Improving Table Retrieval with Question Generation from Partial Tables

Hsing-Ping Liang, Che-Wei Chang, Yao-Chung Fan*
Department of Computer Science and Engineering,
National Chung Hsing University, Taiwan
yfan@nchu.edu.tw

Abstract

Recent advances in open-domain question answering over tables have widely adopted large language models (LLMs) under the Retriever-Reader architecture. Prior works have effectively leveraged LLMs to tackle the complex reasoning demands of the Reader component, such as text-to-text, text-to-SQL, and multi-hop reasoning. In contrast, the Retriever component has primarily focused on optimizing the query representation—training retrievers to retrieve relevant tables based on questions, or to select keywords from questions for matching table segments. However, little attention has been given to enhancing how tables themselves are represented in embedding space to better align with questions. To address this, we propose QGpT (Question Generation from Partial Tables), a simple yet effective method that uses an LLM to generate synthetic questions based on small portions of a table. These questions are generated to simulate how a user might query the content of the table currently under consideration. The generated questions are then jointly embedded with the partial table segments used for generation, enhancing semantic alignment with user queries. Without the need to embed entire tables, our method significantly improves retrieval performance across multiple benchmarks for both dense and late-interaction retrievers.[1]

1 Introduction

Table-based question answering (Table-QA) has drawn increasing attention in recent years, beginning with early works on Wiki-style tables (Pasupat and Liang, 2015; Zhong et al., 2017). These early Table-QA tasks typically operate under the assumption that the relevant table is provided alongside the question. While this assumption simplifies the problem and allows a focused evaluation of reasoning over structured data, it fails to reflect the challenges of real-world open-domain usage scenarios.

In practical settings, users do not typically specify which table to consult. Instead, they ask questions in natural language, and the system must first determine which tables might contain the relevant information before reasoning. The assumption of a known target table is thus insufficient in many realistic applications, where the answer may reside in any one of potentially thousands of tables within a large corpus.

To address this gap, recent research has moved toward integrating table retrieval into the QA pipeline. This shift is marked by the adoption of Retriever-Reader architectures (Chen et al., 2017), where a retriever component is responsible for identifying candidate tables, followed by a reader module that performs QA over the retrieved tables.

As a pioneering study, NQ-TABLES (Herzig et al., 2021) provides naturally phrased questions and their corresponding answer tables extracted from the Natural Questions dataset (Kwiatkowski et al., 2019), and transforms TAPAS (Herzig et al., 2020), a BERT-based table reader, into a dense table retriever (DTR) by adopting the DPR (Karpukhin et al., 2020) training paradigm.

Yet another study named CLTR (Pan et al., 2021) integrates the RCI (Row-Column Intersection; Glass et al., 2021) model with BM25 (Robertson et al., 2009) for table retrieval, and introduces the E2E-WTQ dataset for RCI retriever fine-tuning.

However, studies such as NQ-TABLES and CLTR operate under the assumption that the answer to a question resides in a specific cell within a single, relatively short table. While this setting facilitates controlled evaluation, it significantly limits the generalizability of these approaches to real-world applications, where answering questions often requires reasoning over multiple or lengthy tables.

Therefore, MMQA (Wu et al., 2025) extend the

[1]The code and reconstructed corpora are available at https://github.com/cc3374twa/QGpT

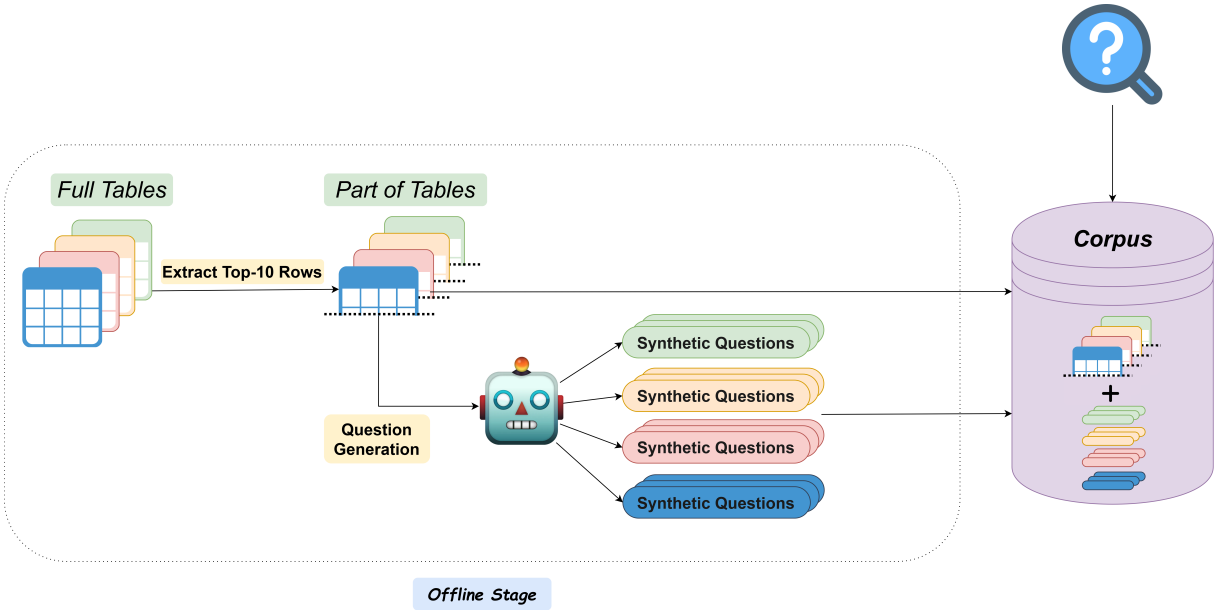

Figure 1: The **QGpT** pipeline: In the offline stage, top-10 rows from full tables are used to generate synthetic questions via LLM. The questions and table snippets are embedded and stored in the corpus, enhancing retrieval through semantic alignment without encoding full tables.

Table-QA paradigm by introducing complex reasoning tasks that integrate question answering with Text-to-SQL, using the Spider dataset (Yu et al., 2018) as a foundation. They propose a *multi-table retrieval* (MTR) setting, in which a single question may require retrieving and reasoning over multiple relevant tables. Their approach decomposes the original query into sub-queries, enabling independent retrieval and re-ranking of candidate tables.

Building on the idea of query decomposition, TableRAG (Chen et al., 2024b) adopts a similar strategy by splitting questions into schema-level and cell-level components and performing separate retrieval for each. Additionally, it segments tables into schema and content parts, allowing scalable retrieval across corpora with millions of cells, and focuses on retrieving localized table segments to improve efficiency and accuracy.

In summary, Table-QA has evolved from simple single-table, short-answer settings (Pasupat and Liang, 2015; Zhong et al., 2017; Herzig et al., 2021; Chen et al., 2020) to more complex, multi-table, and reasoning-intensive tasks (Wu et al., 2025; Li et al., 2025). Nevertheless, practical deployment remains challenging due to the growing size and noise of real-world tables. Traditional retrievers (Robertson et al., 2009; Karpukhin et al., 2020; Khattab and Zaharia, 2020) struggle to capture full-table context in token limits, while more recent approaches (Pan et al., 2021; Lin et al., 2023; Chen et al., 2024b) primarily rely on table segmentation based on rows, columns, or schema structures to improve matching between query keywords and table fragments. These methods focus on keyword-level matching rather than learning a semantically rich representation of the table that aligns with user questions. Moreover, they are often bound to dataset-specific assumptions or require retriever fine-tuning (Pan et al., 2021; Lin et al., 2023).

To tackle this issue, we propose Question Generation from Partial Tables (QGpT), a simple yet effective table retrieval method for long and complex Table-QA tasks. Our approach requires only a small snippet of the table and leverages LLMs to generate simulated questions that are likely to be asked. These questions are jointly embedded with the partial table snippet, enabling a dense representation with minimal token budget while improving retrieval accuracy.

As shown in Figure 1, QGpT is applied during the offline phase to augment the table corpus. During inference, it can adapt to various retrievers without fine-tuning, and enhances the performance of both dense and late-interaction retrievers (Chen et al., 2024a; Jha et al., 2024). Our QGpT framework offers a lightweight and generalizable solution that improves semantic alignment between questions and tables, reduces context size, and maintains retrieval performance in increasingly large and complex table settings.

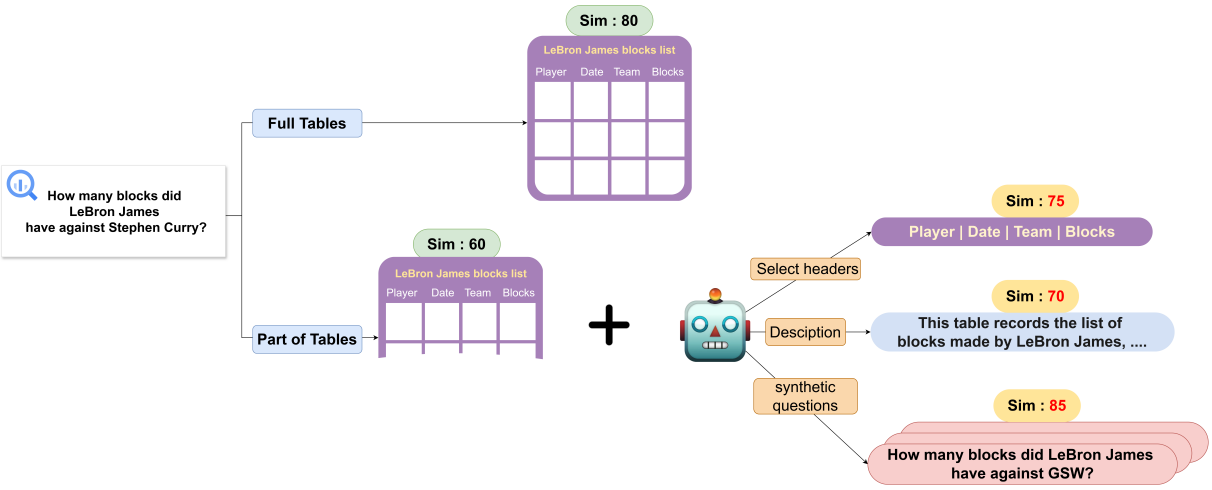

Figure 2: An illustration of three strategies for enriching truncated tables to enhance semantic alignment with the input query: selecting headers, generating table descriptions, and generating synthetic questions.

2 Related Work

2.1 Table Question Answering (TableQA)

Early TableQA tasks typically assume the target table is known, focusing on reasoning within a single table. Wiki-TableQuestions (Pasupat and Liang, 2015) is a pioneering dataset that provides semi-structured Wikipedia tables with questions involving simple answering. WikiSQL (Zhong et al., 2017) reframes the task as a Text-to-SQL problem, enabling structured query-based answering. Spider (Yu et al., 2018) introduces higher complexity by requiring reasoning over multiple tables and supporting diverse SQL logic.

Subsequent works began exploring more natural question formulations and open-ended scenarios. FeTaQA (Nan et al., 2022) introduced free-form, multi-hop reasoning questions requiring sentence-level answers across multiple rows. OpenWikiTable (Kweon et al., 2023) expanded WikiSQL and WikiTableQuestions by incorporating a larger table collection and more natural question expressions.

More recently, MimoTable (Li et al., 2025) uses real-world spreadsheets of varying size and complexity, containing multi-sheet structures and nested tables. It categorizes difficulty levels based on file count, number of sheets, and header complexity, making it one of the most comprehensive TableQA benchmarks.

Overall, the evolution of TableQA datasets has shifted from "single-table, structured QA pairs" to "multi-table, semantically ambiguous, natural language" tasks. However, most of these datasets do not consider the open-domain setting where tables must be retrieved from a large corpus before answering.

2.2 Table Retrieval in Open-Domain QA

A core challenge in open-domain TableQA is efficiently retrieving relevant tables from large corpora. OTT-QA (Chen et al., 2020) introduces table retrieval from Wikipedia to support open-domain question answering, highlighting the need for integrated retriever-reader systems. NQ-TABLES (Herzig et al., 2021) converts questions from Natural Questions (Kwiatkowski et al., 2019) into table-based QA pairs and incorporates a retrieval subtask. E2E-WTQ (Pan et al., 2021) extends WikiTableQuestions into a retrieval setting and proposes Cell-Level Table Retrieval (CLTR), focusing on cell-level semantic matching.

LI-RAGE (Lin et al., 2023) combines the concept of joinable tables with late-interaction retrievers (Khattab and Zaharia, 2020; Santhanam et al., 2022b), enabling joint training of retrievers and readers for better performance. TableRAG (Chen et al., 2024b) decomposes questions and tables into schema-level and cell-level components, enabling compression of million-cell tables into retrievable chunks.

MMQA (Wu et al., 2025) introduces Multi-Table Retrieval (MTR), which leverages LLMs like GPT-4 (Achiam et al., 2023) for query decomposition without retriever fine-tuning. It further employs LLM-based retrievers such as TableLlama (Zhang et al., 2024) and SGPT (Muennighoff, 2022).

While advances in retriever architectures—from sparse (Robertson et al., 2009) to dense (Karpukhin

Retriever	Table size	R@1	R@3	R@5	R@10	Avg
BGE-m3-dense	Full-Table (8K)	44.68 38.20	61.82 54.06	67.62 62.52	75.30 70.07	62.36 56.21
	1K tokens	46.09 40.27	**62.83** **56.22**	**68.90** 63.47	**77.46** 70.06	**63.82** 57.51
	2K tokens	45.15 40.15	62.75 55.62	68.16 **63.70**	76.70 70.61	63.19 57.52
	5K tokens	44.53 38.86	61.82 55.26	67.69 62.69	75.53 70.32	62.39 56.78
	Top10-rows	**46.24** **40.52**	62.29 55.99	67.77 62.92	75.22 **70.98**	62.88 **57.60**
Jina-ColBERT-v2	Full-Table (8K)	42.15 26.53	57.59 37.97	61.93 43.67	69.06 54.06	57.68 40.56
	1K tokens	48.82 **37.56**	64.24 **53.11**	**70.40** **58.61**	**75.25** **67.09**	64.68 **54.09**
	2K tokens	46.67 32.09	61.10 45.69	66.22 52.44	72.10 62.24	61.52 48.12
	5K tokens	42.62 26.70	56.97 39.22	62.09 44.44	68.12 54.50	57.45 41.22
	Top10-rows	**51.74** 36.70	**64.47** 51.21	69.23 57.30	74.38 66.11	**64.96** 52.83

Table 1: Recall@k performance on the **MiMoTable-English** dataset (normal font) and **MiMoTable-Chinese** dataset (shown in next to each value) across different retrievers and table representation lengths. Note that all table titles in table embeddings are excluded . Best and second-best scores are bolded and underlined respectively per language.

Statistics	English	Chinese
Number of Tables	206	295
Number of Sheets	295	464
Number of Queries	641	995
Max Tokens per Sheet	8,974	1,227,845
Tokens / sheet		
<1k	35%	29%
1k–2k	41%	36%
2k–5k	20%	22%
>5k	4%	13%

Table 2: Comparison of MiMoTable-English and MiMoTable-Chinese statistics.

et al., 2020; Herzig et al., 2020) and late-interaction models (Khattab and Zaharia, 2020; Santhanam et al., 2022b; Lin et al., 2023)—have significantly improved retrieval performance, our proposed framework is agnostic to the underlying retriever. It can flexibly integrate with various retrieval paradigms and consistently enhance performance across different backbone models.

3 Methodology

3.1 Partial Table Selection

To support complex TableQA tasks, it is essential to reduce large tables by removing irrelevant cells. A common approach is to retain only the header / schema or to constrain the input by a fixed number of tokens or rows. However, the former is heavily dependent on the nature of the dataset: SQL-based datasets typically rely heavily on headers, while in datasets like MiMoTable (Li et al., 2025), which aim to increase difficulty, headers are not necessarily positioned in the first row and may even be multi-level.

To inform our strategy for partial table selection, we first analyze the length distribution of tables in the MiMoTable dataset (see Table 2). Given the wide variance in table sizes, we compare two representative approaches for truncation: limiting by token length and selecting the top-10 rows. We construct English and Chinese table corpora accordingly and evaluate retrieval performance using BGE-m3 (Chen et al., 2024a) and Jina-ColBERT-v2 (Jha et al., 2024)—both supporting up to 8K tokens—as our retrievers. As shown in Table 1, the top-10 rows selection achieves comparable performance to 1K tokens truncation in complex table QA tasks. Considering the diversity of table lengths across datasets and the simplicity of implementation, we adopt the top-10 rows as our default strategy for partial table input.

To further improve semantic alignment between user questions and compressed table inputs, we explore several strategies for enriching table representations. As illustrated in Figure 2, these include selecting table headers, generating natural language descriptions, and producing synthetic questions by LLMs. To realize these strategies, we leverage LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024) to implement these strategies. Table 3 presents the retrieval results using each method. Interestingly, even without table content, simulated

Retriever	Method	R@1	R@3	R@5	R@10	Avg
BGE-m3-dense	pT	44.27	61.95	67.25	75.27	62.19
	header-only	33.06	49.18	57.59	65.04	51.28
	desc-only	36.05	51.85	61.04	71.48	55.10
	QG-only	48.09	64.47	**72.39**	79.28	66.06
	pT + header	45.55	62.83	68.81	76.95	63.54
	pT + desc	45.85	64.47	71.41	78.99	65.18
	QGpT	**50.66**	**66.42**	72.35	**80.80**	**67.58**
Jina-ColBERT-v2	pT	54.25	69.16	75.83	81.97	70.30
	header-only	51.14	67.42	74.14	80.72	68.36
	desc-only	47.37	65.21	72.04	81.49	66.53
	QG-only	52.76	70.10	75.17	80.45	69.62
	pT + header	57.72	74.04	80.08	85.34	74.29
	pT + desc	**58.53**	74.71	**80.40**	**86.15**	**74.95**
	QGpT	57.94	**75.21**	79.11	83.26	73.88

Table 3: Recall@k results on the **MiMoTable-English** dataset comparing different table representation strategies and retrievers. The base table input *pT* corresponds to the top-10 rows of each table. Additional representations—headers, descriptions, and questions—are generated using the **LLaMA3.1-8B-Instruct** model. We evaluate combinations such as *pT* with *header*, *desc* and *QG*. Scores are reported using dense and Multi-vector retriever.

Dataset	#Q	#Tables	Type
OTT-QA	2.2K	789	TQA
FeTaQA	2K	2K	TQA
E2E-WTQ	241	2.1K	TQA
MiMoTable (en)	641	295 sheets	Long, TQA
MMQA (2-Tbl)	2591	2591 / 644	Long, MTR,
MMQA (3-Tbl)	721	721 / 391	TQA, SQL

Table 4: Dataset statistics. For MMQA, we report both the original and reconstructed table counts (original/ours).

questions alone can achieve comparable or better performance than partial tables. Combining simulated questions with partial tables further improves retrieval performance consistently. Motivated by these findings, we propose a unified table representation method—**QGpT** (Question Generation from Partial Tables)—which augments the top-10 table rows with generated questions to construct an enriched table corpus.

3.2 Question Generation from Partial Tables

QGpT is highly extensible and model-agnostic. The simulated questions are generated during an offline preprocessing stage, enabling integration with various retrieval paradigms (e.g., query decomposition or different retrievers) during online inference.

Offline Stage Given a table corpus $\mathcal{C}_T$, we convert each table into a markdown format and extract its name and the top-10 rows to construct a new partial table corpus $\mathcal{C}_t$. For each partial table $t_i \in \mathcal{C}_t$, we use a language model M to generate a set of questions $\{q_{t_i,j}\}$ such that the number of generated questions j satisfies:

$$j \geq \left\lceil \frac{|\mathcal{H}_{t_i}|}{2} \right\rceil$$

where $|\mathcal{H}_{t_i}|$ is the number of headers in Table t_i. The resulting *augmented* partial table $t'_i = (t_i, q_{t_i,1}, q_{t_i,2}, \ldots, q_{t_i,j})$ is then embedded using an embedding model E, producing a set of vectors for the table corpus:

$$E(\mathcal{C}_{t'}) = \{E(t'_1), E(t'_2), \ldots, E(t'_n)\}$$

Online Stage Given a user query $q_i \in \mathcal{Q}$, we compute its embedding $E(q_i)$ and perform cosine similarity with all table representations:

$$\text{sim}(E(q_i), E(t'_j)) \quad \forall j \in [1, n]$$

The top-k most similar entries are then retrieved.

4 Experiments

4.1 Experimental Settings

Datasets For the Single Table Retrieval task, we conduct experiments on OTT-QA (Chen et al.,

Model	Method & Recall@k		Dataset				Avg
			MimoTable	OTTQA	FetaQA	E2E-WTQ	
BGE-m3 dense	pT	R@1	44.27	52.17	31.75	39.83	42.01
	+QGpT		50.66 $_{\uparrow 6.39}$	51.45 $_{\downarrow 0.72}$	33.95 $_{\uparrow 2.20}$	41.49 $_{\uparrow 1.66}$	44.39 $_{\uparrow 2.38}$
	pT	R@5	67.25	78.27	48.08	59.75	63.34
	+QGpT		72.35 $_{\uparrow 5.10}$	78.14 $_{\downarrow 0.13}$	50.87 $_{\uparrow 2.79}$	65.98 $_{\uparrow 6.23}$	66.84 $_{\uparrow 3.50}$
	pT	R@10	75.27	86.04	55.62	70.54	71.87
	+QGpT		80.80 $_{\uparrow 5.53}$	86.68 $_{\uparrow 0.64}$	57.86 $_{\uparrow 2.24}$	72.61 $_{\uparrow 2.07}$	74.49 $_{\uparrow 2.62}$
Jina-ColBERT-v2	pT	R@1	54.25	54.43	35.30	48.55	48.13
	+QGpT		57.94 $_{\uparrow 3.69}$	55.15 $_{\uparrow 0.72}$	37.19 $_{\uparrow 1.89}$	51.45 $_{\uparrow 2.90}$	50.43 $_{\uparrow 2.30}$
	pT	R@5	75.83	76.87	50.82	65.56	67.27
	+QGpT		79.11 $_{\uparrow 3.28}$	78.73 $_{\uparrow 1.86}$	52.17 $_{\uparrow 1.35}$	71.37 $_{\uparrow 5.81}$	70.35 $_{\uparrow 3.08}$
	pT	R@10	81.97	83.83	57.36	70.95	73.53
	+QGpT		83.26 $_{\uparrow 1.29}$	86.04 $_{\uparrow 2.21}$	58.61 $_{\uparrow 1.25}$	76.76 $_{\uparrow 5.81}$	76.17 $_{\uparrow 2.64}$

Table 5: Single-table retrieval performance (Recall@k) across four QA datasets using two retrievers. The base method *pT* uses the top-10 table rows, while *QGpT* denotes the enhancement via question generation. Only embeddings for **OTTQA** exclude table titles, while all other datasets include them. ↑ indicates the improvement over the corresponding *pT* baseline.

2020) and FeTaQA (Nan et al., 2022) following the evaluation settings proposed by TARGET (Ji et al., 2024), as well as on E2E-WTQ (Pan et al., 2021) and MiMoTable-English (Li et al., 2025).

For the Multi-Table Retrieval task, we evaluate on the MMQA dataset (Wu et al., 2025). However, the original MMQA dataset only provides question-table-answer triples without releasing a unified table corpus. Moreover, tables with the same `table_name` may differ in structure across different examples—some pointing to semantically distinct tables despite identical names. To ensure consistency and reproducibility, we reconstruct the MMQA table corpus by performing a schema-based deduplication. Specifically, we (1) group tables by `table_name` and enumerate distinct schema variants, (2) assign unique identifiers (e.g., `department__1`, `department__2`) for structurally distinct tables, and (3) update all question-table mappings accordingly. This process results in a flattened and de-duplicated table corpus that supports robust retrieval evaluation. A summary of all datasets used can be found in Table 4.

Models Throughout all experiments, we adopt BGE-m3 (Chen et al., 2024a) and Jina-ColBERT-v2 (Jha et al., 2024) as base retrievers, which support up to 8K tokens and represent dense and late-interaction paradigms, respectively. All question generation is performed using LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024), with prompt details provided in Appendix A.

To analyze the impact of model capacity on query decomposition in Multi-Table Retrieval (MTR), we experiment with multiple large language models, including LLaMA-3.1-8B-Instruct, GPT-4o-mini, and GPT-4o (Hurst et al., 2024), within the MTR framework.

Evaluation Metrics We report performance using the **Recall@K** metric across all experiments. For Multi-Table Retrieval, we follow the MMQA evaluation settings with $K = \{2, 5, 10\}$.

4.2 Implementation Details

All experiments are conducted using two NVIDIA RTX A6000 GPUs. We use RAGatouille[2] and Milvus[3] (Wang et al., 2021) as the retrieval infrastructure for vector indexing and search for Jina-ColBERT-v2 and BGE-m3, respectively. For Milvus, we adopt `index_type=IVF_FLAT`, `metric_type=IP`, and `nlist=256` to balance retrieval speed and accuracy. For RAGatouille, we leverage PLAID (Santhanam et al., 2022a), a high-performance indexing engine specifically designed for late interaction retrievers, enabling efficient

[2] https://github.com/AnswerDotAI/RAGatouille
[3] https://milvus.io/

Retriever	Method	R@2	R@5	R@10	Avg
BGE-m3-dense	pT	48.65 (39.36)	68.54 (61.10)	80.94 (74.69)	62.55
	MTR (LLaMA3.1-8b)	44.60 (36.50)	65.32 (60.02)	79.07 (74.15)	59.94
	+QGpT	49.63 (38.81)	68.33 (62.17)	80.13 (**75.86**)	62.49
	MTR (GPT4o-mini)	45.60 (36.51)	67.32 (59.51)	79.80 (72.61)	60.23
	+QGpT	50.19 (37.93)	71.37 (61.24)	**82.58** (74.86)	63.03
	MTR (GPT4o)	46.15 (36.41)	67.79 (58.90)	81.05 (73.17)	60.58
	+QGpT	51.10 (38.09)	**71.89** (60.40)	82.13 (74.45)	63.01
	QGpT	**52.24** (**40.05**)	71.47 (**63.36**)	82.33 (75.36)	**64.14**
Jina-ColBERT-v2	pT	58.18 (45.79)	77.87 (70.16)	87.09 (81.63)	70.12
	MTR (LLaMA3.1-8b)	54.34 (42.65)	73.90 (66.84)	83.93 (78.30)	66.66
	+QGpT	54.28 (43.52)	74.98 (67.67)	84.22 (79.71)	67.40
	MTR (GPT4o-mini)	56.40 (41.24)	75.27 (65.18)	86.36 (75.96)	66.74
	+QGpT	56.33 (42.15)	76.18 (66.81)	86.17 (78.11)	67.63
	MTR (GPT4o)	56.76 (42.15)	75.82 (65.72)	86.80 (76.53)	67.30
	+QGpT	56.99 (43.23)	76.91 (67.82)	86.91 (77.97)	68.31
	QGpT	**59.49** (**46.75**)	**78.41** (**71.43**)	**87.25** (**83.14**)	**71.08**

Table 6: Recall@k performance on the **MMQA** dataset with different retrievers and query methods. Metrics reflect performance on the 2-table setting; scores in parentheses show corresponding results under the 3-table setting (in small font). All methods are built upon the *pT* baseline (top-10 table rows). *MTR* uses sub-query generation via **LLaMA3.1-8B-Instruct**, **GPT-4o**, or **GPT-4o-mini**. *QGpT* questions are generated using **LLaMA3.1-8B-Instruct**.

token-level matching at scale.

5 Experimental Results

5.1 Main Results

Single Table Retrieval We evaluate retrieval performance across datasets ranging from short and simple to long and complex tables. As shown in Table 5, QGpT consistently improves retrieval performance across all Single Table QA datasets compared to using partial tables alone. Notably, on the MiMoTable dataset—which features longer and more complex tables—both the dense and late-interaction retrievers benefit significantly from QGpT. This demonstrates that QGpT is particularly effective in scenarios where aggressive table compression is required, offering robust performance regardless of table complexity.

Multi Table Retrieval Building on the same partial table setup, we compare Multi-Table Retrieval (MTR), QGpT, and their combination on the MMQA dataset. Note that our evaluation is based on a reconstructed table corpus. Therefore, our results are not directly comparable to those reported in the original paper.

Table 6 shows that while MTR performance slightly improves with larger query decomposition models (e.g., GPT-4o), it still underperforms compared to directly using partial tables for retrieval. This gap may stem from differences in implementation, such as our exclusion of the original paper's hand-crafted one-shot examples or the use of full tables during retrieval. However, integrating QGpT into MTR substantially closes the performance gap, and QGpT alone consistently outperforms the baseline across settings. These results highlight QGpT's effectiveness in both single- and multi-table retrieval, providing strong gains even when only limited table information is available.

5.2 Ablation studies

Can Simulated Questions Bridge the Semantic Gap? To understand whether simulated questions truly enhance the semantic alignment between partial tables and questions, we conduct an ablation study on the OTT-QA dataset. As noted by TARGET (Ji et al., 2024), OTT-QA questions often align closely with table titles, and excluding titles from the corpus can dramatically reduce BM25 Recall@10 from 95% to 44%.

In our study, we generate two versions of simulated questions using QG models: one with access

Model & Method	R@1	R@3	R@5	R@10	Avg
BGE-m3-dense (pT)	52.17	70.73	78.27	86.04	71.80
+QGpT w/o title	51.45 $_{\downarrow 0.72}$	70.64 $_{\downarrow 0.09}$	78.14 $_{\downarrow 0.13}$	86.68 $_{\uparrow 0.64}$	71.23 $_{\downarrow 0.57}$
+QGpT w/ title	60.79 $_{\uparrow 8.62}$	78.14 $_{\uparrow 7.41}$	84.37 $_{\uparrow 6.10}$	91.46 $_{\uparrow 5.42}$	78.69 $_{\uparrow 6.89}$
Jina-ColBERT-v2 (pT)	54.43	70.01	76.87	83.83	71.29
+QGpT w/o title	55.15 $_{\uparrow 0.72}$	71.27 $_{\uparrow 1.26}$	78.73 $_{\uparrow 1.86}$	86.04 $_{\uparrow 2.21}$	72.80 $_{\uparrow 1.51}$
+QGpT w/ title	60.07 $_{\uparrow 5.64}$	75.34 $_{\uparrow 5.33}$	80.80 $_{\uparrow 3.93}$	87.71 $_{\uparrow 3.88}$	75.98 $_{\uparrow 4.70}$

Table 7: Recall@k results on the **OTT-QA** dataset with different retrievers and *QGpT* enhancements. All embedding representations exclude table titles, while *QGpT* question generation is conducted **with or without referencing table titles**.

to table titles, and another without. We then embed both variants alongside partial tables that exclude the titles. As shown in Table 7, incorporating simulated questions generated with access to titles leads to significant performance improvements, outperforming both the baseline and the QGpT variant that does not use titles. These results validate that simulated questions can effectively bridge the semantic gap, helping partial tables better align with user queries during retrieval.

6 Conclusion

In this work, we propose QGpT (Question Generation from Partial Tables), a simple yet effective framework to enhance table retrieval by bridging the semantic gap between compressed table inputs and user queries. By leveraging LLMs to generate simulated questions from partial tables, QGpT provides a semantically enriched representation that improves retrieval performance across both single- and multi-table QA benchmarks.

Our extensive experiments demonstrate that QGpT consistently outperforms traditional partial table baselines on diverse datasets, including long, noisy, and multi-table settings. Notably, the framework is model-agnostic and can be flexibly integrated with different retrievers without requiring fine-tuning.

Limitations

While QGpT offers a generalizable solution for enhancing table retrieval, several limitations remain:

LLM dependency: The quality of simulated questions relies heavily on the capabilities of the underlying LLM. Lower-quality LLMs may generate irrelevant or redundant questions, limiting retrieval gains.

Generation latency: Although question generation occurs offline, large-scale preprocessing for hundreds of thousands of tables may introduce overhead in real-world deployments.

Acknowledgement

This work is supported by NSTC 112-2634-F-005-002-project Smart Sustainable New Agriculture Research Center (SMARTer), NSTC Taiwan Project under grant 112-2221-E-005-075-MY3, and Delta Research Center.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Si-An Chen, Lesly Miculicich, Julian Eisenschlos, Zifeng Wang, Zilong Wang, Yanfei Chen, Yasuhisa Fujii, Hsuan-Tien Lin, Chen-Yu Lee, and Tomas Pfister. 2024b. Tablerag: Million-token table understanding with language models. *Advances in Neural Information Processing Systems*, 37:74899–74921.

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

Michael Glass, Mustafa Canim, Alfio Gliozzo, Saneem Chemmengath, Vishwajeet Kumar, Rishav Chakravarti, Avi Sil, Feifei Pan, Samarth Bharadwaj, and Nicolas Rodolfo Fauceglia. 2021. Capturing row and column semantics in transformer based question answering over tables. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1212–1224, Online. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Eisenschlos. 2021. Open domain question answering over tables via dense retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 512–519, Online. Association for Computational Linguistics.

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

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Rohan Jha, Bo Wang, Michael Günther, Georgios Mastrapas, Saba Sturua, Isabelle Mohr, Andreas Koukounas, Mohammad Kalim Wang, Nan Wang, and Han Xiao. 2024. Jina-ColBERT-v2: A general-purpose multilingual late interaction retriever. In *Proceedings of the Fourth Workshop on Multilingual Representation Learning (MRL 2024)*, pages 159–166, Miami, Florida, USA. Association for Computational Linguistics.

Xingyu Ji, Aditya Parameswaran, and Madelon Hulsebos. 2024. Target: Benchmarking table retrieval for generative tasks. In *NeurIPS 2024 Third Table Representation Learning Workshop*.

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, Online. Association for Computational Linguistics.

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.

Sunjun Kweon, Yeonsu Kwon, Seonhee Cho, Yohan Jo, and Edward Choi. 2023. Open-wikitable: Dataset for open domain question answering with complex reasoning over table. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8285–8297.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Zheng Li, Yang Du, Mao Zheng, and Mingyang Song. 2025. MiMoTable: A multi-scale spreadsheet benchmark with meta operations for table reasoning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2548–2560, Abu Dhabi, UAE. Association for Computational Linguistics.

Weizhe Lin, Rexhina Blloshmi, Bill Byrne, Adria de Gispert, and Gonzalo Iglesias. 2023. LI-RAGE: Late interaction retrieval augmented generation with explicit signals for open-domain table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1557–1566, Toronto, Canada. Association for Computational Linguistics.

Niklas Muennighoff. 2022. Sgpt: Gpt sentence embeddings for semantic search. *arXiv preprint arXiv:2202.08904*.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, Dragomir Radev, and Dragomir Radev. 2022. FeTaQA: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Feifei Pan, Mustafa Canim, Michael Glass, Alfio Gliozzo, and Peter Fox. 2021. Cltr: An end-to-end, transformer-based system for cell-level table retrieval and table question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 202–209.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th*

International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1470–1480.

Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022a. Plaid: An efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 1747–1756, New York, NY, USA. Association for Computing Machinery.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022b. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, Seattle, United States. Association for Computational Linguistics.

Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, and 1 others. 2021. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2614–2627.

Jian Wu, Linyi Yang, Dongyuan Li, Yuliang Ji, Manabu Okumura, and Yue Zhang. 2025. MMQA: Evaluating LLMs with multi-table multi-hop complex questions. In *The Thirteenth International Conference on Learning Representations*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2024. TableLlama: Towards open large generalist models for tables. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6024–6044, Mexico City, Mexico. Association for Computational Linguistics.

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

A Prompt Details for Question Generation

We provide the exact prompts used in our LLM-based pipeline for extracting headers and generating simulated questions from partial tables. Prompts were designed to be highly structured and instructive, guiding the LLM (LLaMA-3.1-8B-Instruct) to handle inputs. All prompts return a strict JSON format, suitable for programmatic post-processing.

A.1 Header Extraction + Question Generation (Full Pipeline)

If header extraction is desired (used in most QGpT scenarios), we use the following prompt format:

You are an expert in table data analysis. Given a table with its file name, sheet name, and a portion of its content (first ten rows), your task is to **extract key headers and generate questions** based on the table & headers.

Important Considerations:

- The table may contain nan or Unnamed: values, which represent empty merged cells in the original table. These **should not** be considered as meaningful data points or headers.
- The **true column headers may not always be in the first row or first column**. Carefully analyze the table to identify the correct headers.
- If the table has **multi-level headers**, preserve the hierarchical structure without merging or altering the text.
- If the table has an **irregular header structure** (such as key-value formatted headers where column names are listed separately), extract the correct header names accordingly.
- **Ignore rows that contain mostly empty values (nan, Unnamed:) or placeholders without meaningful data.**
- **Do not generate python code, extract headers and questions on your own.**
- The type of Questions could be one of (lookup, calculate, visualize, reasoning).
- **Generate question using the language of the table.**

Tasks:

1. **Extract Header Names:**
   - Identify the **true headers** by analyzing the structure of the table.
   - **Exclude** placeholder values like "nan" and "Unnamed:".
   - If the table contains **multi-level headers**, keep them as separate levels without merging.
   - If the table has **key-value headers**, extract the correct column names.
2. **Generate Questions (Context-Specific to the Table):**
   - Formulate **questions that can only be answered using this specific table**.
   - Ensure **each question involves 1 to 3 different headers** to capture interactions between data & columns.
   - Ensure the header diversity in all the questions.
   - Use " to mark the headers in the question.
   - **Total number of questions should larger than the half number of extracted headers**

Output Format (Strictly JSON format)
Only return a JSON dictionary object with the extracted headers and questions, without any additional explanations or formatting.

{{ "headers": ["header1", "header2",

"..."], "questions": ["question1",

"question2", "..."] }}

Input Table: <table>

A.2 Question Generation Only (Without Header Extraction)

If header extraction is skipped (e.g., MMQA), we apply a simplified prompt:

You are an expert in table data analysis. Given a table with its file name and a portion of its content (first ten rows), your task is to **generate questions** based on the table & headers.

Important Considerations:

- **Do not generate python code, generate questions on your own.**
- The type of Questions could be one of (Numerical, List, Count, Select).
- **Generate question using the language of the table.**

Tasks:

- **1. Generate Questions (Context-Specific to the Table):**
- Formulate **questions that can only be answered using this specific table**.

- Ensure **each question involves 1 to 3 different headers** to capture interactions between data & columns.
- Ensure the header diversity in all the questions.
- Use ” to mark the headers in the question.
- **Total number of questions should larger than the half number of extracted headers**

****Output Format (Strictly JSON format)****
Only return a JSON dictionary object with the extracted headers and questions, without any additional explanations or formatting.

{ "questions": ["question1",

"question2","..."]

Input Table: <table>