES requires 3×–18× fewer LLM calls, while significantly outperforming state-of-the-art baselines such as CHAIN-OF-TABLE and DY&PYAGENT, with statistical significance ($p < 0.01$)[3] on both the WikiTQ and TabFact datasets.

5.2 Ablation Study

To assess the impact of each proposed component of TUNES, we conduct an ablation study by evaluating different ablated versions of $\text{TUNES}_{\text{GPT-4o-mini}}$. Given budget constraints, we evaluate these ablated variants on a randomly selected subset of **1,000** questions from the WikiTableQuestions test set. Table 2 presents the results for the full-component $\text{TUNES}_{\text{GPT-4o-mini}}$ alongside the ablation study results on this subset.

Approach	WikiTQ
$\text{TUNES}_{\text{GPT-4o-mini}}$	72.0
without Entity Identification	61.3
without Graph Search	62.0
without Semantic Search	69.7
without Full-text Search	70.2

Table 2: Ablation results for $\text{TUNES}_{\text{GPT-4o-mini}}$.

Without Entity Identification: We exclude the entity identification component from TUNES. Instead of searching for entities, the search strategy shifts from an entity-oriented approach to a row-oriented one. That is, each row becomes a search document for both full-text and semantic searches and serves as a node in the graph without any relationships. The results indicate that this shift from entity-oriented to row-oriented search reduces accuracy, with a drop of 72.0% - 61.3% = 10.7% compared to the full TUNES. This shows that clarifying the

[3]Based on one-sided McNemar tests from in-house experiments.

Error Type	Description	%	Exa.
[Entity] Table structure identification	The LLM incorrectly identifies the positions of primary keys and attributes.	4%	D.1.1
[Search] Insufficient entity retrieval	Entity-oriented search fails to retrieve an adequate number of entities required to answer the question.	8%	D.2.1
[Search] Incorrect self-generated entity	Cypher execution process generates inaccurate information.	50%	D.2.2
[Answer] Comparative error	The LLM incorrectly compares quantities such as distance or time.	8%	D.3.1
[Answer] Numerical error	The LLM performs incorrect calculations.	24%	D.3.2
[Answer] Logical error	The LLM incorrectly extracts and utilizes provided information in the reasoning process.	20%	D.3.3
[Answer] Others	N/A	2%	

Table 3: Error types by components—Entity Identification (denoted as [Entity]), Entity-Oriented Search (denoted as [Search]), and LLM-based Answer Generation (denoted as [Answer])—in $\text{TUNES}_{\text{Llama-3.1-70B-Instruct}}$. The total percentage does not add up to 100% as some samples contain more than one error. See error examples (Exa.) in Appendix D.

context for table entities, along with the relationships between entities and their attributes, enhances performance, underscoring the effectiveness of the proposed approach.

Without Graph Search: Graph search is excluded from the search strategy, leaving the task of searching solely to full-text and semantic searches. As shown in Table 2, the removal of graph search reduces performance, resulting in a 10% decrease in accuracy. This decrease shows that utilizing Cypher, a graph query language, to query the graph effectively leverages the relationships between entities and attributes, along with the constraints that must be satisfied in the question. As a result, it produces more relevant entities in the search results.

Without Semantic Search: Semantic search is excluded from the search strategy. As shown in Table 2, removing semantic search decreases TUNES's performance by 2.3%.

Without Full-Text Search: Full-text search is excluded from the search strategy. Although this removal negatively impacts TUNES's performance, the decrease of 1.8% is not as large as with the removal of other components. This shows that the integration of graph queries and semantic search in TUNES reduces reliance on keyword matching.

5.3 Error Analysis

Table 3 reports the types of errors across each component from $\text{TUNES}_{\text{Llama-3.1-70B-Instruct}}$ on WikiTableQuestions.

Entity Identification Errors: The LLM excels in entity identification within tables, maintaining an error rate of just 4%. Most errors occur in tables with complex structures, especially those with duplicated row and column names.

Entity-Oriented Search Errors: Here, 58% of the errors are related to the quality of the entity-oriented search. Among these, only 8% are due to an insufficient number of retrieved entities, which can result from Cypher query syntax errors or questions requiring a large number of entities. The main issue lies in our Cypher execution process, which generates inaccurate new entities and attributes due to mistakes in intermediate calculations, such as incorrectly selecting entities or calculating functions.

LLM-based Answer Generation Errors: The biggest challenge for the LLM is performing calculations, such as addition and subtraction, which have an error rate of 24%. The second major challenge is logical errors, with an error rate of 20%. These errors occur because the LLM does not fully understand the question or the table's content, leading to incorrect information extraction. Other challenges include errors in comparing complex quantities, such as determining which athlete finishes a race the fastest.

Overall, TUNES still faces challenges related to inaccurate information generated from Cypher queries and the limitations of the LLM in calculations, comparisons, and reasoning.

6 Conclusion

We propose a novel approach **TUNES** to tackle the challenges of table understanding, with three main goals: (1) effectively leveraging the semantic similarities between questions and table data, along with the implicit relationships between table cells, to reduce the need for data preprocessing and keyword matching; (2) ensuring that table cells are semantically tightly connected to enhance contextual clarity; and (3) pioneering the use of a graph query language (Cypher) to improve table understanding. Experimental results show that TUNES achieves a state-of-the-art performance. In the future, we plan to extend TUNES to address other complex downstream tasks related to table understanding. Our TUNES implementation is publicly available at: `https://github.com/nhungnt7/TUNES`.

Acknowledgement

This work was supported by Monash eResearch capabilities, including M3.

This work was completed while Hoang Ngo and Dat Quoc Nguyen were at Movian AI, Vietnam.

References

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335, 2024a.

Si-An Chen, Lesly Miculicich, Julian Martin Eisenschlos, Zifeng Wang, Zilong Wang, Yanfei Chen, Yasuhisa Fujii, Hsuan-Tien Lin, Chen-Yu Lee, and Tomas Pfister. TableRAG: Million-Token Table Understanding with Language Models. In *Proceedings of The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b.

Wenhu Chen. Large Language Models are few(1)-shot Table Reasoners. In *Findings of the Association for Computational Linguistics: EACL 2023*, pp. 1120–1130, 2023.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. TabFact: A Large-scale Dataset for Table-based Fact Verification. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. *Transactions on Machine Learning Research*, 2023.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. Binding Language Models in Symbolic Languages. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. TURL: Table Understanding through Representation Learning. *ACM SIGMOD Record*, 51(1):33–40, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pp. 4171–4186, 2019.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 Herd of Models. *arXiv preprint*, arXiv:2407.21783, 2024.

Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An Evolving Query Language for Property Graphs. In *Proceedings of the 2018 International Conference on Management of Data*, pp. 1433–1445, 2018.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided Language Models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 10764–10799, 2023.

Carlos Gemmell and Jeff Dalton. ToolWriter: Question Specific Tool Synthesis for Tabular Data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 16137–16148, 2023.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. TaPas: Weakly Supervised Table Parsing via Pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4320–4333, 2020.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. TABBIE: Pretrained Representations of Tabular Data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3446–3456, 2021.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 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*, pp. 9237–9251, 2023.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed Prompting: A Modular Approach for Solving Complex Tasks. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Kezhi Kong, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Chuan Lei, Christos Faloutsos, Huzefa Rangwala, and George Karypis. OpenTab: Advancing Large Language Models as Open-domain Table Reasoners. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Weizhe Lin, Rexhina Blloshmi, Bill Byrne, Adria de Gispert, and Gonzalo Iglesias. An Inner Table Retriever for Robust Table Question Answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9909–9926, 2023.

Guang Liu, Jie Yang, and Ledell Wu. PTab: Using the Pre-trained Language Model for Modeling Tabular Data. *arXiv preprint*, arXiv:2209.08060, 2022.

Tianyang Liu, Fei Wang, and Muhao Chen. Rethinking Tabular Data Understanding with Large Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 450–482, 2024.

Ggaliwango Marvin, Nakayiza Hellen, Daudi Jjingo, and Joyce Nakatumba-Nabende. Prompt Engineering in Large Language Models. In *Proceedings of the International Conference on Data Intelligence and Cognitive Informatics*, pp. 387–402, 2023.

Md Nahid and Davood Rafiei. TabSQLify: Enhancing Reasoning Capabilities of LLMs Through Table Decomposition. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5725–5737, 2024.

Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. Can Foundation Models Wrangle Your Data? *Proceedings of the VLDB Endowment*, 16(4):738–746, 2022.

Thi-Nhung Nguyen, Hoang Ngo, Kiem-Hieu Nguyen, and Tuan-Dung Cao. A Self-enhancement Multitask Framework for Unsupervised Aspect Category Detection. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8043–8054, 2023.

Thi-Nhung Nguyen, Hoang Ngo, Dinh Phung, Thuy-Trang Vu, and Dat Quoc Nguyen. Planning for Success: Exploring LLM Long-term Planning Capabilities in Table Understanding. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning*, 2025.

Ansong Ni, Srini Iyer, Dragomir Radev, Ves Stoyanov, Wen-tau Yih, Sida I Wang, and Xi Victoria Lin. LEVER: Learning to verify language-to-code generation with execution. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 26106–26128, 2023.

Panupong Pasupat and Percy Liang. 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)*, pp. 1470–1480, 2015.

Sohan Patnaik, Heril Changwal, Milan Aggarwal, Sumit Bhatia, Yaman Kumar, and Balaji Krishnamurthy. CABINET: Content Relevance-based Noise Reduction for Table Question Answering. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. Evaluating the Text-to-SQL Capabilities of Large Language Models . *arXiv preprint*, arXiv:2204.00498, 2022.

Stephen Robertson and Hugo Zaragoza. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, 2009.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 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*, pp. 645–654, 2024.

Dingzirui Wang, Longxu Dou, and Wanxiang Che. A Survey on Table-and-Text HybridQA: Concepts, Methods, Challenges and Future Directions. *arXiv preprint*, arXiv:2212.13465, 2023a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *Proceedings of the 11th International Conference on Learning Representations*, 2023b.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. TUTA: Tree-based Transformers for Generally Structured Table Pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1780–1790, 2021.

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. Chain-of-Table: Evolving Tables in the Reasoning Chain for Table Understanding. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022.

Tingyu Xie, Qi Li, Jian Zhang, Yan Zhang, Zuozhu Liu, and Hongwei Wang. Empirical Study of Zero-Shot NER with ChatGPT. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7935–7956, 2023.

Bohao Yang, Chen Tang, Kun Zhao, Chenghao Xiao, and Chenghua Lin. Effective Distillation of Table-based Reasoning Ability from LLMs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, pp. 5538–5550, 2024.

Xianjun Yang, Yujie Lu, and Linda Petzold. Few-Shot Document-Level Event Argument Extraction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8029–8046, 2023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 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*, pp. 174–184, 2023.

Bowen Zhao, Changkai Ji, Yuejie Zhang, Wen He, Yingwen Wang, Qing Wang, Rui Feng, and Xiaobo Zhang. Large Language Models are Complex Table Parsers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14786–14802, 2023.

Yilun Zhao, Yitao Long, Hongjun Liu, Ryo Kamoi, Linyong Nan, Lyuhao Chen, Yixin Liu, Xiangru Tang, Rui Zhang, and Arman Cohan. DocMath-eval: Evaluating math reasoning capabilities of LLMs in understanding long and specialized documents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16103–16120, 2024.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Jin Ziqi and Wei Lu. Tab-CoT: Zero-shot Tabular Chain of Thought. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 10259–10277, 2023.

Prompt
Task: Given a table, your task is to identify the primary key and its position, and then identify the position of the attribute names within the table. If the table does not contain a primary key, generate one and return its position. The table positions should be referenced as a 2D Python array, with indexing starting at [0, 0]. Negative indices, such as -1 or -2, may be used for the inserted primary key. Output Template: Primary Key: [<a list of primary key>] Primary Key Position: {'column': [<column numbers>]} or {'row': [<row numbers>]} Attribute Names Position: column or row Examples: {example} Complete: {table} Output:

Table 4: Prompt to identify primary key.

Prompt
Your task is to find the relationship between primary_key and attributes, along with a description. Output Template: - <attribute name>: <relationship with primary key> | description Input and Output example: {examples} Complete: attributes: {attributes} primary_key: {primary_key} relationships:

Table 5: Prompt to generate relations.

A Prompts

Tables 4, 5, 6 and 7 show prompts used in our TUNES framework.

B Cypher code example

Cypher code to retrieve data for query "How many more times does Loose Women air than This Morning?" is shown in Figure 5.

Prompt
Schema: {schema} Your task is to extract a subgraph containing all necessary nodes to answer to question. Please return Cypher code only. Note that: 1. Value of all properties is string (convert to number if needed). 2. If the question is to the order of the value in the table, please answer based on the properties column_address (int) and row_address (int) of the table. 3. If the question requires compare strings please use toLower to compare both. Input and Output example: {examples} Complete: Question: {question} Cypher code:

Table 6: Prompt to generate Cypher query.

Prompt
Context: {context} Question: {question} Your task is to answer the question based on given context. Please provide the answer extracted only, do not include any rewrite or new content. If the question is related to the location of the data in the table, please answer based on the column address or row address. Note that -1 mean all the column/row. You can explain the answer in a short context in the next row and show the confidence score.

Table 7: Prompt to generate the final answer.

```
MATCH (:Entity {title: 'Loose Women'})-[:air_in]->(y:Year)
WITH collect(y.value) AS yearsLooseWomen
MATCH (:Entity {title: 'This Morning'})-[:air_in]->(y:Year)
WITH yearsLooseWomen, collect(y.value) AS yearsThisMorning
WITH size(yearsLooseWomen) as countLooseWomen,
     size(yearsThisMorning) as countThisMorning,
     size(yearsLooseWomen) - size(yearsThisMorning) as difference

CREATE (lw: Attribute {countLooseWomen: countLoosewomen})
CREATE (tm: Attribute {countThisMorning: countThisMorning})
CREATE (diff: Attribute {difference: difference})
CREATE (result: Entity {query: "How many times does Loose Women air in
    more than This Morning?"})
CREATE (result)-[:related_to]->(lw)
CREATE (result)-[:related_to]->(tm)
CREATE (result)-[:has_answer]->(diff)

RETURN result, lw, tm, diff
```

Figure 5: Cypher code example to retrieve data for query "How many more times does Loose Women air than This Morning?"

C Dataset statistics

Table 8 presents the statistics of the WikiTableQuestions and TabFact test sets.

Statistics	WikiTQ	TabFact
Number of Questions	4343	2024
Number of Tables	421	298
Min/Max Number of Rows	6/518	5/49
Min/Max Number of Columns	5/20	3/21

Table 8: Statistics of the WikiTableQuestions (WikiTQ) and TabFact test sets.

D Example of errors

D.1 Entity identification error

D.1.1 Table structure identification error

Figure 6 illustrates a table structure identification error.

Result	Encrypted	Result	Encrypted	Result	Encrypted	Result	Encrypted	Result	Encrypted
0	57	19	108	38	113	57	91	76	79
1	109	20	125	39	116	58	37	77	65
2	60	21	82	40	121	59	92	78	49
3	46	22	69	41	102	60	51	79	67
...	...	...	...	...	...	...	...	...	...
16	126	35	93	54	90	73	50	92	115
17	120	36	38	55	110	74	70	93	98
18	68	37	103	56	44	75	104		

Question: What is the difference between the encryption of result 1 and result 38?

Extracted entities:

Primary Key	Attributes Names

Result	Encrypted	Result	Encrypted	Result	Encrypted	Result	Encrypted	Result	Encrypted
1	109	20	125	39	116	58	37	77	65

Predicted Answer: 0 X
Golden Answer: 4 V

Description: TUNES identifies that 'Result' in column 0 is the primary key, while other column names represent attributes. Consequently, the search engine can only retrieve entity 'Result 1' based on its primary key, leading to insufficient information to answer the question.

Note that: representing the entity in tabular form is for illustration purposes solely.

Figure 6: Illustration of Table structure identification error.

D.2 Search errors

D.2.1 Insufficient entity retrieval error

Figure 7 and Figure 8 illustrate insufficient entity retrieval errors.

D.2.2 Incorrect self-generated entity error

Figure 9 illustrates an incorrect self-generated entity error.

D.3 Answer generation errors

D.3.1 Comparative error

Figure 10 illustrates a comparative error.

D.3.2 Numerical error

Figure 11 illustrates a numerical error.

D.3.3 Logical error

Figure 12 illustrates a logical error.

Chart year	Artist	Album	Song	Billboard Hot 100
2014	Puff Daddy	Fothcoming Album	Big Homie	New Single
2014	Rick Ross f/Jay Z	Fothcoming Album	Devil Is A Lie	New Single
2013	Chris Brown	X	Fine China	31
2013	Ludacris	Forthcoming Album	Raised in the South	New Single
...	...	...	...	...
2012	Chris Brown	Fortune	Don't Judge Me	18
2009	Justin Bieber	My World	Love Me	37
...	...	...	...	...
2009	Trey Songz	Ready	I Invented Sex	42
2010	Ciara	Basic Instinct	Ride ft. Ludacris	45

Question: How many releases were not on a new single?

Extracted entities:

Chart year	Artist	Album	Song	Billboard Hot 100
2013	Chris Brown	X	Fine China	31
...	...	...	...	...
2009	Trey Songz	Ready	I Invented Sex	42
2010	Ciara	Basic Instinct	Ride ft. Ludacris	45

Predicted Answer: 50 X
Golden Answer: 52 V

Description: The search output is missing 'Don't Judge Me' and 'Love Me', resulting in insufficient data to answer the question accurately.

Note that: representing the entity in tabular form is for illustration purposes solely.

Figure 7: Illustration of Insufficient entity retrieval error.

Tie No.	Home team	Score	Away team	Date	Attendance
...	...	...	...	...	...
10	Fulham	3 – 0	Charlton Athletic	26 January 2003	12,203
11	Brentford	0 – 3	Burnley	25 January 2003	9,563
12	Manchester United	6 – 0	West Ham United	26 January 2003	67,181
13	Norwich City	1 – 0	Dagenham & Redbridge	25 January 2003	21,164
14	Crystal Palace	0 – 0	Liverpool	26 January 2003	26,054
15	Farnborough Town	1 – 5	Arsenal	25 January 2003	35,108
16	Stoke City	3 – 0	Bournemouth	26 January 2003	12,004

Question: Fulham and Stoke City both won with how many points?

Cypher:

```
MATCH (n)
WHERE toLower(n.home_team) CONTAINS toLower('Fulham') OR toLower(n.home_team) CONTAINS toLower('Stoke
City')
WITH n
WHERE n.score CONTAINS '–'
WITH split(n.score, '–') AS score
WHERE toInteger(score[0]) > toInteger(score[1])
```

Error:

```
Query cannot conclude with MATCH (must be a RETURN clause, a FINISH clause, an update clause, a unit
subquery call, or a procedure call with no YIELD)
```

Description: Cypher query does not include a RETURN clause, so it fails to produce any result.

Figure 8: Illustration of Insufficient entity retrieval error.

Year	Class	Car	Team
1972	S 3.0	Ligier JS2 Maserati 3.0L V6	Automobiles Ligier
1973	S 3.0	Ligier JS2 Maserati 3.0L V6	Automobiles Ligier
1974	S 3.0	Ligier JS2 Maserati 3.0L V6	Automobiles Ligier
1977	S +2.0	Renault Alpine A442 Renault 2.0L Turbo V6	Renault Sport
1978	S +2.0	Mirage M9 Renault 2.0L Turbo V6	Grand Touring Cars Inc.
1990	C1	Porsche 962C Porsche Type-935 3.0L Turbo Flat-6	Joest Porsche Racing
1993	GT	Venturi 500LM Renault PRV 3.0 L Turbo V6	Jacadi Racing
1994	GT2	Porsche 911 Carrera RSR Porsche 3.8 L Flat-6	Larbre Compétition
1996	GT1	McLaren F1 GTR BMW S70 6.1L V12	Team Bigazzi SRL

Question: How many consecutive years did team automobiles ligier run the maserati in the 70's?

Cypher query:

```
MATCH (n)
WHERE toLower(n.team) CONTAINS toLower('automobiles ligier')
AND toLower(n.car) CONTAINS toLower('maserati')
AND n.name >= '1970' AND n.name < '1980'
WITH n
ORDER BY n.row_address
WITH collect(n) AS years
UNWIND range(1, size(years) - 1) AS i
WITH years[i] AS current, years[i - 1] AS previous
WHERE current.row_address = previous.row_address + 1
RETURN current
```

Extracted entities:

Year	Class	Car	Team
1973	S 3.0	Ligier JS2 Maserati 3.0L V6	Automobiles Ligier
1974	S 3.0	Ligier JS2 Maserati 3.0L V6	Automobiles Ligier

Predicted Answer: 2 X
Golden Answer: 3 V

Description: The query only checks adjacent years, allowing it to identify pairs of consecutive years, but it does not return the first year in this sequence.

Note that: representing the entity in tabular form is for illustration purposes solely.

Figure 9: Illustration of Incorrect self-generated entity error.

Year	Name	Label
1982	He's Gonna Take You Home	Becket
1982	Thanks to You	Becket
1983	I Need You Now	Jive
1984	Thin Line	Power House
1986	Say It Again	Spring
1987	Send It C.O.D.	New Image

Question: Which singles are newer? "thin line" or "say it again"?

Extracted entities:

Year	Name	Label
1984	Thin Line	Power House
1986	Say It Again	Spring

Predicted Answer: Thin Line
Thin Line was released in 1984, while Say it Again was released in 1986. Therefore, Thin Line is newer than Say it Again because 1984 < 1986. X

Golden Answer: Say It Again V

Description: The LLM incorrectly claims that "Thin Line" is newer than "Say it Again,". Since 1984 < 1986, "Thin Line" is actually older, not newer.

Note that: representing the entity in tabular form is for illustration purposes solely.

Figure 10: Illustration of Comparative error.

Date	Opponents	Venue	Result	Scorers	Attendance
28 Aug 1920	Reading	H	0–1		14,500
1 Sep 1920	Bristol Rovers	A	2–3	Walker, Wolstenholme	10,000
4 Sep 1920	Reading	A	0–4		10,000
9 Sep 1920	Bristol Rovers	H	0–2		8,000
11 Sep 1920	Plymouth Argyle	A	1–5	Wolstenholme	12,000
18 Sep 1920	Plymouth Argyle	H	0–0		8,000
...	...	...	...	...	...

Question: what was the total attendance of the first five games in the season?

Extracted entities:

Date	Opponents	Venue	Result	Scorers	Attendance
28 Aug 1920	Reading	H	0–1		14,500
1 Sep 1920	Bristol Rovers	A	2–3	Walker, Wolstenholme	10,000
4 Sep 1920	Reading	A	0–4		10,000
9 Sep 1920	Bristol Rovers	H	0–2		8,000
11 Sep 1920	Plymouth Argyle	A	1–5	Wolstenholme	12,000

Predicted Answer: 60,500
The total attendance of the first five games (14,500 + 10,000 + 10,000 + 8,000 + 12,000) is calculated by adding the attendance figures for each game. X

Golden Answer: 54500 V

Description: The LLM incorrectly predicts the total attendance as 60,500, despite the correct calculation showing that the sum of the first five games' attendances (14,500 + 10,000 + 10,000 + 8,000 + 12,000) equals 54,500, not 60,500.

Note that: representing the entity in tabular form is for illustration purposes solely.

Figure 11: Illustration of Numerical error.

Chord	Root	Minor third	Perfect fifth	Major seventh
CmM7	C	E ♭	G	B
C♯mM7	C♯	E	G♯	B♯ (C)
D ♭ mM7	D ♭	F ♭ (E)	A ♭	C
...	...	...	...	...
EmM7	E	G	B	D♯
...	...	...	...	...
AmM7	A	C	E	G♯
...	...	...	...	...

Question: the chords e minor major seventh and a minor major seventh have which note in common?

Extracted Entities:

Chord	Root	Minor third	Perfect fifth	Major seventh
EmM7	E	G	B	D♯
AmM7	A	C	E	G♯

Predicted Answer: G X

Golden Answer: E V

Description: The LLM incorrectly predicts the root of the chord as G, even though the correct root for the EmM7 chord is provided as E.

Note that: representing the entity in tabular form is for illustration purposes solely.

Figure 12: Illustration of Logical error.