Exploring the Impact of Table-to-Text Methods on Augmenting LLM-based Question Answering with Domain Hybrid Data

Dehai Min[*1,4] **Nan Hu**[*1,4] **Rihui Jin**[1,4] **Nuo Lin**[1] **Jiaoyan Chen**[2]
Yongrui Chen[1,4] **Yu Li**[1,4] **Guilin Qi**[1,4†] **Yun Li**[3] **Nijun Li**[3] **Qianren Wang**[3]

[1]School of Computer Science and Engineering, Southeast University, China
[2]Department of Computer Science, The University of Manchester, United Kingdom
[3]Advanced Cognitive AI Lab, Huawei Technologies, China
[4]Key Laboratory of New Generation Artificial Intelligence Technology and Its
Interdisciplinary Applications (Southeast University), Ministry of Education, China
{zhishanq, nanhu, gqi}@seu.edu.cn

Abstract

Augmenting Large Language Models (LLMs) for Question Answering (QA) with domain specific data has attracted wide attention. However, domain data often exists in a hybrid format, including text and semi-structured tables, posing challenges for the seamless integration of information. Table-to-Text Generation is a promising solution by facilitating the transformation of hybrid data into a uniformly text-formatted corpus. Although this technique has been widely studied by the NLP community, there is currently no comparative analysis on how corpora generated by different table-to-text methods affect the performance of QA systems. In this paper, we address this research gap in two steps. First, we innovatively integrate table-to-text generation into the framework of enhancing LLM-based QA systems with domain hybrid data. Then, we utilize this framework in real-world industrial data to conduct extensive experiments on two types of QA systems (DSFT and RAG frameworks) with four representative methods: Markdown format, Template serialization, TPLM-based method, and LLM-based method. Based on the experimental results, we draw some empirical findings and explore the underlying reasons behind the success of some methods. We hope the findings of this work will provide a valuable reference for the academic and industrial communities in developing robust QA systems.

1 Introduction

Enhancing the performance of Large Language Models (LLMs) in domain-specific Question Answering (QA) has been a focus of research, predominantly employing two key approaches (Ling et al., 2023; Wang et al., 2023a): Domain-Specific Fine-Tuning (DSFT) which involves training LLMs on the domain-specific corpus (Gururangan et al., 2020; Wu et al., 2023), and Retrieval-Augmented Generation (RAG) which utilizes a domain-specific corpus as an external knowledge base (Lewis et al., 2020b). These approaches, leveraging the inherent text processing strengths of LLMs, have been widely adopted in text-only scenarios, yielding significant improvements (Zhao et al., 2023a).

However, real-world data in many domains typically exists in a hybrid format, comprising not only text but also substantial volumes of semi-structured tables, as observed in e.g., scientific literature and medical reports (Chen et al., 2020c; Zhu et al., 2021). These tables frequently appear alongside text within the same document, providing semantically supplementary or complementary information crucial for a comprehensive understanding of the content (Chen et al., 2020a). In exploring the potential of leveraging hybrid data to enhance the performance of LLMs, it is crucial to effectively integrate these data, ensuring the coexistence of text and tables. The current methods for handling the heterogeneity of text and tables have significant drawbacks: 1) Directly flattening tables by concatenating cells row by row not only results in the loss of structural information embedded in the original table but also severs the informational links between cells (Sui et al., 2023; Xie et al., 2022). 2) Mapping text and tables to different vector spaces separately and then integrating them, not only increases complexity but also disrupts the semantic connection between the two types of data (Li et al., 2021; Huang et al., 2022).

One promising solution is table-to-text generation (Luo et al., 2022; Cheng et al., 2022), which aims to generate natural language statements that faithfully describe the information in the provided table. Through this, we can transform hybrid data into a unified natural language representation that is more suitable for use by LLMs, while also preserving the important information from the tables and the semantic connections between the data. Al-

---

* Equal Contributions.
† Corresponding author.

though table-to-text generation has been widely studied by the NLP community, there is currently no comparative analysis on how corpora generated by different table-to-text methods affect the performance of domain-specific QA systems.

In this work, we address this research gap by two steps. First, we innovatively integrate table-to-text generation into the framework of enhancing LLM-based QA systems with domain hybrid data. Then, we utilize this framework to conduct extensive experiments on two types of QA systems (DSFT and RAG paradigms) with four representative table-to-text methods. We choose the following four strategies: 1) **Markdown** format; 2) **Template** serialization; 3) **TPLM-based** method; 4) **LLM-based** method. These strategies differ in complexity and underlying technology. The Markdown and Template serialization offer simplicity, while the TPLM-based and LLM-based methods leverage the capabilities of advanced language models to generate more nuanced text.

In terms of implementation, we collect a real-world hybrid dataset called ICT-DATA, by extracting text and tables from numerous documents about Information and Communication Technology (ICT) products. It is important to note that the text contained in tables accounts for approximately 18% of the total content in ICT-DATA (based on word count statistics). We employ different table-to-text methods to process the tables in ICT-DATA, obtaining different ICT corpora. These corpora are then utilized to build QA systems. Moreover, we create a benchmark dataset named ICTQA, which consists of QA pairs based on the knowledge of ICT-DATA. This dataset is particularly suitable for evaluating enhanced LLMs, as it includes some industry-specific knowledge not covered in the general LLMs training stage.

To our knowledge, our research is the first to comprehensively compare different table-to-text strategies on LLM-based QA systems enhanced by domain hybrid data. Our main findings are as follows:

- Table-to-text methods significantly impact the performance of QA systems, with relative score differences ranging from 2.8% to 9.0% in human evaluation and 4.8% to 16% in GPT-4 evaluation. In two systems, selecting the appropriate method can yield considerable benefits.
- In the DSFT paradigm, LLM-based and TPLM-based consistently outperform others across various model settings, demonstrating their superiority. In the RAG paradigm, while the LLM-based method still performs excellently, the Markdown has shown unexpected effectiveness.
- The varying frequency of domain-specific terms and verbs produced by these methods, alongside the differing quality of semantic representations in the generated text chunks, which appear to be pivotal factors influencing performance disparities across the two systems.

2 Table-to-Text Generation

Table-to-text generation (Parikh et al., 2020; Chen et al., 2020b; Cheng et al., 2022) aims to create natural language descriptions from semi-structured tabular data, such as web tables. As shown in Figure 1, we apply four representative table-to-text methods to textualize the tables in ICT-DATA, forming four different corpora. Formally: Let F_i : Table $\rightarrow$ Text represent four table-to-text functions for $i = 1, 2, 3, 4$. With the original ICT-DATA $D = \{\text{Tab}, \text{Text}\}$, each F_i converts tables Tab into text. The resulting ICT Corpora C_i are formed by combining these texts with Text:

$$C_i = F_i(\text{Tab}) \cup \text{Text}, \quad i = 1, 2, 3, 4$$

We next provide a detailed introduction of these four methods. Table 1 provides a comparative analysis of these methods in terms of their resource requirements, processing speeds, and text diversity.

- **Markdown** format: A straightforward method to represent tables in Markdown format. It does not involve model training and can be rapidly processed via scripts without manual intervention.
- **Template** serialization: This method uses a set of templates designed based on table features for textualization (Li et al., 2023; Ye et al., 2019). It achieves slightly higher diversity in the generated text compared to the Markdown method, attributed to the use of multiple pre-prepared templates to accommodate different types of tables, which requires some manual involvement.
- **TPLM-based** method: This method involves fine-tuning Traditional Pre-trained Language Models (TPLMs), such as T5 (Raffel et al., 2020)

Method	Resource	Speed	Diversity
Markdown	CPU	Fast	Low
Template	CPU	Fast	Moderate
TPLM-based	GPU	Moderate	High
LLM-based	GPU or API	Low	Very High

Table 1: Comparison of table-to-text methods: resource usage, generation speed and diversity of generated text.
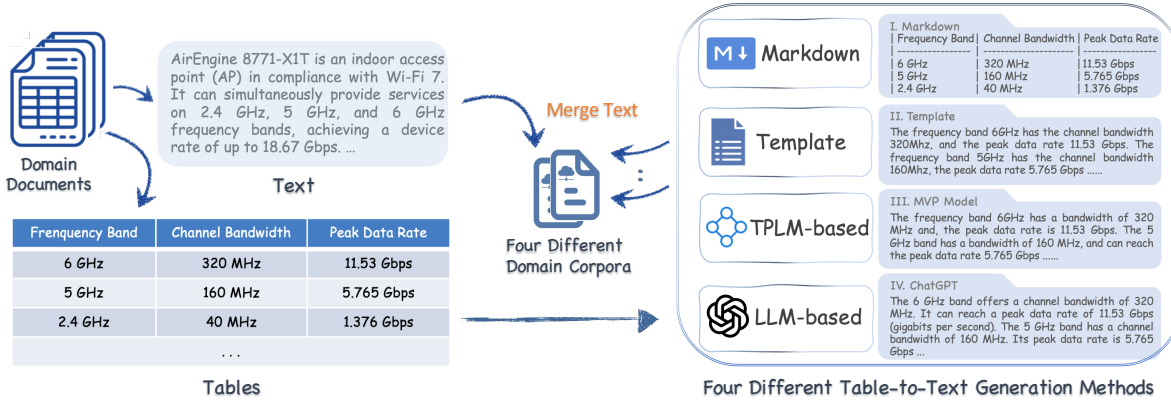
Figure 1: Illustration of four domain corpora generation process. Different table-to-text methods are applied to tables of domain documents, generating different text. These generated texts are then merged with the original document texts, yielding different domain corpora.

and BART (Lewis et al., 2020a), on specific table-to-text generation task datasets (Liu et al., 2022). In this paper, we utilize the MVP model (Tang et al., 2023), which initially pre-trains the BART model on numerous natural language generation datasets, followed by fine-tuning on various cross-domain table-to-text datasets. It allows customized adjustment of the output through fine-tuning, offering higher flexibility and domain adaptability, while requiring more computational resources.

• **LLM-based** method: Recent endeavors employing LLMs for this task have drawn significant attention (Bian et al., 2023). Impressively, Zhao et al. (2023b) demonstrate that GPT-* models often outperform the best-performing fine-tuned models. We refer to their findings and utilize ChatGPT in a one-shot setting in our work. Similar to TPLM-based methods, this approach can be custom-tailored using In-Context Learning. Moreover, using the APIs of certain proprietary LLMs might pose risks of domain data leakage.

Some examples of table-to-text, along with the specific templates and prompts for ChatGPT used in this paper, can be found in Appendix B.

3 Building LLM-based QA Systems with Domain Corpora

We will introduce separately how two LLM-based QA systems utilize these corpora. Their framework overview can be viewed in Figure 2.

Domain-Specific Fine-Tuning. In this approach, we first pre-train the LLM on the ICT corpus using next-token prediction (Radford et al., 2018), enabling the model to incrementally learn domain knowledge. Subsequently, we adapt the model to
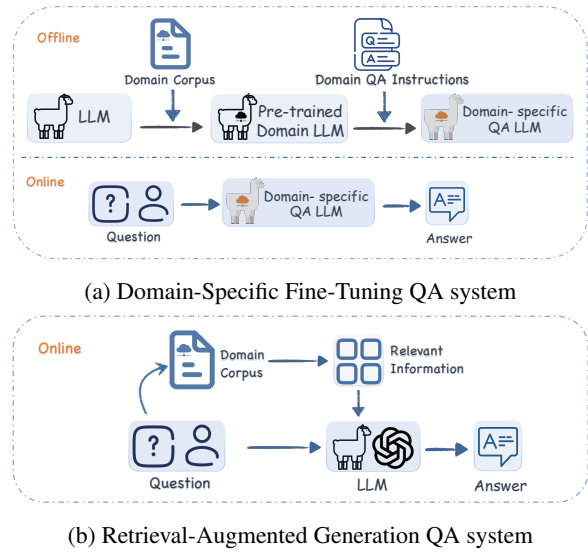

(a) Domain-Specific Fine-Tuning QA system

(b) Retrieval-Augmented Generation QA system

Figure 2: Framework of domain-enhanced QA systems.

the QA task through instruction tuning (Ouyang et al., 2022). Formally, an original LLM M, is pre-trained on each ICT Corpus C_i, to obtain an updated foundation model M'_i:

$$M'_i = \text{Pre-Train}(M, C_i), \quad i = 1, 2, 3, 4$$

The updated models are then further trained on the same instruction set I tailored for the QA task, resulting in the final QA oriented models M_i^{QA}:

$$M_i^{QA} = \text{FineTune}(M'_i, I), \quad i = 1, 2, 3, 4$$

Retrieval-Augmented Generation. In this paradigm, we adopt the framework proposed by LangChain (Chase, 2022) with the Dense Passage Retriever (DPR) method (Karpukhin et al., 2020), which consists of a multi-step process: 1) Splitting the large-sized Corpus C_i into smaller chunks $\{p_j\}^{C_i}$; 2) Encoding each text chunk p_j into a d-dimensional vector by an encoder $E_P(\cdot)$, which

captures its semantic essence; 3) Building an indexed Vector Store for these vectors, optimizing the storage for efficient retrieval; 4) For each query Q, retrieving the K most relevant text chunks, $P = \{p_k\}_{k=1}^{K}$; 5) Using both the query Q and the retrieved prompts P to generate the final answer with the LLM.

4 Dataset and Evaluation Metrics

4.1 Evaluation Dataset

ICT-DATA. We collect ICT-DATA based on 170 English technical documents related to ICT products. Each product document consists of tables and text, whose contents include product descriptions, configuration guides, terms, and definitions, etc. The total storage size is approximately 6GB. Moreover, the number of words in the table data accounts for about 18% of the total number of words in the dataset. In Appendix A.2, we provide detailed statistics and the preprocessing methods used for the table data.

ICTQA. We create the ICTQA dataset to evaluate the performance of domain QA systems, by collecting 9,000 questions with long-form answers from the actual ICT product technical support QA platform. All the answers are written by experts based on product documents. We manually select 500 questions as the test set, whose answers involve knowledge from both tables and text. The remaining QA pairs are used as the training set for the instruction fine-tuning phase in the DSFT paradigm. We show statistics and some examples in Appendix A.1.

4.2 Evaluation Metrics

To evaluate the model's responses, we employ both automated and manual evaluation methods.

Automated Evaluation Metrics. Given that traditional lexical-overlap-based metrics (such as BLEU and ROUGE) are inadequate for evaluating the quality of long-form answers generated by LLMs (Krishna et al., 2021; Kamalloo et al., 2023), we use GPT-4 as an evaluator with a demonstration setting, scoring responses based on their similarity to the golden answer (Liu et al., 2023). The score ranges from 0 to 5 with discrete values; 0 indicates incoherent answers with repeated fields or responses like "I don't know the answer", 1 represents minimal similarity to the golden answer, and 5 denotes an accurate answer.

Human Evaluation. Given the limitations in evaluating long-form answers using existing automated metrics (Wang et al., 2023b; Kamalloo et al., 2023), three evaluators with domain knowledge are asked to score responses based on the helpfulness and similarity to the golden answer, using the same scoring criteria with a range of 0 to 5 as the GPT-4 evaluator.

For fairness and to eliminate potential bias, responses are presented anonymously to both the GPT-4 and human evaluators. The full prompt, evaluation setup for human and scoring criteria are detailed in Appendix D.

5 Experimental Setup

QA Systems of the DSFT Paradigm. Within the DSFT paradigm, we utilize Meta's OPT (1.3B to 13B) (Zhang et al., 2022) and Llama2-base (7B, 13B) (Touvron et al., 2023) as foundation models. The OPT models offer variable sizes to enhance robustness. To mitigate training costs, we employ the QLoRA (Dettmers et al., 2023) strategy for pre-training and instruction fine-tuning. The instruction template can be found in Appendix A.3.

QA Systems of the RAG Paradigm. We use the Llama2-chat models (7B, 13B, and 70B) and GPT-3.5-turbo for inference. We divide the corpus into smaller chunks, ensuring the integrity of sentences and keeping their lengths below 3000 characters. Subsequently, text chunks are vectorized using the BGE embedding model (Zhang et al., 2023). We utilize the FAISS library (Johnson et al., 2021) to retrieve the vectors of the top-3 relevant text chunks based on similarity. These chunks are input to the LLM with the corresponding questions for answering through the RAG-Chain from LangChain (Chase, 2022).

Fair Comparison. To maintain consistency and control variables, all models are trained or used under the same settings on four different corpora. Detailed training parameters and GPU costs are available in Appendix C.

6 Results

In the following subsections, we will discuss three research questions regarding our study.

6.1 RQ1: How do these methods affect the performance of QA systems?

Table 2 shows the average scores for different QA system setups on the ICTQA test set. We can see

Metrics	Table-to-Text Method	Domain-Specific Fine-Tuning						Retrieval-Augmented Generation			
		OPT-1.3B	OPT-2.7B	OPT-6.7B	OPT-13B	Llama2-7B	Llama2-13B	GPT-3.5-turbo	Llama2-7B	Llama2-13B	Llama2-70B
Human Eval.	Markdown	2.05	2.41	2.38	2.51	2.82	3.05	3.29	**3.72**	<u>3.98</u>	<u>3.94</u>
	Template	2.04	2.40	2.26	2.47	2.82	3.04	<u>3.36</u>	3.44	3.96	3.76
	TPLM-based	<u>2.12</u>	<u>2.43</u>	<u>2.43</u>	<u>2.58</u>	**3.20**	<u>3.13</u>	3.26	3.27	3.92	3.64
	LLM-based	**2.18**	**2.57**	**2.51**	**2.62**	<u>2.96</u>	**3.19**	**3.62**	<u>3.71</u>	**4.26**	**4.09**
	RSD(%)	2.80	3.40	5.00	3.00	7.60	3.00	7.20	9.00	6.80	9.00
GPT-4 Eval.	Markdown	1.74	2.16	2.27	2.25	2.7	3.06	<u>3.28</u>	**3.66**	<u>3.67</u>	**3.74**
	Template	1.81	2.22	2.39	2.34	2.84	3.08	3.27	3.06	3.38	3.37
	TPLM-based	<u>2.33</u>	<u>2.46</u>	<u>2.45</u>	<u>2.53</u>	**3.20**	<u>3.19</u>	<u>3.28</u>	2.9	3.41	3.30
	LLM-based	**2.57**	**2.69**	**2.73**	**2.86**	<u>3.06</u>	**3.30**	**3.64**	<u>3.59</u>	**3.69**	<u>3.54</u>
	RSD(%)	16.60	10.60	9.20	12.20	10.00	4.80	7.40	15.20	6.20	8.80

Table 2: The average scores from Human Evaluation and GPT-4 Evaluation of the QA systems with four representative table-to-text methods. In each setting, the best result is shown in bold, and the second-best result is underlined. Relative Score Difference (RSD) is calculated using the formula (Highest Score − Lowest Score)/5.

that there are significant differences in the performance of the two types of QA systems enhanced by corpora generated from different table-to-text methods. Their Relative Score Differences range from 2.8% to 9.0% in human evaluation and from 4.8% to 16% in GPT4 evaluation. For a more detailed observation, we present the score distribution from human evaluation of the DSFT QA models based on OPT-6.7B in Figure 3. From this figure, we can observe significant differences in score distribution among different QA models, reflecting their performance variations. From Table 2, we note that in the DSFT paradigm, both TPLM-based and LLM-based methods, which utilize language models for table-to-text generation, perform well across different models. Particularly, the LLM-based method shows the best performance in many models. On the other hand, the RAG paradigm provides a different observation. While the LLM-based method continues to exhibit excellent performance, the Markdown format shows a significant and unexpected improved performance in the RAG paradigm compared to DSFT, even best-performing in some models. To further illustrate these findings, we show the competition results of some QA system scores in Figure 4. We can clearly observe that the methods with higher average scores also have a higher probability of achieving better scores for each question. These observations underscore the necessity of choosing the appropriate method for processing table data when building domain-specific QA systems.

6.2 RQ2: What are the potential reasons for their different performances?

Since DSFT and RAG systems utilize domain corpora in different ways, we will discuss them separately in this section.
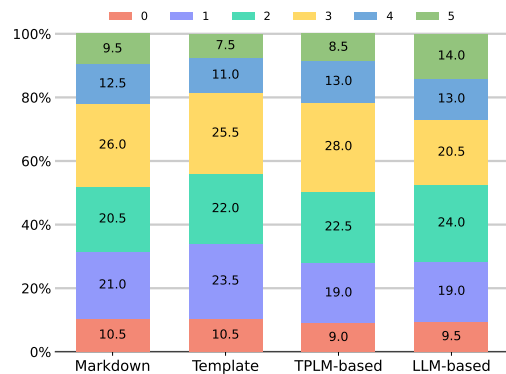
For the DSFT paradigm. Inspired by the findings

Figure 3: The scores distribution from human evaluation for the DSFT QA systems based on OPT-6.7B.

Freq (k)	C_1· Markdown	C_2· Template	C_3· TPLM-based	C_4· LLM-based
Term	821	1040	2358	2254
Verbs	313	315	682	1207

Table 3: Absolute frequency of verbs and terms contained in the corpora C_i generated by different methods.

of (Biderman et al., 2023; Razeghi et al., 2022; Elazar et al., 2023), which suggest a correlation and causal relationship between the ability of LLMs to answer factual questions and the frequency of salient entities found in their pre-training corpora, we also observe that different table-to-text methods have inconsistent preferences for domain verbs when describing tables. Following the approach of (Zevallos et al., 2023; Wang et al., 2023c), we extract domain term sets and related verb sets from the QA pairs in the ICTQA test set. We then calculate the absolute frequency of these terms and verbs as they appear in the corpora generated by different table-to-text methods. In Table 3, we can clearly see significant differences in these frequencies across different corpora. For example, LLM-based methods show a term frequency more than twice that of Template methods, with verb frequency quadrupling. This is because LLM-based methods tend to supplement the subject with the
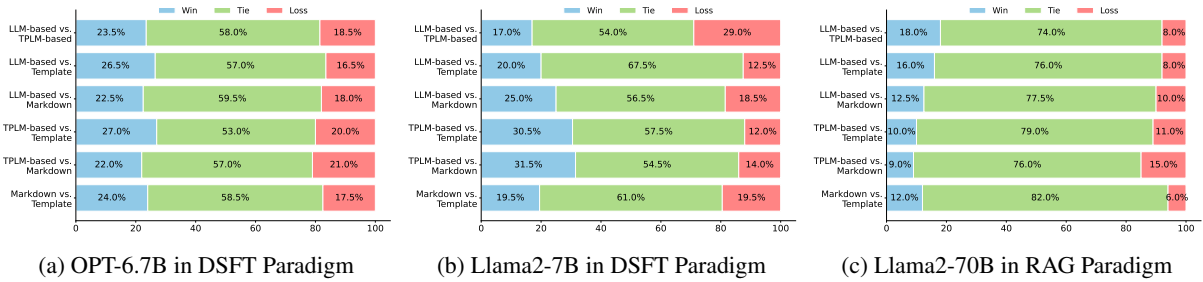
| | (a) OPT-6.7B in DSFT Paradigm | (b) Llama2-7B in DSFT Paradigm | (c) Llama2-70B in RAG Paradigm |

Figure 4: Comparison of human evaluation scores between QA models using different Table-to-Text methods. 'A vs. B win' indicates the percentage of test set instances where Model A's score surpasses Model B's.

domain entity corresponding to the attribute when describing tables, and exhibits greater diversity in verbs. In contrast, Template methods use more pronouns, such as 'it', and monotonous predicates (usually 'be' verbs). By comparing these frequency rankings with the performance shown in Table 2, we can observe a positive correlation between them: methods with higher frequencies, especially the TPLM and LLM-based methods, correspond to superior QA capabilities in the DSFT systems.

For the RAG paradigm. Under the same LLM reader setup, retrieval accuracy in this semantic space crucially impacts RAG performance (Ma et al., 2023). The retrieval process involves selecting the vectorized chunks with the highest similarity scores to the query vector. To investigate the impact of different methods on retrieval effectiveness, we use t-SNE (Van der Maaten and Hinton, 2008) to visualize the clustering of a query and related chunks in the semantic space at Figure 5. It could be clearly seen that chunks generated by the LLM-based and Markdown methods, which perform well in Table 2, are closer to the query in the semantic space. This makes the chunks related to the query more likely to be retrieved, thereby improving the system's performance. This suggests that in the RAG framework with the DPR method, the texts generated by these methods have more retrieval-friendly semantic representations and better alignment between queries and documents.

Freq (Avg.)	Markdown	Template	TPLM-based	LLM-based
Text Len	998	1259	1138	897

Table 4: The average length of text generated by different methods for each table.

6.3 RQ3: Are there practical suggestions for choosing table-to-text methods?

Through the analysis of RQ1 and RQ2, we know that the LLM-based strategy with ChatGPT is outstanding and reliable in both frameworks. In case
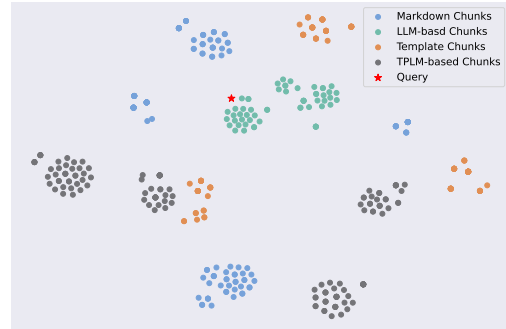

Figure 5: A t-SNE visualization of chunk clusters in the embedding space of the RAG system. 'X Chunks' represents chunks related to the query (red star) from the corpus generated by X table-to-text method.

its drawbacks mentioned in Section 2 are unacceptable, the TPLM-based strategy (i.e., selecting a well-tuned table-to-text model) is a good alternative in the DSFT paradigm. In the RAG paradigm, the simple and easy-to-use Markdown strategy is also a viable substitute. Additionally, although RAG systems using these four methods significantly outperform DSFT systems in terms of performance, building a vector retrieval library demands substantial memory resources. Therefore, referring to Table 4, choosing methods that generate more concise texts, such as LLM-based and Markdown strategies, is a wise decision.

6.4 Additional discussion on experimental results

As shown in Table 2, under the ICT dataset and the experimental setup of this study, the RAG method outperforms the DSFT method in Llama2 models. This demonstrates that RAG has an excellent performance as a lower cost method. We attribute this result to two main reasons: 1). The ICT data used in this study covers dense domain knowledge, and it is still challenging to adapt the LLM well to this complex domain data through incremental pre-training. 2). As the statistical analysis in Ap-

pendix A.1, most of the questions in the ICTQA are quizzes on the knowledge of product manuals. In this scenario, the existing excellent dense vector retrievers have high recall accuracy. The studies of (Gupta et al., 2024) and (Soudani et al., 2024) have respectively conducted detailed experiments on the choice between Fine-Tuning and RAG under the agricultural domain data and Less Popular Knowledge scenarios. Our experimental results in this work further validate their viewpoints. It is also worth noting that in this study, the bge-large-en embedding model (Zhang et al., 2023) embeds text chunks into 1024-dimensional vectors. During the retrieval of relevant chunks based on the questions, the peak running memory requirement is approximately 280G.

Another interesting experimental result is that GPT-3.5-turbo performs worse than the Llama2 family in the RAG paradigm. We manually observe the QA cases and find that GPT-3.5-turbo has a significantly higher probability of outputting "I don't know the answer.", even if the retriever finds text chunks containing the correct answer.

7 Related Work

7.1 Domain Augmented Large Language Models.

In order to enhance the capabilities of LLMs in domain-specific tasks, some works develop LLMs through incremental training on an extensive domain corpus, inheriting the benefits of both the emergent abilities of LLMs and domain-specific knowledge (Luo et al., 2023; Huang et al., 2023). This technology yields significant results, but it demands substantial computational resources and incurs high costs (Wang et al., 2023a). In order to overcome this difficulty, a prompt-based solution that does not require updating model parameters has been proposed. They retrieve relevant domain information from external knowledge bases before answering questions with LLMs (Gao et al., 2023; Wang et al., 2023d; Xu et al., 2023).

7.2 Question Answering over Hybrid Data

Some works study QA tasks on hybrid data that contain both tables and text (Zhu et al., 2021; Chen et al., 2020c,a). Popular approaches often involve designing a complex system that has independent modules to process text and tables separately. The information from these two modules is then merged and fed into a language model to generate answers (Zhong et al., 2022). Additionally, some of these methods not only require annotations of metadata identifying text and tables relevant to the question, but they also rely on the formulation of executable languages to access tables, such as SQL or SPARQL (Nan et al., 2022; Li et al., 2021). These executable languages often have strict assumptions about the structure of the tables. These limitations make these approaches ill-suited for the real-world LLM-based scenario domain QA systems. Therefore, the results of this study were not compared with these baseline models in the experiments.

8 Conclusion

This paper studies the impact of different table-to-text methods on LLM-based QA systems enhanced by domain hybrid data. Specifically, we meticulously compared four representative methods: Markdown formatting, Template serialization, TPLM-based, and LLM-based approaches. Through experiments, we show the superiority of the LLM-based and TPLM-based methods in the DSFT framework, and the excellence of the LLM-based and Markdown methods in the RAG framework. A key discovery is the varying frequency of domain-specific terms and verbs produced by these methods, alongside the differing quality of semantic representations in the generated text chunks, which appear to be pivotal factors influencing performance disparities across the two systems. These insights not only shed light on the nuances of table-to-text generation methods but also have profound implications for the enhancement of LLMs. Furthermore, they offer practical guidance for tailoring domain-specific QA systems to meet particular needs.

Acknowledgements

References

Junyi Bian, Xiaolei Qin, Wuhe Zou, Mengzuo Huang, and Weidong Zhang. 2023. Hellama: Llama-based table to text generation by highlighting the important evidence.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.

Harrison Chase. 2022. Langchain.

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, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020b. Logical Natural Language Generation from Open-Domain Tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online. Association for Computational Linguistics.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020c. 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.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland. Association for Computational Linguistics.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. ArXiv:2305.14314 [cs].

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Amir Feder, Abhilasha Ravichander, Marius Mosbach, Yonatan Belinkov, Hinrich Schütze, and Yoav Goldberg. 2023. Measuring Causal Effects of Data Statistics on Language Model's 'Factual' Predictions. ArXiv:2207.14251 [cs].

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey.

Aman Gupta, Anup Shirgaonkar, Angels de Luis Balaguer, Bruno Silva, Daniel Holstein, Dawei Li, Jennifer Marsman, Leonardo O Nunes, Mahsa Rouzbahman, Morris Sharp, et al. 2024. Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture. *arXiv preprint arXiv:2401.08406*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Huang Huang, Fei Yu, Jianqing Zhu, Xuening Sun, Hao Cheng, Dingjie Song, Zhihong Chen, Abdulmohsen Alharthi, Bang An, Juncai He, Ziche Liu, Zhiyi Zhang, Junying Chen, Jianquan Li, Benyou Wang, Lian Zhang, Ruoyu Sun, Xiang Wan, Haizhou Li, and Jinchao Xu. 2023. Acegpt, localizing large language models in arabic.

Junjie Huang, Wanjun Zhong, Qian Liu, Ming Gong, Daxin Jiang, and Nan Duan. 2022. Mixed-modality Representation Learning and Pre-training for Joint Table-and-Text Retrieval in OpenQA. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4117–4129, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5591–5606, Toronto, Canada. Association for Computational Linguistics.

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.

Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. 2021. Hurdles to Progress in Long-form Question Answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4940–4957, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a.

BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Alexander Hanbo Li, Patrick Ng, Peng Xu, Henghui Zhu, Zhiguo Wang, and Bing Xiang. 2021. Dual reader-parser on hybrid textual and tabular evidence for open domain 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 (Volume 1: Long Papers)*, pages 4078–4088.

Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2023. Table-gpt: Table-tuned gpt for diverse table tasks. *arXiv preprint arXiv:2310.09263*.

Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Xuchao Zhang, Tianjiao Zhao, Amit Panalkar, Wei Cheng, Haoyu Wang, Yanchi Liu, Zhengzhang Chen, Haifeng Chen, Chris White, Quanquan Gu, Jian Pei, and Liang Zhao. 2023. Domain Specialization as the Key to Make Large Language Models Disruptive: A Comprehensive Survey. ArXiv:2305.18703 [cs].

Ao Liu, Haoyu Dong, Naoaki Okazaki, Shi Han, and Dongmei Zhang. 2022. PLOG: Table-to-logic pre-training for logical table-to-text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5531–5546, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Yizhen Luo, Jiahuan Zhang, Siqi Fan, Kai Yang, Yushuai Wu, Mu Qiao, and Zaiqing Nie. 2023. BioMedGPT: Open Multimodal Generative Pre-trained Transformer for BioMedicine. ArXiv:2308.09442 [cs].

Yutao Luo, Menghua Lu, Gongshen Liu, and Shilin Wang. 2022. Few-shot Table-to-text Generation with

Prefix-Controlled Generator. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6493–6504, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315, Singapore. Association for Computational Linguistics.

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.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

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.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th*

ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 3505–3506.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. Impact of pretraining term frequencies on few-shot numerical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 840–854, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Heydar Soudani, Evangelos Kanoulas, and Faegheh Hasibi. 2024. Fine tuning vs. retrieval augmented generation for less popular knowledge.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2023. GPT4Table: Can Large Language Models Understand Structured Table Data? A Benchmark and Empirical Study. ArXiv:2305.13062 [cs] version: 3.

Tianyi Tang, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. 2023. MVP: Multi-task Supervised Pretraining for Natural Language Generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8758–8794, Toronto, Canada. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, Yidong Wang, Linyi Yang, Jindong Wang, Xing Xie, Zheng Zhang, and Yue Zhang. 2023a. Survey on Factuality in Large Language Models: Knowledge, Retrieval and Domain-Specificity. ArXiv:2310.07521 [cs].

Jiaan Wang, Yunlong Liang, Fandong Meng, Zengkui Sun, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023b. Is ChatGPT a good NLG evaluator? a preliminary study. In *Proceedings of the 4th New Frontiers in Summarization Workshop*, pages 1–11, Singapore. Association for Computational Linguistics.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023c. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Yubo Wang, Xueguang Ma, and Wenhu Chen. 2023d. Augmenting black-box llms with medical textbooks for clinical question answering.

Chaoyi Wu, Weixiong Lin, Xiaoman Zhang, Ya Zhang, Yanfeng Wang, and Weidi Xie. 2023. PMC-LLaMA: Towards Building Open-source Language Models for Medicine. ArXiv:2304.14454 [cs].

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 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, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Benfeng Xu, Chunxu Zhao, Wenbin Jiang, PengFei Zhu, Songtai Dai, Chao Pang, Zhuo Sun, Shuohuan Wang, and Yu Sun. 2023. Retrieval-augmented domain adaptation of language models. In *Proceedings of the 8th Workshop on Representation Learning for NLP (RepL4NLP 2023)*, pages 54–64, Toronto, Canada. Association for Computational Linguistics.

Fangkai Yang, Pu Zhao, Zezhong Wang, Lu Wang, Bo Qiao, Jue Zhang, Mohit Garg, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. 2023. Empower large language model to perform better on industrial domain-specific question answering. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 294–312, Singapore. Association for Computational Linguistics.

Rong Ye, Wenxian Shi, Hao Zhou, Zhongyu Wei, and Lei Li. 2019. Variational template machine for data-to-text generation. In *International Conference on Learning Representations*.

Rodolfo Zevallos, Mireia Farrús, and Núria Bel. 2023. Frequency Balanced Datasets Lead to Better Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7859–7872, Singapore. Association for Computational Linguistics.

Peitian Zhang, Shitao Xiao, Zheng Liu, Zhicheng Dou, and Jian-Yun Nie. 2023. Retrieve Anything To Augment Large Language Models. ArXiv:2310.07554 [cs].

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open Pre-trained Transformer Language Models. ArXiv:2205.01068 [cs].

Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Li Yun,

Hejie Cui, Zhang Xuchao, Tianjiao Zhao, et al. 2023a. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *arXiv preprint arXiv:2305.18703*.

Yilun Zhao, Haowei Zhang, Shengyun Si, Linyong Nan, Xiangru Tang, and Arman Cohan. 2023b. Investigating table-to-text generation capabilities of large language models in real-world information seeking scenarios. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 160–175, Singapore. Association for Computational Linguistics.

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 question answering. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4531–4537.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3277–3287.

A ICT Datasets

A.1 ICTQA

To analyze various question types, we follow the classification method of Yang et al. (2023). This approach categorizes questions based on their first interrogative word and assigns tags reflecting the nature of the information sought. The statistical data of this classification is detailed in Table 5. Specifically, the ICTQA dataset labels questions under tags like 'Parameter', 'Configuration', and 'Command', each indicating the type of information requested. For instance, 'Parameter' relates to queries about specific values or settings, while 'Configuration' pertains to questions regarding the setup of systems or processes. Additionally, questions are grouped by their first interrogative word. This categorization sheds light on user inquiries: 'What' typically seeks factual details, and 'How' focuses on procedures or techniques. The average length of questions and answers in ICTQA is 75.13 characters and 160.25 characters, respectively. Table 6 shows examples from ICTQA questions.

Question Tag (%)		1st Question word (%)	
Parameter	19.55	What	29.33
Configuration	17.94	How	16.84
Command	12.25	Why	11.6
Other	50.26	Which	9.14
Avg # of length		Can	6.57
Question	75.13	Is	4.24
Answer	160.25	Other	22.28

Table 5: Statistics of ICTQA

A.2 ICT-DATA

In the process of collecting the ICT-DATA, we perform preprocessing on the table data. Specifically,
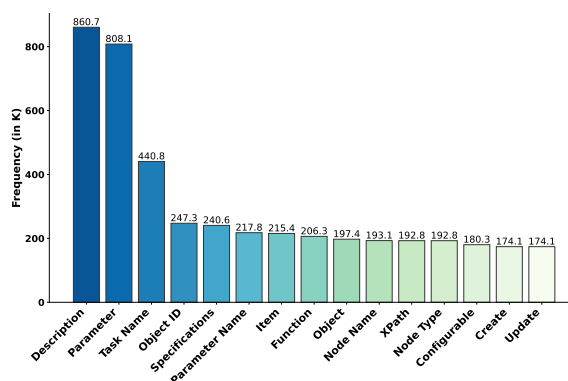

Figure 6: Top 15 Frequent Cell Contents in the Header Row of Tables.

> Question 1:
> What is the range for the "number" parameter when configuring the maximum number of routes supported by the VPN instance IPv4 address family on the USG9500?
>
> Answer 1:
> The value for the "number" parameter can range from 1 to 500,000 on the USG9500.
>
> ---
>
> Question 2:
> Can NetStream sampling be enabled on the ingress or transit node for traffic over Segment Routing tunnels?
>
> Answer 2:
> Sampling on the ingress or transit node is not supported.
>
> ---
>
> Question 3:
> How can I monitor the connectivity between a MEP and an RMEP or between a MEP and a MIP on other devices using 802.1ag MAC ping?
>
> Answer 3:
> Run ping mac-8021ag mep mep-id mep-id [md md-name ma ma-name] mac mac-address | remote-mep mep-id mep-id [-c count | -s packetsize | -t timeout | -p priority-value].

Table 6: Three examples from the ICTQA dataset.

Statistics	Value
Total Word Count in ICT-DATA	987M
Total Word Count in ICT Tables	178M
Average Word Count per Table	476.71
Average Number of Cells per Table	13.29
Average Text Length per Cell	35.87

Table 7: ICT-DATA Statistical Overview

to standardize the tables from the dataset, we transform them into N x M arrays. For tables with merged cells, we expand the col-span or row-span attributes, copying the content into individual cells. Additionally, to illustrate the characteristics of tables in the ICT domain, Figure 6 shows the top 15 frequent cell contents in the header rows of all tables in the ICT-DATA dataset. Table 7 provides a detailed statistical overview of the ICT-DATA. The total number of words in the dataset reaches 987 million, of which there are 178 million words in the tables, accounting for about 18% of the total dataset. On average, each table contains about 477 words, about 13 cells, and the average text length of each cell is about 36 words.

A.3 Instruction Template

Table 8 shows the instruction template we use. We fill the question and answer slots in the template with the QA pairs from the ICTQA dataset to form a set of instructions.

B Table-to-Text Generation Setups

B.1 Template Design for Table Serialization

The tables in the ICT-DATA dataset consist of two types: relational tables and key-value pair tables. These two types of tables can be easily distinguished by matching keywords in the header row cells and considering the number of columns in the table. As illustrated in Table 9, we develop distinct templates for each type of table: 1) Key-value pair tables, as shown in Table 15, contain m key-value pairs that describe entities mentioned in the table's title. 2) Relational tables, as shown in Table 16, include a main column (MC) and n attribute columns (AC), where the cells in the main column represent the entities described by the table. The main column can be identified through simple rules, including the uniqueness of its content and the presence of specific keywords in the header row. To enhance the diversity of the text produced through the Template serialization, we compile a specialized glossary. When a term from this glossary is found in a table's header, the corresponding template content is adjusted accordingly. For example, if the string in the main column is "Name", "The $[AC_1]$ of the $[MC]$ named $[C_{M_1}]$ is $[C_{M_1A_1}]$." will be changed as "The $[AC_1]$ of the $[C_{M_1}]$ is $[C_{M_1A_1}]$."

B.2 Prompt for LLM-based Method

In Table 10, we present a prompt template specifically designed for the LLM-based table-to-text method. This template is tailored for generating natural language descriptions from tables in a two-dimensional array format, and includes a demonstration. In the prompt, we instruct LLMs not to

> Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.
> **Instruction:** Please answer the following questions concerning ICT products.
> **Input:** {Question}
> **Response:** {Answer}

Table 8: The instruction template.

<table>
<tr><td colspan="2">

Template 1 for relational tables

Table Title

MC	AC_1	AC_2	...	AC_n
C_{M_1}	$C_{M_1 A_1}$	$C_{M_1 A_2}$	...	$C_{M_1 A_n}$
...				
C_{M_m}	$C_{M_m A_1}$	$C_{M_m A_2}$	...	$C_{M_m A_n}$

Generated Text:
The following sentences describe [Table Title]. The $[AC_1]$ of the $[MC]$ named $[C_{M_1}]$ is $[C_{M_1 A_1}]$. Its $[AC_2]$ is $[C_{M_1 A_2}]$. ... Its $[AC_n]$ is $[C_{M_1 A_{n-1}}]$. ... The $[AC_1]$ of the $[MC]$ named $[C_{M_m}]$ is $[C_{M_m A_1}]$. ... Its $[AC_n]$ is $[C_{M_m A_{n-1}}]$.

Template 2 for key-value pair tables

Table Title

Header 1	Header 2
K_1	V_1
K_2	V_2
...	
K_m	V_m

Generated Text:
The following sentences describe [Table Title]. The $[K_1]$ of $[C_P]$ is $[V_1]$. Its $[K_2]$ is $[V_2]$. ... Its $[K_m]$ is $[V_m]$.

</td></tr>
</table>

Table 9: Two Templates in the Template Method.

Now you have a task to complete. Task description: You will be given a table (with the 2d array format with the Caption). You need to generate a natural language description of the contents of the table. You can only generate content from the table content, do not generate other related or unrelated content. Here is an examples.

Table: Caption: Parameters for the ip link add name and ip link del dev.
[['Parameter', 'Description', 'Value'], ['name NAME', 'Specifies the name of a bridge.', 'The value is a string of 1 to 15 case-sensitive characters without spaces.'], ['dev DEV', 'Specifies the name of a bridge.', 'The value is a string of 1 to 15 case-sensitive characters without spaces.'], ['type bridge', 'Indicates that the device type is bridge.', '-']].

Description: The table provides details on the parameters for the ip link add name and ip link del dev commands. There are different parameters for configuring a bridge. The "name NAME" parameter is for specifying the name of a bridge and accepts a string with 1 to 15 case-sensitive characters, excluding spaces. The "dev DEV" parameter also specifies the name of a bridge and requires a string of 1 to 15 case-sensitive characters without spaces. The "type bridge" parameter indicates that the device being configured is of the type 'bridge.' It does not require a specific value.

Table: {Table}
Description:

Table 10: The prompt template designed for the LLM-based table-to-text generation.

---

output any additional information (which does not appear in the table, but comes from the internal knowledge of the LLM), regardless of whether this information is relevant to the table content.

B.3 Table-to-Text Generation Examples

In Table 15, we showcase the conversion of a simple two-column table into text using four different methods. For a more complex scenario involving a multi-column table with empty cells, refer to the example provided in Table 16. Lastly, the adaptation of these methods for a table featuring both multiple columns and merged cells is displayed in Table 17.

C Training Setup and GPU costs

This paper involves model training within the DSFT QA framework. We utilize an A100 40GB node equipped with 4 GPUs for both pre-training and fine-tuning of the Llama2-13B model. The pre-training and fine-tuning phases for other models are performed on a V100 32GB node with 8 GPUs. These processes leverage the DeepSpeed framework (Rasley et al., 2020). Table 11 provides a detailed overview of the GPU costs for training each model under the markdown method setting.

During the pre-training stage, we adopt an unsupervised learning approach, focusing on next-token prediction. To optimize memory usage, the DeepSpeed Zero Redundancy Optimizer (Zero Stage 2) is employed (Rajbhandari et al., 2020). Training parameters include a per-GPU batch size of 16 and a fixed seed value (1234) to ensure reproducibility. Each model undergoes a single epoch of training, utilizing a cosine learning rate scheduler with an initial rate of 2e-4. The learning rate warmup ratio is set at 0.05, accompanied by a weight decay rate of 0.01. Data preprocessing is performed with a block size of 512. In the QLoRA configuration, trainable parameters include the transformer's query, key, value, and output projection matrices, along with token embeddings and the language model head. Other parameters include: a LoRA rank of 64, an alpha value of 128, a dropout rate of 0.05 for the LoRA layer, and float16 for PyTorch (Paszke et al., 2019) tensors.

For the instruction fine-tuning phase, we derive the instruction dataset from the ICTQA training

Model	Pre-training Cost (GPU hours)	Fine-Tuning Cost (GPU hours)	Total Cost (GPU hours)	GPU Node
OPT-1.3B	102	1	103	8*V100-32G
OPT-2.7B	176	2	178	8*V100-32G
OPT-6.7B	339	5	344	8*V100-32G
OPT-13B	660	5	665	8*V100-32G
Llama2-7B	519	5	524	8*V100-32G
Llama2-13B	549	7	556	4*A100-40G

Table 11: The GPU cost of training each model under the QLoRA strategy using a corpus generated by the **Markdown** table-to-text generation method. The training cost with corpora produced by other methods is close to that of Markdown. The GPU hours are computed as follows: (iteration time (seconds) $\times$ number of iterations $\times$ number of GPUs $\div$ 3600 seconds/hour).

set. The training parameters for this phase are: 5 training epochs, a maximum token length of 512, a batch size of 8 per GPU, and a learning rate of 1e-4 with a cosine decaying scheduler. The configuration of QLoRA remains the same as in the pre-training phase.

D Evaluation Setup

D.1 Prompt for LLM as Evaluator

Table 12 details the prompt template used for evaluating responses via LLM (GPT-4) in a one-shot setting. The task of the GPT-4 is to compare four responses to a grounded answer and assigning each a score ranging from 0 to 5. Additionally, Table 13 provides an example of the one-shot demonstration used in this evaluation.

D.2 Setup and Criteria for Human Evaluation

For the human evaluation, three co-authors of this paper, all with domain expertise in ICT products, are designated as evaluators. Each sample receives three independent evaluations from these qualified evaluators. We analyze the consistency of scoring across the three evaluations. If the ranking order of the four responses remains consistent and the score difference for the same response across different evaluators does not exceed one point, the evaluation is deemed reliable. In cases of significant discrepancies, evaluators are requested to reassess the sample. In all evaluation documents, the sources of the responses are anonymously presented as 'A','B','C', and 'D'. Table 14 shows the scoring criteria for human evaluation, which is consistent with the scoring criteria for LLM as evaluator presented in Table 12.

E Case Analysis

We demonstrate a QA case of a RAG QA system built upon the LLaMA-70B-chat model. As shown

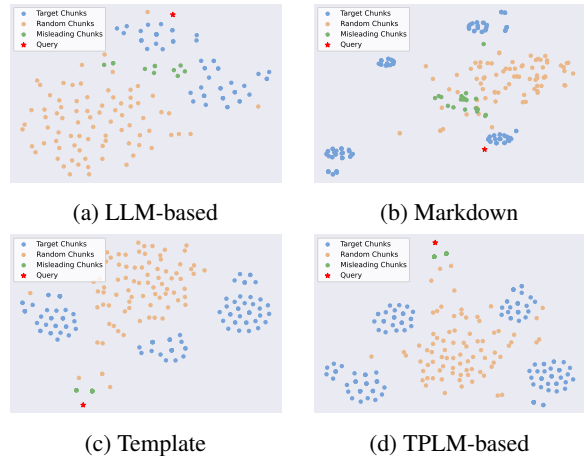(a) LLM-based   (b) Markdown

(c) Template   (d) TPLM-based

Figure 7: T-SNE visualization of chunk clusters in the embedding space for the four table-to-text methods in the RAG system case study. 'Random Chunks' represent chunks randomly selected from the corpus.

in Table 18, our RAG QA system successfully retrieves the correct context information containing the query answer from the corpora generated by the Markdown and LLM-based methods. However, it fails to retrieve correct information from the corpora generated by the TPLM-based and Template methods. We show their t-SNE visualization in the semantic space in Figure 7. In this case, it can be clearly seen that the misleading text chunks generated by the TPLM-based and template method, which are related to entities in the query but do not contain the correct answer, are semantically closer to the query. This leads to the failure to retrieve the correct chunks (i.e., Target chunks in the figure) containing the query's answer, indicating that the text generated by these two methods has poor semantic representations.

[System]
You will be provided with a question, the correct answer to the question, and four candidate answers. Your responsibility is to evaluate the consistency between the candidate answers and the correct answer. The focus should be on understanding the correlation or similarity of the content, rather than grammar or style. Please make sure you understand these guidelines before proceeding.

Consult this guide whenever needed:
0, penalty:
The candidate answers have issues such as repetitive sentences, which can significantly impair the helpfulness of the response.
1, Very low correlation:
Indicates that the candidate answer is almost entirely unrelated or opposite to the correct answer.
2, Low correlation:
Indicates that the candidate answer significantly deviates from the correct answer.
3, Moderate correlation:
Suggests that the candidate answer shares some similarities with the correct answer but may lack several key points or include extra unrelated content.
4, High correlation:
Indicates that the candidate answer is largely consistent with the correct answer, missing only minor points or details.
5, Very high correlation:
Signifies that the candidate answer is almost identical to or captures the complete essence of the correct answer.

You will need to categorize the four candidate answers A, B, C, and D based on their relevance.
For example, A:1, B:2, C:3, D:5 means: A's correlation with the correct answer falls into 1. Very low correlation. B's correlation with the correct answer is higher than A's, at 2. Low correlation. C's correlation with the correct answer is 3. Moderate correlation. D's correlation with the correct answer is 5. Very high correlation.

<A demonstration of the evaluation>

[User]
Evaluation Form (only score, do not output any other explanation):
<Question>: question
<Correct answer> (Standard Answer): golden answer
A answer: response_1
B answer: response_2
C answer: response_3
D answer: response_4
Score:

Table 12: The prompt of the LLM evaluator gives scores on four response candidates.

<Question>: How long can the Information field be in a PPP data packet?
<Correct answer> (Standard Answer): The maximum length for the Information field, including the Padding field, is the maximum receive unit (MRU). The MRU defaults to 1500 bytes and can be negotiated.

A answer: The length of the information field is limited to 1500 bytes.
B answer: The information field can be up to 1500 bytes in length. The maximum size of the information field is specified by RFC 1661 and is set at 1500 bytes. This allows for the transmission of large packets, but also ensures that the protocol remains efficient and reliable.
C answer: The length of the information field is variable and can range from 0 to 1536 bytes.
D answer: The length of the information field is not specified, but it must be at least 1 byte. The information field is variable length and contains the protocol specific information that the peer requires to establish the link. The length of the information field is not specified, but it must be at least one byte. The format of the information field is defined by the protocol being used. For example, IPX uses the information field to specify the IPX network number and the IPX node address.

Score: A:5, B:5, C:4, D:2

Table 13: One-Shot Demonstration for the LLM Evaluator.

The scoring criteria for human evaluation:

0, penalty:
The candidate answers have issues such as repetitive sentences, which can significantly impair the helpfulness of the response.

1, Very low correlation:
Indicates that the candidate answer is almost entirely unrelated or opposite to the correct answer.

2, Low correlation:
Indicates that the candidate answer significantly deviates from the correct answer.

3, Moderate correlation:
Suggests that the candidate answer shares some similarities with the correct answer but may lack several key points or include extra unrelated content.

4, High correlation:
Indicates that the candidate answer is largely consistent with the correct answer, missing only minor points or details.

5, Very high correlation:
Signifies that the candidate answer is almost identical to or captures the complete essence of the correct answer.

Table 14: The scoring criteria for human evaluation.

Table Example 1: Basic information about the PLCh-Power-1

Item	Details
Description	Modem,PLCh-Power-1,Three-phase V200 PLC head module,No structural, built-in,DC 12V,NULL
Part Number	50030265
Model	PLCh-Power-1
Communication module type	Head-end Module

Markdown:
Table Caption: Basic information about the PLCh-Power-1
Item	Details
:—	:—
Description	Modem,PLCh-Power-1,Three-phase V200 PLC head module,No structural, built-in,DC 12V,NULL
Part Number	50030265
Model	PLCh-Power-1
Communication module type	Head-end Module

Template: The following sentences describe about Basic information about the PLCh-Power-1. The Description of the PLCh-Power-1 is "Modem,PLCh-Power-1,Three-phase V200 PLC head module,No structural, built-in,DC 12V,NULL". Its Part Number is 50030265. Its Model is PLCh-Power-1. Its Communication module type is Head-end Module.

TPLM-based: The description item is Modem,PLCh-Power-1,Three-phase V200 PLC head module,No structural, built-in,DC 12V,NULL. The part number item is 50030265. The model item is PLCh-Power-1. The communication module type item is Head-end Module.

LLM-based: The table provides details about the description, part number, model, and communication module type. The item described is a Modem, specifically the PLCh-Power-1, which is a three-phase V200 PLC head module. It is noted that it has no structural built-in and operates on DC 12V. The part number for this item is 50030265, and the communication module type is identified as a Head-end Module.

Table 15: A table-to-text example of four methods for a simple table with only two columns.

Table Example 2: Indicators on the PLC-IH-1

Silkscreen	Name	Color	Status	Description
-	PWR indicator	Green	Steady on	The module is powered on.
-	PWR indicator	-	Off	The module is powered off.
-	PLC_T/R indicator	Red	Blinking	The module is receiving data.
-	PLC_T/R indicator	Green	Blinking	The module is sending data.
-	Broadband carrier data sending status indicator of phase A	Green	Steady on	Broadband carrier data is sent through phase A.
-	Broadband carrier data sending status indicator of phase B	Green	Steady on	Broadband carrier data is sent through phase B.
-	Broadband carrier data sending status indicator of phase C	Green	Steady on	Broadband carrier data is sent through phase C.

Markdown:
Table Caption: Indicators on the PLC-IH-1
|Silkscreen| Name | Color| Status| Description|
|:—|:—|:—|:—|:—|
|-| PWR indicator| Green| Steady on| The module is powered on.|
-	PWR indicator	-	Off	The module is powered off.
-	PLC_T/R indicator	Red	Blinking	The module is receiving data.
-	PLC_T/R indicator	Green	Blinking	The module is sending data.
-	Broadband carrier data sending status indicator of phase A	Green	Steady on	Broadband carrier data is sent through phase A.
-	Broadband carrier data sending status indicator of phase B	Green	Steady on	Broadband carrier data is sent through phase B.
-	Broadband carrier data sending status indicator of phase C	Green	Steady on	Broadband carrier data is sent through phase C.

Template: The following sentences describe about Indicators on the PLC-IH-1. The Color of the PWR indicator is Green. Its Status is Steady on. Its Description is The module is powered on. The Status of the PWR indicator is Off. Its Description is The module is powered off. The Color of the PLC_T/R indicator is Red. Its Status is Blinking. Its Description is The module is receiving data. The Color of the PLC_T/R indicator is Green. Its Status is Blinking. Its Description is The module is sending data. The Color of the Broadband carrier data sending status indicator of phase A is Green. Its Status is Steady on. Its Description is Broadband carrier data is sent through phase A. The Color of the Broadband carrier data sending status indicator of phase B is Green. Its Status is Steady on. Its Description is Broadband carrier data is sent through phase B. The Color of the Broadband carrier data sending status indicator of phase C is Green. Its Status is Steady on. Its Description is Broadband carrier data is sent through phase C.

TPLM-based: The - silkscreen (also known as PWR indicator) is a function that can be expressed as a function. The - silkscreen is green. The - silkscreen is Steady on. The - silkscreen has the symbol The module is powered on.. The - silkscreen (also known as PWR indicator) is a function that can be expressed as a function. The - silkscreen is -. The - silkscreen is off. The - silkscreen has the symbol The module is powered off.. The - silkscreen (also known as PLC_T/R indicator) is a function that can be expressed as a function. The - silkscreen is red. The - silkscreen is blinking. The - silkscreen has the symbol The module is receiving data.. The - silkscreen (also known as PLC_T/R indicator) is a function that can be expressed as a function. The - silkscreen is green. The - silkscreen is blinking. The - silkscreen has the symbol The module is sending data.. The - silkscreen (also known as Broadband carrier data sending status indicator of phase A) is a function that can be expressed as a function. The - silkscreen is green. The - silkscreen is Steady on. The - silkscreen has the symbol Broadband carrier data is sent through phase A.. The - silkscreen (also known as Broadband carrier data sending status indicator of phase B) is a function that can be expressed as a function. The - silkscreen is green. The - silkscreen is Steady on. The - silkscreen has the symbol Broadband carrier data is sent through phase B.. The - silkscreen (also known as Broadband carrier data sending status indicator of phase C) is a function that can be expressed as a function. The - silkscreen is green. The - silkscreen is Steady on. The - silkscreen has the symbol Broadband carrier data is sent through phase C..

LLM-based: The table provides information about various indicators on a module. The "Silkscreen" column specifies the name of each indicator, while the "Color" column indicates the color of the indicator. The "Status" column describes the behavior of the indicator, such as whether it is steady on or blinking. The "Description" column provides additional details about the indicator's function. There are two indicators named "PWR indicator" in the table. The first one is green in color and is steady on, indicating that the module is powered on. The second "PWR indicator" is not specified for color and is off, indicating that the module is powered off. There are two indicators named "PLC_T/R indicator" in the table. The first one is red in color and is blinking, indicating that the module is receiving data. The second "PLC_T/R indicator" is green in color and is also blinking, indicating that the module is sending data. There are three indicators related to broadband carrier data sending status, each associated with a specific phase (A, B, and C). All three indicators are green in color and are steady on, indicating that broadband carrier data is being sent through each respective phase.

Table 16: A table-to-text example of four methods for a table with multiple columns and empty cells.

Table Example 3: Relationship between the device group type and device networking reliability

Type	Networking	Remarks
Multi-active device group	TOR devices configured with M-LAG	Two to eight devices that use the same device model and VTEP IP address can be added to a device group.
	Multi-active gateway or gateways configured with M-LAG	Only NVE devices support joining multi-active device groups.
	Multi-active NE routers	
	Multi-active vDHCP device group	Two devices can be added to one device group. They must use the same model and different IP addresses.

Markdown:
Table Caption: Relationship between the device group type and device networking reliability
Type	Networking	Remarks
:—	:—	:—
Multiple-active device group	ToR devices configured with M-LAG	Two to eight devices that use the same device model and VTEP IP address can be added to a device group.
Multiple-active device group	Multiple-active gateway or gateways configured with M-LAG	Two to eight devices that use the same device model and VTEP IP address can be added to a device group.
Multiple-active device group	Multiple-active NE routers	Two to eight devices that use the same device model and VTEP IP address can be added to a device group.
Multiple-active device group	Multiple-active vDHCP device group	Two to eight devices that use the same device model and VTEP IP address can be added to a device group.

Template: The following sentences describe about Relationship between the device group type and device networking reliability. The Networking of the Type named Multiple-active device group is ToR devices configured with M-LAG. Its Remarks is Two to eight devices that use the same device model and VTEP IP address can be added to a device group. The Networking of the Type named Multiple-active device group is Multiple-active gateway or gateways configured with M-LAG. Its Remarks is Two to eight devices that use the same device model and VTEP IP address can be added to a device group. The Networking of the Type named Multiple-active device group is Multiple-active NE routers. Its Remarks is Two to eight devices that use the same device model and VTEP IP address can be added to a device group. The Networking of the Type named Multiple-active device group is Multiple-active vDHCP device group. Its Remarks is Two to eight devices that use the same device model and VTEP IP address can be added to a device group.

TPLM-based: The multiple-active device group type is ToR devices configured with M-LAG. The multiple-active device group type is Two to eight devices that use the same device model and VTEP IP address can be added to a device group.. The multiple-active device group type is Multiple-active gateway or gateways configured with M-LAG. The multiple-active device group type is Two to eight devices that use the same device model and VTEP IP address can be added to a device group.. The multiple-active device group type is Multiple-active NE routers. The multiple-active device group type is Two to eight devices that use the same device model and VTEP IP address can be added to a device group.. The multiple-active device group type is Multiple-active vDHCP device group. The multiple-active device group type is Two to eight devices that use the same device model and VTEP IP address can be added to a device group.

LLM-based: The table provides information about different types of device groups and their configurations. The "Multiple-active device group" type can be configured with various networking options such as ToR devices with M-LAG, multiple-active gateway or gateways with M-LAG, multiple-active NE routers, and multiple-active vDHCP device group. Each of these types allows for the addition of two to eight devices that share the same device model and VTEP IP address.

Table 17: A table-to-text example of four methods for a table with multiple columns and merged cells.