Hybrid Graphs for Table-and-Text based Question Answering using LLMs

Ankush Agarwal [*1] **Ganesh S**[†1,2] **Chaitanya Devaguptapu**[1]

[1]Fujitsu Research India
[2]IIT Madras
ankush.agarwal@fujitsu.com, ganeshsenrayan@outlook.com, email@chaitanya.one

Abstract

Answering questions that require reasoning and aggregation across both structured (tables) and unstructured (raw text) data sources presents significant challenges. Current methods rely on fine-tuning and high-quality, human-curated data, which is difficult to obtain. Recent advances in Large Language Models (LLMs) have shown promising results for multi-hop question answering (QA) over single-source text data in a zero-shot setting, yet exploration into multi-source Table-Text QA remains limited. In this paper, we present a novel Hybrid Graph-based approach for Table-Text QA that leverages LLMs without fine-tuning. Our method constructs a unified Hybrid Graph from textual and tabular data, pruning information based on the input question to provide the LLM with relevant context concisely. We evaluate our approach on the challenging Hybrid-QA and OTT-QA datasets using state-of-the-art LLMs, including GPT-3.5, GPT-4, and LLaMA-3. Our method achieves the best zero-shot performance on both datasets, improving Exact Match scores by up to 10% on Hybrid-QA and 5.4% on OTT-QA. Moreover, our approach reduces token usage by up to 53% compared to the original context.

1 Introduction

In today's data-rich world, information is scattered across various sources, which can be broadly divided into two types: structured (tables, databases) and unstructured (raw text from various files). The ability to effectively answer questions that require reasoning and aggregation across diverse data sources has become increasingly crucial. Such questions are often referred to as *hybrid questions* and the task is often referred to as Table-Text Question Answering (QA). For instance, consider the question: *"What place was achieved by the person who finished the Berlin marathon in 2:13.32 in 2011, the first time he competed in a marathon?"*, answering this question requires extracting the name of the person who finished the Berlin marathon in 2011 from raw unstructured text and linking it with a structured data source like a database to determine their place. The ability to answer hybrid questions has immense real-world value, as it enables the combination of relevant information from multiple sources, leading to more comprehensive and capable QA systems.

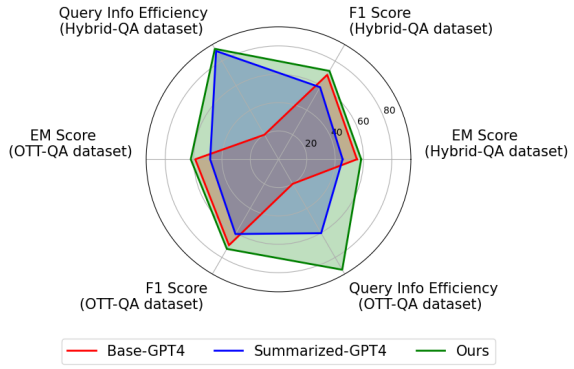

Figure 1: **Multi-Dimensional Improvements**: Our method (with GPT-4 as reader LLM) demonstrates superior results on Hybrid-QA and OTT-QA. **Metrics used:** EM: Exact-Match with the gold answer, F1-Score, Query Info Efficiency: normalized metric ($\frac{1}{\text{Input Token Size}}$) that quantifies the efficiency of using fewer input tokens to represent the same documents, w.r.t. reader LLM.

Existing QA methods primarily focus on single data sources, either structured or unstructured. These approaches are limited when answering questions that require reasoning and aggregation across ***both*** structured and unstructured data. Datasets like Hybrid-QA (Chen et al., 2020b) and OTT-QA (Chen et al., 2020a) necessitate combining information from tables and text, however current methods applied to these datasets (Lee et al., 2023; Eisenschlos et al., 2021; Kumar et al., 2023; Lei et al., 2023) rely on the availability of training data

*Corresponding author : ankush.agarwal@fujitsu.com
†Work done during internship at Fujitsu Research India.

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 858–875

with pre-established connections between the structured and unstructured data. In real-world applications, such training data may not always be available and is often expensive to collect. For example, even for general data sources like Wikipedia, there are only two publicly available datasets (Hybrid-QA and OTT-QA) that contain hybrid questions.

Given sufficient context, large language models (LLMs) (Team et al., 2023; Achiam et al., 2023) can effectively extract answers from the provided context for a wide range of question categories and domains. Recently, LLM-based zero-shot QA has emerged as a robust alternative to traditional fine-tuning-based QA methods. For the task of Table-Text QA, a straightforward approach is to provide the entire text and table as input to the LLM alongside the question. However, this method is not a scalable approach due to token cost and the limited context length of LLMs. Approaches such as context truncation (Xie et al., 2022) and summarization (Jin et al., 2024) using off-the-shelf techniques like those in LangChain[1] have been proposed. Nevertheless, indiscriminate truncation or summarization can lead to significant performance degradation, as critical information may be omitted, resulting in less accurate or incomplete answers (as shown in Section-5, Table-3).

To enhance the ability of LLMs to answer hybrid questions and efficiently retrieve and leverage relevant context, we propose ODYSSEY[2]. Our approach operates in a zero-shot setting, aiming to filter out noise and provide the LLM with the most relevant context in a concise manner. At a high level, our method consists of two main steps: i) Constructing a ***single*** Hybrid Graph from both textual and tabular data, and ii) Pruning information from the graph based on the question. To demonstrate the effectiveness of our method compared to existing approaches, we present a case study on the Hybrid-QA dataset (Chen et al., 2020b) in Figure 2.

We evaluate ODYSSEY on Hybrid-QA Chen et al. (2020b) and OTT-QA Chen et al. (2020a) datasets using three state-of-the-art LLMs for QA: GPT-3.5, GPT-4 and LLaMA-3. We compare our method against relevant baselines across five metrics: Exact Match (EM) (Zhang et al., 2023), F1-Score (Lei et al., 2023), Precision, Recall, and BERTScore-F1 (Zhang et al.), demonstrating that constructing and leveraging a Hybrid-Graph by combining structured and unstructured information sources results in significant performance gains. Our key contributions can be summarized as follows:

- A novel approach that jointly distills information from structured and unstructured data sources to construct a Hybrid Graph.
- An increase in performance over the current SoTA fine-tuning-free approach, improving EM and F1 scores by 7.3% and 20.9% for Hybrid-QA using GPT-4 (see Table 4).
- A significant reduction in the input token size. Our Hybrid Graph based approach uses up to 45% and 53% fewer tokens than the original table and text for the Hybrid-QA and OTT-QA datasets respectively (see Table 6).

2 Related Work

Question Answering: With the advent of language models, the task of question answering over structured and unstructured data sources has gained a significant traction and interest. For structured data sources like tables, recent methods have focused on generating SQL queries from natural language input (Wang et al., 2020; Hui et al., 2022; Li et al., 2023a,b; Gao et al., 2023), converting structured data to graphical forms (Jiang et al., 2023; Perozzi et al., 2024; Tan et al., 2024), or allowing language models to directly interact with the tables (Herzig et al., 2020; Wang et al., 2021). Similarly, efforts have been made to develop efficient QA systems for unstructured text (Seo et al., 2016; Hu et al., 2018; Perez et al., 2020; Seonwoo et al., 2020). While many approaches have been proposed for answering questions on either structured or unstructured data sources while treating them in isolation, there is a scarcity of methods that focus on Table-Text QA systems, *i.e.*, questions that require reasoning and aggregation of information from both structured and unstructured data sources simultaneously.

Table-Text Question Answering: The task of Table-Text QA based on text-and-tabular data has recently started gaining traction because of the creation and availability of datasets like Hybrid-QA (Chen et al., 2020b) and OTT-QA (Chen et al., 2020a), both of which are based on Wikipedia as the primary data source containing both text and tabular data.

To tackle challenges associated with Table-Text

[1] https://www.langchain.com

[2] Named after the primary goal of our approach to navigate from complex multi-hop Table-Text data to hybrid graphs that provide greater clarity

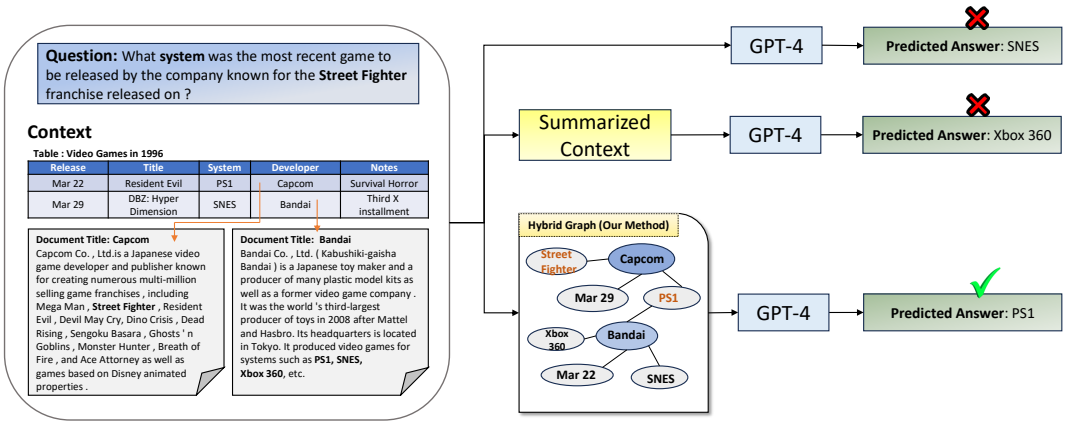

Figure 2: **Case study on Hybrid-QA:** Comparison of our method (ODYSSEY) against various baselines on an example from the Hybrid-QA dataset. Baselines: (i) **Question + Context:** Providing the LLM only the question without any additional context (ii) **Question + Summarized Context:** Passing the question along with the summarized documents and table. Our method delivers accurate answer because the Hybrid Graph efficiently connects "Street Fighter" from the document "Capcom" with the relevant table, guiding GPT-4 in generating the correct response, i.e., "PS1" from the "System" table column.

QA, several approaches have been proposed (Chen et al., 2020b; Wang et al., 2022; Lei et al., 2023). While these methods perform well, they rely on supervised fine-tuning data to learn and aggregate information across text and tables for question answering. Collecting supervised data for the task of Table-Text QA is not only resource intensive (Orr and Crawford, 2023) but may not always be possible.

Fine-Tuning-free Table-Text QA: The emergence of large-scaled pre-trained LLMs (Achiam et al., 2023; Le Scao et al., 2023) has revolutionized the field of NLP and especially has opened up ways to address the task of QA in a fine-tuning-free manner. Based on a given question, the relevant context is retrieved from the data corpus using state-of-the-art retrievers (Robertson et al., 2009; Karpukhin et al., 2020; Khattab and Zaharia, 2020) and this context is passed along with the question to the LLM. Recent work (Shi et al., 2024) tackles this issue by generating and executing code in a few-shot setting, similar to natural language-to-SQL conversion, but it does not account for noise in the context.

ODYSSEY is a fine-tuning-free zero-shot approach for Table-Text QA. Our method efficiently prunes relevant information from both tabular and textual sources and performs the Table-Text QA task. Furthermore, ODYSSEY effectively addresses the unique challenges associated with Table-Text QA, such as integrating information from heterogeneous sources and performing multi-hop reasoning across tables and text.

3 Methodology

In this section, we explain the key parts of our approach for solving Hybrid-QA in a fine-tuning-free manner. We begin by describing the problem setting, followed by a detailed explanation of our method, which includes question analysis, graph construction and graph traversal. Finally, we discuss the retrieval and the information that is passed into the LLM.

3.1 Problem Definition

Assume a structured data source $\mathcal{T}$ with rows denoted using $\mathcal{R}$, where $r_i \in \mathcal{R}$ denotes the i^{th} row, and headers denoted using $\mathcal{H}$, where $h_j \in \mathcal{H}$ denotes the j^{th} header. Let $\mathcal{T}(r_i, h_j)$ represent the value in the i^{th} row and the j^{th} header / column. For a few cells in every row, certain cells in column h_j contain unstructured data sources linked to them. Specifically, $\mathcal{T}(r_i, h_j)$ is linked to k documents $\{d_{ij}^1, d_{ij}^2, \ldots, d_{ij}^k\}$ which contain unstructured information.

Given a natural language question $\mathcal{Q}$ related to both the structured data $\mathcal{T}$ and the unstructured data $\mathcal{D}$ (comprising the linked documents), the objective is to output an answer $\mathcal{A}$.

To aid in the explanation of our methodology, we will use the following example throughout this section. Consider the following question $\mathcal{Q}$ from one of the datasets we consider in our experiments: *"The driver who finished in position 4 in the 2004 United States Grand Prix was of what nationality?"*. This question requires reasoning and aggregation

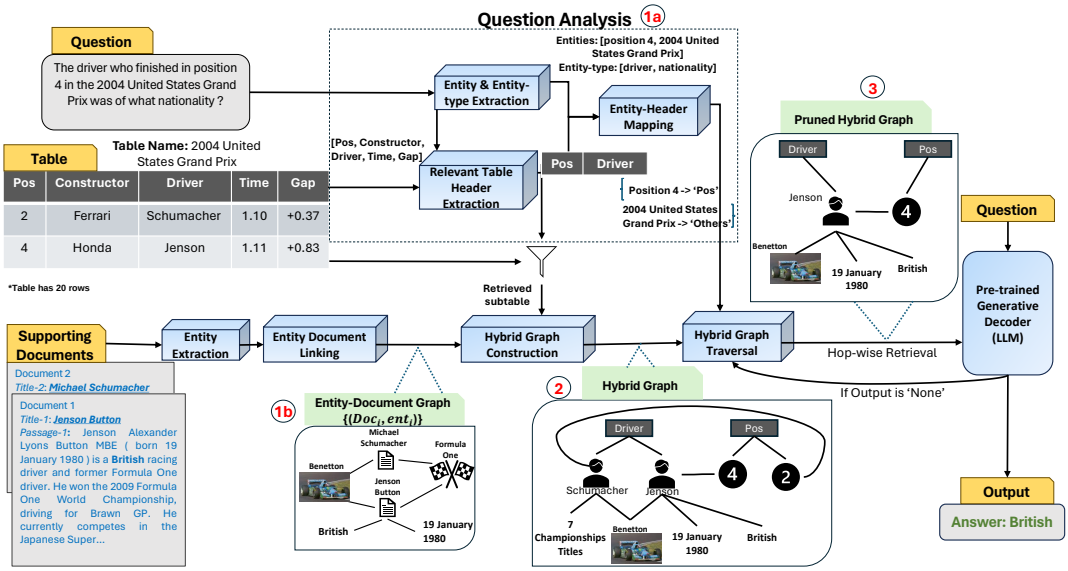

Figure 3: **Overview of the ODYSSEY framework.** Our method comprises of 3 steps: i) Question Analysis, ii) Hybrid Graph Construction, and iii) Hybrid Graph Traversal. First, we begin with Question Analysis (1a in the figure) from where we get question entities, retrieved sub-table, and entity-header mapping. Next, we construct the Entity-Document Graph (1b in the figure). Using entity-doc graph and retrieved sub-table, we construct the Hybrid Graph (2 in the figure). At last, we perform Hybrid Graph Traversal (3 in the figure) to get the pruned graph which serves as input for the LLM. For a detailed walkthrough, refer to Appendix A.2

across a structured data source and an unstructured data source. The structured data source, table $\mathcal{T}$, contains the following headers $\mathcal{H} = \{$`'Pos'`, `'Constructor'`, `'Driver'`, `'Time'`, `'Gap'`$\}$ with 20 rows ($|\mathcal{R}| = 20$), and each row $r_i \in \mathcal{R}$ contains 5 cells $\mathcal{T}(r_i, h_j)$ for $j = 1, \ldots, 5$. Documents linked to a cell $\mathcal{T}(r_i, h_j)$ in the table are denoted as $\mathcal{D}(r_i, h_j) = \{d_{ij}^1, d_{ij}^2, \ldots, d_{ij}^k\}$. Our objective is to generate the answer $\mathcal{A}$ using the language model $\mathcal{L}$, which spans table cells and linked documents.

3.2 Proposed Method (ODYSSEY)

Our proposed method consists of the following key steps:

1. **Question Analysis:** We analyze the question to identify key entities and entity types that will later aid in graph construction, traversal, and answering the question (Section 3.2.1).
2. **Hybrid Graph Construction:** We locate the tabular evidence through structure modeling and model the relationships among heterogeneous evidence (Section 3.2.2).
3. **Hybrid Graph Traversal:** We prune the Hybrid Graph based on question-derived entities and perform multi-hop reasoning and traversal (Section 3.2.3).
4. **Reader LLM:** We use a Language Model to generate the final answer using the pruned graph and linked documents (Section 3.2.4).

3.2.1 Question Analysis

To effectively address questions based on a given context, our approach utilizes an LLM to identify entities and entity types within the question. For example, in figure 3 - 1a, we identify entities such as `'position 4'` and `'2004 United States Grand Prix'`, along with entity types `'driver'` and `'nationality'`. These entities and entity types guide us in determining relevant table headers, such as `'Pos'` and `'Driver'`, where `'Pos'` corresponds to `'position 4'` and `'Driver'` corresponds to the entity type. We establish an entity-header mapping by aligning entities with the retrieved relevant headers. Entities that match specific headers are mapped accordingly, while those that do not fit any header are categorized as `'Others'` (see figure 3 - 1a). The effectiveness of this mapping technique will be demonstrated in Section 3.2.3. The Question Analysis prompt used in our method is detailed in Appendix A.3.1.

3.2.2 Hybrid Graph Construction

Our Hybrid Graph consists of two connected components: the table and documents. By selecting relevant table headers, we retrieve the sub-table. Simultaneously, from the documents, we construct

the Entity-Document graph. These components are then integrated to form the Hybrid Graph.

Sub-table Retrieval Column headers are selected to retrieve the sub-table. As explained in Section 3.2.1, the `'Pos'` and `'Driver'` table header columns are selected for the Hybrid Graph. Each row cell in the table is connected with others, and we replicate these connections to visualize the sub-table in a graph format. Refer to Figure 3 - (2) to see the connections between the row cells.

Entity-Document Graph Construction The entity-document graph includes document and entity nodes, connected by single edges. Named entities are extracted from documents, and edges are formed between document and entity nodes, creating a bipartite graph that shows their relationships (see figure 3 - (1b)).

As discussed in Section 3.1, certain table cells are linked to specific documents. We connect the entities from each document to their corresponding retrieved sub-table cells to form the Hybrid Graph (see figure 3 - (2)).

3.2.3 Hybrid Graph Traversal

After constructing the Hybrid Graph, we prune it based on the question to filter out noise. Using the entity-header mapping dictionary (described in Section 3.2.1), we perform a Breadth-First Search (BFS) to semantically match the question entities with table column cells and entities in the entity-document graph. For semantic matching, we first gather all document entities into a set called `entity_total`. In the entity-header mapping, entities mapped to a header are aligned with the corresponding column, while those mapped to `'Others'` are aligned with `entity_total` (as explained in Section 3.2.1). For example, as shown in Figure 3, the entity `'Positon 4'` is mapped to the header `'Pos'`, and after semantic matching, it aligns with the entity '4'. In contrast, the entity `'2004 United States Grand Prix'` does not match any entity in `entity_total`. If the header-mapped entities are not successfully matched, we expand the matching process to include other table columns to account for potential noise in the entity-header mapping.

Once the semantically matched entities are obtained, we initiate a BFS traversal using them as starting points. In figure 3, the starting point is entity `'4'`. We perform a 3-hop traversal, which results in a pruned graph (as shown in Figure 3 - (3)). During traversal, we record the paths in the graph and store them in a hop-wise dictionary, categorized by 1-hop, 2-hop, and 3-hop. If no entities are matched during semantic matching, we store the entire table and linked passages. The number of such cases is mentioned in Section 5.1.

3.2.4 Reader LLM prompt

After obtaining the pruned graph and storing it in a hop-wise dictionary, we use an LLM to answer the question. Initially, only 1st-hop items are provided as context. If no answer is found, the LLM returns `'None'`. We then include 2nd-hop items as context, along with relevant linked passages given as output from the 1st-hop LLM call, and concatenate the 1st-hop and 2nd-hop tables. This iterative process continues up to 3 hops. If the LLM still returns `'None'` after 3 hops, the entire table-text is used as context. The number of such cases is mentioned in Section 5.1. We limit the process to 3 hops due to minimal improvements observed in LLM performance beyond that during hyperparameter tuning (see Section 4). The LLM reader prompt used in our method is detailed in Appendix A.3.2.

4 Experimental Setup

This section details our experimental setup, including the datasets used to evaluate our method, the baselines for comparison, evaluation metrics, and implementation specifics.

4.1 Experimental Datasets

We evaluate ODYSSEY on two Table-Text Hybrid-QA datasets: Hybrid-QA (Chen et al., 2020b) and OTT-QA (Chen et al., 2020a).

Hybrid-QA is a large-scale, complex, multihop Table-Text Hybrid-QA benchmark comprising tables and texts from Wikipedia. Each table row describes various attributes of an instance, linked to corresponding Wikipedia passages that provide detailed descriptions. Dataset statistics are provided in Table 1.

Split	Train	Dev	Test	Total
In-Passage	35,215	2,025	20,45	39,285
In-Table	26,803	1,349	1,346	29,498
Computed	664	92	72	828
Total	62,682	3,466	3,463	69,611

Table 1: Data Split: In-Table means the answer comes from plain text in the table, and In-Passage means the answer comes from certain passage.

OTT-QA extends Hybrid-QA into a large-scale

open-domain QA dataset over tables and text, requiring both table and passage to be retrieved before answering questions. It samples around 2,000 questions from the in-domain Hybrid-QA dataset, mixing them with newly collected out-domain questions for the dev and test sets. OTT-QA consists of 41,469 questions in the training set, 2,214 in the dev set, and 2,158 in the test set. Unlike Hybrid-QA, many questions in OTT-QA have multiple plausible inference chains. Dataset statistics are provided in Table 2.

	Train	Dev	Test
Total	41,469	2,214	2,158

Table 2: OTT-QA Statistics

Since we operate in a zero-shot setting, we do not use the training set. Instead, we uniformly sample 500 examples from the dev sets of both Hybrid-QA and OTT-QA for comparison with baselines. For hyperparameter tuning, we select 50 questions from the Hybrid-QA dev set, ensuring they are distinct from our experimental set. For OTT-QA, we choose the newly added questions different from Hybrid-QA and use the retrieved table and retrieved text (Chen et al., 2020a) to test our methodology. The decision to use a sample dataset (Trivedi et al., 2023; Li and Du, 2023) for testing our methodology is driven by the costs associated with OpenAI API[3].

4.2 Baselines

We operate in a fine-tuning-free, zero-shot and closed-domain setting. Our experiments involve both closed-source and open-source large language models. For closed-source models we utilize two popular LLMs - GPT-3.5-turbo-1106[4] and GPT-4-1106-preview (Achiam et al., 2023), both at a temperature setting of 0. For open source LLMs, we employ Llama3-8B[5]. We compare our method against three baselines:

- **Base**: To test the parametric knowledge of the reader LLM, we provide only the question and elicit a response.
- **Base w/ Table & Text** (Zhang et al., 2023): To test the hybrid tabular and textual QA understanding of the LLM, we provide the full context (table and passages) following the prompt outlined in Zhang et al. (2023).
- **Base w/ Table & summarized Text**[6]: To reduce the noise in the unstructured information and to make it easier for LLM to find the answer, we employ LangChain's proposed approach to summarize (Jin et al., 2024) the text and then pass the summary of the text with the entire table to the LLM for answering the questions.
- We also show results when both the text and the table are summarized. Refer to Appendix A.1.2 for details.

All prompts used for the above mentioned baselines are detailed in Appendix A.4.

4.3 Evaluation Metrics

For evaluation, we employ several metrics: Exact Match (EM) (Zhang et al., 2023), F1-Score (Lei et al., 2023), Precision (P), and Recall (R) to assess the correctness of predicted answers. These metrics are implemented using the same codebase as Hybrid-QA (Chen et al., 2020b). Additionally, for semantic evaluation, we employ BERTScore-F1 (B) (Zhang et al.) and utilize the `bert-base-uncased` model (Devlin et al., 2018) for computing similarity matching.

4.4 Implementation Details

ODYSSEY hyperparameters We use GPT-3.5-turbo-1106 for question analysis, including entity extraction, relevant header fetching from tables, and entity-header mapping. To fetch entities from linked passages, we use the SpaCy[7] transformers model with `en_core_web_trf`[8], which utilizes a RoBERTa-base (Liu et al., 2019) model. These components collectively enable us to construct the Hybrid Graph.

For graph traversal, we match question entities with the Hybrid Graph using the `instructor-xl` model [9], running on a GPU with 6GB of VRAM. We select the highest-ranked entity surpassing a 0.8 threshold. During the breadth-first search traversal, we store up to 3-hops to pass to the LLM reader. These values were determined by experimenting on

[3] https://openai.com/api/
[4] https://platform.openai.com/docs/models/gpt-3-5-turbo
[5] https://huggingface.co/meta-llama/Meta-Llama-3-8B
[6] https://blog.langchain.dev/semi-structured-multi-modal-rag/
[7] https://pypi.org/project/spacy-transformers/
[8] https://huggingface.co/spacy/en_core_web_trf
[9] https://huggingface.co/hkunlp/instructor-xl

50 samples from the dev set. The average runtime of ODYSSEY is 3.6 seconds per instance.

Text Summarization To implement the Base with Table and Summarized Text, we use GPT-3.5-turbo-1106 for summarizing the text. Alternatively, this summarization can be done using fine-tuned models like T5 (Raffel et al., 2020).

5 Results and Analysis

In this section, we present the results of our method and compare them with the baselines discussed in Section 4.2. We show results using Llama3-8B, GPT-3.5, and GPT-4, along with fine-tuned models on the Hybrid-QA and OTT-QA datasets (see Tables 3, 4, and 5) across various evaluation metrics discussed in Section 4.3. Additionally, we analyze token efficiency, *i.e.*, the number of tokens passed as input to the LLM (see Table 6). Furthermore, we provide a detailed analysis of our method, including an ablation study (Section 5.3), error analysis (Section 5.4), and a hop-wise breakdown of the results, illustrated in Figure 4.

5.1 Evaluation on Tabular-and-Text QA

We show evaluation on Hybrid-QA and OTT-QA datasets in Table 3, 4, 5.

Table 3 shows the performance of ODYSSEY, our method with hop-wise extraction (denoted as 'w/ hopwise' in the table). It consistently outperforms all baseline methods across Exact Match (EM), F1-score, Precision, Recall and BERTScore-F1 on the Hybrid-QA and OTT-QA datasets for all three LLMs. This improvement underscores the effectiveness of our approach in efficiently extracting crucial information while effectively filtering noise from the table-text in a question specific manner. It is worth noting that our model performs best in the zero-shot setting for closed-source LLMs like GPT-3.5 and GPT-4, and also shows strong performance with the comparatively smaller open-source model Llama3-8B for the task of Table-Text QA.

As mentioned in Section 3.2.4, if the LLM outputs `'None'` after 3-hops, it becomes necessary to pass the entire table and text as input to the LLM. These occurrences, where graph traversal is not possible or the LLM requires the full context, account for approximately 10% and 8% of the experiment data for Hybrid-QA and OTT-QA, respectively, across all models. However, it is important to note that, even without utilizing the complete experimental data, we achieve an EM score of 58% on HybridQA and 62.0% on OTTQA using GPT-4.

Table 4 and 5 compare our zero-shot method with other approaches, including fine-tuning, few-shot methods (Pan et al., 2021; Shi et al., 2024), and CoT (Wei et al., 2022). Our method outperforms the fine-tuned models on the OTT-QA dataset and achieves comparable results on the Hybrid-QA dataset. Table 4 presents results on the Hybrid-QA development set, which includes a randomly selected subset of 200 samples, as employed by HPROPRO (Shi et al., 2024) with a temperature setting of 0. For the final LLM call, we utilized GPT-4-0613 (consistent with Shi et al. (2024)), and GPT-3.5-turbo for Question Analysis. Notably, our method demonstrates a 7.3% improvement in exact match (EM) and a 20.9% increase in F1 score compared to HPROPRO.

5.2 Efficient Query Context Handling

In complex QA tasks, such as Hybrid tabular and textual QA with LLMs like GPT-3.5 and GPT-4, retrieving relevant information from the data corpus is crucial. This enables the LLMs to better connect the links between the structured and unstructured data for more accurate QA. Our method achieves this task by effectively filtering out noise, which not only increases QA accuracy (as shown in Table 3) but also results in a reduced reader input context size compared to the original context (as shown in Table 6). Specifically, the Question Analysis component of our method, discussed in Section 3.2.1 requires on average an input token size of 989 for Hybrid-QA and 850 for OTT-QA respectively. This amounts to total input token sizes to 4846 and 3595, respectively, representing a 32.65% and 38.7% reduction compared to the original context for Hybrid-QA and OTT-QA datasets respectively. Additionally, the output token size is smaller across all methods; our method produces around 70 tokens, while the baselines typically produce 4 tokens.

We analyze the questions answered by our method using a hop-wise criteria on the Hybrid-QA dataset (as shown in Figure 4). Our method answers 144 questions with GPT-3.5 and 190 questions with GPT-4, achieving EM scores of 50.7% and 64.8% in 1-hop (see left-side of Figure 4) with average token sizes of 1369 and 1479 (see right-side of Figure 4), respectively. For both Hybrid-QA and OTT-QA, almost 90% questions were answered using 1-hop and 2-hop connections. This underscores the effectiveness of our Hybrid Graph-

Datasets	Hybrid-QA					OTT-QA				
Methods	EM (↑)	F1 (↑)	P (↑)	R (↑)	B (↑)	EM (↑)	F1 (↑)	P (↑)	R (↑)	B (↑)
Reader: gpt-4-1106-preview (zero-shot)										
Base	4.60	12.44	12.08	12.25	62.10	4.85	12.44	12.25	14.24	64.30
Base w/ Table & Text (Zhang et al., 2023)	55.40	68.84	68.92	71.79	85.54	58.86	72.28	72.16	74.28	87.51
Base w/ Table & Summarized Text[2]	45.29	58.72	58.78	61.39	81.14	48.31	60.90	61.47	63.12	82.02
Our Method w/o hopwise	58.20	71.54	71.75	74.35	86.30	61.00	72.64	73.60	74.27	88.06
Our Method w/ hopwise	**58.40**	**71.80**	**71.62**	**74.22**	**86.53**	**62.02**	**73.02**	**73.40**	**75.13**	**88.18**
Reader: gpt-3.5-turbo-1106 (zero-shot)										
Base	4.20	11.54	11.62	12.45	65.05	5.27	12.20	12.35	13.44	66.17
Base w/ Table & Text (Zhang et al., 2023)	40.22	53.47	54.18	55.57	81.18	42.41	54.06	54.04	**55.8**	81.63
Base w/ Table & Summarized Text[2]	41.19	51.64	52.03	53.12	81.05	37.34	49.58	49.80	51.73	79.87
Our Method w/o hopwise	41.8	52.37	52.82	53.72	81.31	42.19	53.61	54.18	55.30	81.58
Our Method w/ hopwise	**44.2**	**55.82**	**55.28**	**56.90**	**83.98**	**44.30**	**54.08**	**55.04**	54.67	**81.73**
Reader: Llama3-8B (zero-shot)										
Base	2.0	7.07	6.91	7.07	59.05	0.64	7.00	6.77	8.72	59.71
Base w/ Table & Text (Zhang et al., 2023)	28.6	37.05	37.22	48.07	74.01	33.12	43.43	43.53	45.12	76.75
Base w/ Table & Summarized Text[2]	30.33	39.42	39.60	41.30	75.06	31.22	41.72	42.42	42.88	75.57
Our Method w/o hopwise	33.2	41.37	41.77	42.95	75.15	36.08	45.75	46.60	45.75	77.04
Our Method w/ hopwise	**37.0**	**46.43**	**46.56**	**48.78**	**77.55**	**37.13**	**47.38**	**48.24**	**48.31**	**77.62**

Table 3: **Table-Text QA Evaluation:** We analyze Exact Match (EM), F1-Score, Precision (P), Recall (R), and BERTScore-F1 (B) in (%) to compare our method against baselines in a zero-shot setting using Llama3-8B, GPT-3.5, and GPT-4. The results consistently demonstrate significant improvements across datasets, metrics, and various language models. *Base* (only reader LLM); *w/ Table & Text* (table and passages relevant to the question); *w/ Table & Summarized Text* (table with summarized supporting passages); *w/o hopwise* (pruned information without considering hop-wise extraction).

Method	EM (↑)	F1 (↑)
Hybrid-QA Fine-Tuning		
HYBRIDER (Chen et al., 2020b)	43.5	50.6
HYBRIDER-LARGE (Chen et al., 2020b)	44.0	50.7
DocHopper (Sun et al., 2021)	47.7	55.0
MuGER[2] (Wang et al., 2022)	57.1	67.3
S[3]HQA (Lei et al., 2023) [SoTA]	**68.4**	**75.3**
w/o Fine-Tuning		
Unsupervised-QG (Pan et al., 2021)	25.7	30.5
GPT-4† *w.* Retriever (Shi et al., 2024)	24.5	30.0
GPT-4† + CoT (Wei et al., 2022)	48.5	63.0
HPROPRO† (Shi et al. (2024), ACL 2024)	48.0	54.6
ODYSSEY† (Our Method)	**51.5**	**66.0**

Table 4: Performance on Hybrid-QA validation set. † stands for running on 200 sampled cases using gpt4-0613.

Method	EM (↑)	F1 (↑)
OTT-QA Fine-Tuning		
BM25-HYBRIDER (Chen et al., 2020a)	10.3	13.0
Fusion+Cross-Reader (Chen et al., 2020a)	28.1	32.5
CARP (Zhong et al., 2022)	33.2	38.6
CORE (Ma et al., 2022)	49.0	55.7
COS (Ma et al., 2023) [SoTA]	**56.9**	**63.2**
w/o Fine-Tuning		
GPT-4† + CoT (Wei et al., 2022)	61.0	72.3
ODYSSEY† (Our Method)	**62.02**	**73.02**

Table 5: Performance on OTT-QA validation set. † stands for running on 500 sampled cases using gpt4-1106-preview.

based method in conveying information to the LLM hop-wise.

5.3 Ablation Study

We present following ablation studies in this section: i) Hop-wise retrieval, and ii) Pruned Graph.

Hop-wise We passed pruned information all at once instead of hop-wise retrieval. This means all retrieved information (up to 3-hops) was sent together to the large language model (LLM) for question answering. As hypothesized, the performance of our method remained nearly the same (see 'Our Method w/o hopwise' in Table 3), with a slight decrease observed for GPT-3.5 and Llama3-8b on the Hybrid-QA dataset. Consequently, the average input token size for readers increased by nearly 20% for both datasets.

Pruned Graph We passed the entire constructed Hybrid Graph to the LLM without pruning. Table

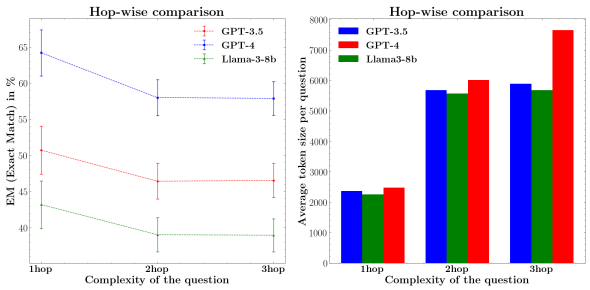

Figure 4: **Hopwise analysis:** For ODYSSEY (our method w/ hopwise), we calculate the cumulative EM score (left-side in figure) and average token size (right-side in figure) utilized by each hop for Llama3-8B, GPT-3.5, and GPT-4 on Hybrid-QA. Bars in left-side of the figure denotes standard error - $\sqrt{\frac{em(1-em)}{n}}$.

Method	Input Token Size (↓)	Input Token Cost (↓)
Dataset: Hybrid-QA		
Original Context	7195	$71.95
Summarized	3923	$39.23
Our Method	3857	$38.57
Dataset: OTT-QA		
Original Context	5866	$58.66
Summarized	3778	$37.78
Our Method	2745	$27.45

Table 6: **Reader Input Token Count and Cost**: We compare our method with baselines on average reader input token size and its pricing in dollars w.r.t. GPT-4 Turbo OpenAI pricing[10] for 1000 samples.

7 in Appendix A.1.1 shows results with complete hybrid graph in comparison to our method. The experiment conducted on the Hybrid-QA and OTT-QA dataset using GPT-4, resulted in a significant drop in performance compared to our method.

5.4 Error Analysis

We performed a detailed error analysis on 100 random samples from the HybridQA development set using our method, ODYSSEY, with GPT-4 in terms of EM score. Out of these, 41 answers were incorrect. The breakdown of our findings is as follows:

1. **Expression Mismatch (17 cases):** Our method provided semantically correct answers expressed differently from the gold standard. For example, it might output "hosted by Regis Philbin," while the standard answer is simply "Regis Philbin." This discrepancy lowers the EM score despite the answers being fundamentally accurate.

2. **Semantic Module Errors (14 cases):**
   - **Entity-Matching (8 cases):** Errors occur when our model fails to match similar yet non-identical entities in the question and text. For instance, "1,301 acres" does not match "1,301," leading to potential misalignment with other numbers.
   - **Entity-Extraction (2 cases):** These errors arise when our model struggles to extract complex or composite entities from the text.
   - **Entity-Header Mapping (4 cases):** Due to complex questions, our method (using GPT) sometimes fails to identify all relevant table headers. We are addressing this by employing all headers when none are identified.

3. **LLM Errors (3 cases):** The Large Language Model occasionally fails to provide the correct answer, despite having the necessary context, especially in response to complex questions.

4. **Dataset Issues (7 cases):** These errors stem from ambiguous questions or anomalies in the dataset, rather than limitations of our method.

6 Conclusion

In this paper, we introduce ODYSSEY, a zero-shot fine-tuning-free approach for Table-Text QA. By leveraging a novel Hybrid Graph-based approach, *Odyssey* effectively navigates the complexities of multi-hop reasoning across structured and unstructured data sources. Our method achieves significant improvements of 10% and 4.5% in Exact Match (EM) scores using GPT-3.5 on the Hybrid-QA and OTT-QA datasets, respectively, compared to the baseline, while reducing the input token size for the LLM reader by 45.5% on Hybrid-QA and 53% on OTT-QA, demonstrating its efficiency in representing relevant information concisely. We believe the insights gained from our method can pave the way for more advanced and efficient QA systems capable of navigating the ever-growing landscape of heterogeneous data sources.

Limitations

The limitations of our work are as follows: 1) While our method achieves the highest accuracy in a zero-shot setting with Llama3-8B, GPT-3.5,

[10]Pricing as of 15-June-2024 at https://openai.com/api/pricing/

and GPT-4, it incurs slightly more processing time than the zero-shot baseline due to additional LLM calls and Hybrid Graph traversal. 2) Our method's performance depends on current LLM capabilities, which may evolve over time. 3) We have only tested Tabular-Text data, whereas future work could explore various multi-modal datasets, including images and videos.

Acknowledgments

We would like to thank Pranoy Panda and the other members of the AI Lab at Fujitsu Research India for their valuable feedback on the manuscript. Our sincere thanks also go to Prof. Pushpak Bhattacharyya, Lokesh N, and Nihar Ranjan Sahoo from IIT Bombay for their insightful comments, which helped improve the final draft.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Yang Wang, and William W Cohen. 2020a. Open question answering over tables and text. In *International Conference on Learning Representations*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Julian Martin Eisenschlos, Maharshi Gor, Thomas Müller, and William W Cohen. 2021. Mate: multi-view attention for table transformer efficiency. *arXiv preprint arXiv:2109.04312*.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4099–4106.

Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Bowen Li, Jian Sun, and Yongbin Li. 2022. s2SQL: Injecting syntax to question-schema interaction graph encoder for text-to-SQL parsers. *arXiv preprint arXiv:2203.06958*.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne 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*, pages 9237–9251.

Hanlei Jin, Yang Zhang, Dan Meng, Jun Wang, and Jinghua Tan. 2024. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods. *arXiv preprint arXiv:2403.02901*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Vishwajeet Kumar, Yash Gupta, Saneem Chemmengath, Jaydeep Sen, Soumen Chakrabarti, Samarth Bharadwaj, and Feifei Pan. 2023. Multi-row, multi-span distant supervision for table+ text question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8080–8094.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Sung-Min Lee, Eunhwan Park, Daeryong Seo, Donghyeon Jeon, Inho Kang, and Seung-Hoon Na. 2023. Mafid: Moving average equipped fusion-in-decoder for question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2337–2344.

Fangyu Lei, Xiang Li, Yifan Wei, Shizhu He, Yiming Huang, Jun Zhao, and Kang Liu. 2023. S3hqa: A three-stage approach for multi-hop text-table hybrid question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational*

Linguistics (Volume 2: Short Papers), pages 1731–1740.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13067–13075.

Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. 2023b. Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13076–13084.

Ruosen Li and Xinya Du. 2023. Leveraging structured information for explainable multi-hop question answering and reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6779–6789.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2022. Open-domain question answering via chain of reasoning over heterogeneous knowledge. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5360–5374.

Kaixin Ma, Hao Cheng, Yu Zhang, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2023. Chain-of-skills: A configurable model for open-domain question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1599–1618.

Will Orr and Kate Crawford. 2023. The social construction of datasets: On the practices, processes and challenges of dataset creation for machine learning.

Liangming Pan, Wenhu Chen, Wenhan Xiong, Min-Yen Kan, and William Yang Wang. 2021. Unsupervised multi-hop question answering by question generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5866–5880.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8864–8880.

Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. 2024. Let your graph do the talking: Encoding structured data for llms. *arXiv preprint arXiv:2402.05862*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Yeon Seonwoo, Ji-Hoon Kim, Jung-Woo Ha, and Alice Oh. 2020. Context-aware answer extraction in question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2418–2428.

Qi Shi, Han Cui, Haofeng Wang, Qingfu Zhu, Wanxiang Che, and Ting Liu. 2024. Exploring hybrid question answering via program-based prompting. *arXiv preprint arXiv:2402.10812*.

Haitian Sun, William W Cohen, and Ruslan Salakhutdinov. 2021. Iterative hierarchical attention for answering complex questions over long documents. *arXiv preprint arXiv:2106.00200*.

Xiaoyu Tan, Haoyu Wang, Xihe Qiu, Yuan Cheng, Yinghui Xu, Wei Chu, and Yuan Qi. 2024. Structx: Enhancing large language models reasoning with structured data. *arXiv preprint arXiv:2407.12522*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037.

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

Yingyao Wang, Junwei Bao, Chaoqun Duan, Youzheng Wu, Xiaodong He, and Tiejun Zhao. 2022. Muger2: Multi-granularity evidence retrieval and reasoning for hybrid question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6687–6697.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 602–631.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2023. Tablellama: Towards open large generalist models for tables. *arXiv preprint arXiv:2311.09206*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Wanjun Zhong, Junjie Huang, Qian Liu, Ming Zhou, Jiahai Wang, Jian Yin, and Nan Duan. 2022. Reasoning over hybrid chain for table-and-text open domain qa. *arXiv preprint arXiv:2201.05880*.

A Appendix

In this section, we provide additional results and details that we could not include in the main paper due to space constraints. In particular, this appendix contains the following:

- Additional Results
- ODYSSEY Walkthrough with a Detailed Example
- Prompts used for ODYSSEY
- Prompts used for baselines

A.1 Additional Results

This section is divided into 2 parts: i) Input Complete Hybrid Graph, and ii) Results on Summarizing both the table and text. Additionally, it includes a table comparing the execution time per instance between our method and the baselines (see Table 9).

A.1.1 Input Complete Hybrid Graph

Table 7 compares our method with the Complete Hybrid Graph approach, where the entire Hybrid Graph is passed directly without any traversal or reduction. This experiment evaluates whether GPT-like LLMs can infer relationships between graph nodes based solely on the question, without additional context. The results demonstrate that our method surpasses the Complete Hybrid Graph in both EM and F1 scores while using fewer input tokens. The Complete Hybrid Graph method failed to provide answers for over 40% of the questions, and requiring larger token sizes due to full table-text passage inclusion. In contrast, our method is more accurate and efficient

A.1.2 Summarising both the table and text

For this baseline, we adopted the method proposed by LangChain[2], which involves summarizing tables and text and pass them as input to the LLM for QA. However, converting structured tables into summarized text led to a loss of information, resulting in lower scores (see Table 8). Although summarizing the context reduces token size, it often fails to effectively filter out noise and may erase relevant information. We address this limitation in our method by pruning relevant information.

A.2 ODYSSEY Walkthrough with a Detailed Example

In this section, we will understand ODYSSEY with an illustrated example. Below, we describe the context of the chosen example, including the question, table, and passages.

Question: The driver who finished in position 4 in the 2004 United States Grand Prix was of what nationality ?

Table Headers: [Pos, Driver, Constructor, Time, Gap]

The table contains 20 rows, with some cells linked to passages. We will list some of these passages below.

Passages:

1. Michael_Schumacher : Michael Schumacher (born 3 January 1969) is a retired German racing driver who raced in Formula One for Jordan Grand Prix , Benetton and Ferrari , where he spent most of his career , as well as for Mercedes upon his return to the sport . Widely regarded as one of the greatest Formula One drivers ever , and regarded by some as the greatest of all time , Schumacher is the only driver in history to win

Datasets	Hybrid-QA			OTT-QA		
Methods	EM (↑)	F1 (↑)	Input Token Size(↓)	EM (↑)	F1 (↑)	Input Token Size(↓)
Reader: gpt-4-1106-preview (zero-shot)						
Complete Hybrid Graph	54.0	65.30	5449	48.0	64.12	5763
Our Method	**58.0**	**71.80**	**3857**	**59.0**	**69.84**	**2745**

Table 7: **Comparison of EM, F1, and Avg Token Size across Hybrid-QA and OTT-QA datasets with GPT-4 model:** The table compares the performance between the Complete Hybrid Graph as Input and Our Method.

Methods	EM (↑)	F1 (↑)	B (↑)
GPT-4-1106-preview	31.92	43.19	74.64
GPT-3.5-turbo-1106	30.1	41.6	75.13
Llama3-8B	21.11	29.02	70.61

Table 8: Results for context w/ summarized table and summarized passages for the Hybrid-QA dataset.

Methods	Time (in seconds)
Base	1
Base w/ Table & Text	1.5
Base w/ Table and Summarized Text	7
Our Method w/o hopwise	3.6
Our Method w/ hopwise	5

Table 9: Per-instance execution time of each method presented in the main paper (in Table 3).

seven Formula One World Championships , five of which he won consecutively ....

2. Jenson_Button: Jenson Alexander Lyons Button MBE (born 19 January 1980) is a **British racing driver** and former Formula One driver . He won the 2009 Formula One World Championship , driving for Brawn GP . He currently competes in the Japanese Super GT Series driving a Honda NSX-GT for Team Kunimitsu , in which he won the title in 2018 . Button began karting at the age of eight and achieved early success ....

Question Analysis

In this phase, we extract relevant entities from the question, identify pertinent headers from all table headers in relation to the question, and then map the extracted entities to the identified headers.

Entity Extraction from the Question: ['driver', 'position 4', '2004 United States Grand Prix', 'nationality']

Relevant Headers: ['Pos', 'Driver']

Entity-Header Mapping: { 'driver': ['Driver'], 'position 4': ['Pos'], '2004 United States Grand Prix': ['Others'], 'nationality': ['Others'] }

Hybrid Graph Construction

After completing the question analysis phase, our next step is to jointly connect the table and passages for efficient retrieval. To achieve this, we construct the Hybrid Graph.

For Hybrid Graph construction, there are 2 processes:

- Table Retrieval: The relevant table headers are selected as sub-table, which becomes part of the Hybrid Graph.
- Passage Retrieval: Entities are extracted from all the passages, and an Entity-Document graph is created, linking each extracted entity to its corresponding passage or document.
  Example of entities extracted: ['Jenson Alexander Lyons Button MBE', '19 January 1980', 'British', 'Formula One', '2009', 'Formula One World Championship', 'Brawn GP', ..]

After retrieving the sub-table and creating the Entity-Document graph, we connect the entities from the passages to their corresponding table cells, forming the Hybrid Graph.

Hybrid Graph Traversal

With the entities extracted from the question, we traverse the Hybrid Graph up to 3 hops and store the results in a hop-wise dictionary. A sample of the dictionary is shown below for our example:

{"1-hop": ["4; Pos", "Driver", "Jenson Button"], "2-hop": ["Jenson Button; Driver", "entity", "British"], "3-hop":

["British", "Driver", "Juan Pablo Montoya"]}
In the dictionary, within each tuple, the word following ';' represents the header for the cell, while we track the entities originating from the table and those extracted from the passages.

LLM Reader

After obtaining the traversed graph, we pass it to the LLM hop-wise. Before providing it as input to the LLM, we preprocess our graph paths to form a sub-table, including the corresponding linked passages.

For this example, we pass the question along with the 1st hop of the dictionary. After preprocessing, we obtain the sub-table and the passages attached to the cells in the sub-table, which are then passed to the reader LLM for QA. The LLM identifies that 'Jenson Button' is the driver who achieved the position, and from the corresponding passage, it retrieves his nationality as 'British,' providing this as the output.

A.3 Prompts used for ODYSSEY

This section details the prompts used for our method, ODYSSEY, and is divided into: i) Question Analysis and ii) Final LLM Reader call.

A.3.1 Question Analysis

We start with Question Analysis, which uses a 1-shot example and follows three steps:

i. Entity Extraction - Extract entities from the question, using the table name and headers as additional input for efficiency.
ii. Relevant Table Headers - Extract useful headers from the table for answering the question, using the extracted entities as additional input.
iii. Entity-Header Mapping - Map the extracted question entities to the fetched table headers.

Entity Extraction from the Question

Agent Introduction: You are an agent who is going to be assisting me in a question answering task. For this task, I need to first identify the named entities in the question.

Task: Identify the named entities in the provided question. These entities will serve as key elements for extracting pertinent information from the available sources, which include table name and its headers.

Output format:

Entities: [‘<entity1>’, ‘<entity2>’, ...]

Use the below example to better understand the task

Input:

Question: What was the nickname of the gold medal winner in the men’s heavyweight Greco-Roman wrestling event of the 1932 Summer Olympics?
Table Name: Sweden at the 1932 Summer Olympics
Table Headers: ["Medal", "Name", "Sport", "Event"]

Output:

Entities: [‘nickname’, ‘medal’, ‘gold’, ‘men’s heavyweight’, ‘Greco-Roman Wrestling event’, ‘1932 Summer Olympics’]

Input:

Question: {question}
Table Name: {table_id}
Table Headers: {table_headers}

Output:

Relevant Header Extraction

Agent Introduction: You are an agent who is going to be assisting me in a question answering task. I have a table as a source of information. I have already extracted the relevant entities from the question. For this task, I need to first identify the column headers that are relevant in the question.

Task: Identify the relevant column headers from the provided list, based on the extracted entities from the question. I will also provide the extracted entities from the question and name of the table.

Output format:

Relevant headers: [‘<header-1>’, ‘<header-2>’, ...]

Use the below example to better understand the task

Input:
Question: What was the nickname of the gold medal winner in the men’s heavyweight Greco-Roman wrestling event of the 1932 Summer Olympics?
Table Name: Sweden at the 1932 Summer Olympics
Table Headers: ["Medal", "Name", "Sport", "Event"]
Entities extracted from question: ["gold medal", "men’s heavyweight", "Greco-Roman Wrestling", "1932 Summer Olympics"]

Output:

Relevant headers: ["Medal", "Name", "Sport", "Event"]

Input:

Question: {question}
Table Name: {table_id}
Table Headers: {table_headers}
Entities extracted from question: {entities}

Output:

Entity to Header Mapping

Agent Introduction: You are an agent who is going to be assisting me in a question answering task. I have a table as a source of information. I have already extracted relevant entities from the question and relevant column headers from the table.

Task: Map the entities extracted from the question with the relevant headers and the table name.

Output format:

"<entity1>": ["<mapping1>", "<mapping2>"],
"<entity2>": ["<mapping1>"]

For each entity extracted from the question, there should be a corresponding <mapping> to an item in the ‘Relevant headers’ column. If none of the headers match the entity, the mapping should be labeled as "Others".

Use the below example to better understand the task

Input:
Question: What was the nickname of the gold medal winner in the men’s heavyweight Greco-Roman wrestling event of the 1932 Summer Olympics?
Table Name: Sweden at the 1932 Summer Olympics
Entities extracted from question: ["gold medal", "men’s heavyweight", "Greco-Roman Wrestling", "1932 Summer Olympics"]
Relevant headers: ["Medal", "Name", "Sport", "Event"]

Output:

"gold medal": ["Medal"],
"men's heavyweight": ["Event"],
"Greco-Roman Wrestling": ["Sport"],
"1932 Summer Olympics": ["Others"]

Input:
Question: {question}
Table Name: {table_id}
Entities extracted from question: {entities}
Relevant Headers: {relevant_headers}

Output:

A.3.2 LLM Reader

This is the final LLM call after Question Analysis for QA, operating in a zero-shot setting.

Hybrid Question Answering for ODYSSEY

Agent Introduction: Hello! I'm your Hybrid-QA expert agent, here to assist you in answering complex questions by leveraging both table data and passage information. Let's combine these sources to generate accurate and comprehensive answers!

Task: Your task involves a central question that requires information from both a table and passages.

Here's the context you'll need:

Table Data: {table_data}

Passages: {passages}

Question: {question}

Final Answer: Provide the final answer in the format below. If the answer cannot be answered with the given context, provide None.

Final Answer Format:

Final Answer: <your answer>

If the final answer is "None", provide the names of passages that are relevant to the above questions.

If no passages are relevant give '[]' as Relevant Passages.

Relevant Passages Format:

Relevant Passages: ['<name-of-passage1>', '<name-of-passage2>', ...]

A.4 Prompts used for baselines

We compare our method with two baselines using context for QA: i) Summarized Context and ii) Complete Context. For table summarization, we pass the entire table to the LLM for summarization. For text, each passage is passed individually. The final LLM reader prompt is the same for both methods.

A.4.1 Table and Text Summarization

Table Summarization Task

Agent Introduction: You are an assistant tasked with summarizing tables.

Task: I have a table that I need help summarizing. The table contains the following columns and data: {table}
Provide a concise summary of the table without removing any numbers, entities or important information. Try to retain all the important information.

Response:

Passage Summarization Task

Agent Introduction: You are an assistant tasked with summarizing passages.

Task: I have a passage that I need help summarizing. The passage is as follows: {text}
Provide a concise summary of the passage without removing any numbers, entities or important information. Try to retain all the important information.

Response:

A.4.2 LLM Reader

This is the LLM call for baselines operating in a zero-shot setting.

Hybrid Question Answering

Agent Introduction: Hello! I'm your Hybrid-QA expert agent, here to assist you in answering complex questions by leveraging both table data and passage information. Let's combine these sources to generate accurate and comprehensive answers!

Task: Your task involves a central question that requires information from both a table and passages.

Here's the context you'll need:

Table Data: {table_data}

Passages: {passages}

Question: {question}

Final Answer: <your answer>