IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 25 April 2025, accepted 11 May 2025, date of publication 14 May 2025, date of current version 21 May 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3569872

RESEARCH ARTICLE

Enhancing RAG Performance by Representing Hierarchical Nodes in Headers for Tabular Data

MINCHAE SONG
NongHyup Financial Research Institution, NongHyup Financial Group, Seoul 03739, Republic of Korea
e-mail: nicenara84@nonghyup.com

ABSTRACT Recent developments in retrieval-augmented generation (RAG) techniques have aimed at integrating structured tabular data with external data sources. Nevertheless, because existing approaches have difficulty effectively retrieving and reasoning from diverse and complex table structures, there is a growing demand for innovative methods that surpass traditional retrieval paradigms. This study introduced hierarchical node-based header representation models designed to enhance the processing and integration of external table data, addressing the key limitations of existing RAG techniques when working with structured table sources. By classifying table types designed for the structural complexity and the level of domain knowledge required for their interpretation, this study explores the impact of using embedded information to improve retrieval accuracy. The findings of this study demonstrate that presenting richer and more meaningful information within table headers significantly enhances RAG performance. This improvement is attributed to the utilization of table header representations, which allow for a more precise identification of hierarchical node-based headers during the table retrieval process. Furthermore, this study confirms that vector database achieves greater accuracy when employing a distributed table collection (DTC) structure than a unified context collection (UCC). These results indicate that the structure of the information extracted from text and tables varies considerably. Our experiment results highlight that separating and organizing such information using the DTC structure proved to be an effective strategy for improving the accuracy of table-specific responses.

INDEX TERMS Hierarchical nodes, retrieval-augmented generation (RAG), table retrieval, vector database.

I. INTRODUCTION

Generative artificial intelligence (AI) based on Large Language Models (LLMs) has recently demonstrated impressive utility across a diverse range of tasks, including mathematical reasoning, general-purpose problem-solving, and code generation [1]. A key factor in the success of LLMs such as GPT-4, CLIP, and DALL·E is the availability of large datasets sourced from the open web. Knowledge is implicitly stored in the weights of LLMs, which often comprise billions of parameters, enabling them to speak somewhat knowledgeably on open-domain topics. The richness and scale of both image and text data are key to this success, demonstrating the power of large, diverse datasets. LLMs are particularly adept at answering complex questions by generating step-by-step natural language reasoning steps, or "chains of thoughts," when prompted appropriately [2]. This approach has been successful when all the information needed to answer a question is either provided as context (e.g., algebraic questions) or assumed to be present in the model's parameters.

However, for many open-domain questions, all required knowledge is not always available or up-to-date in model parameters, and LLMs are prone to the well-documented phenomenon of "hallucination," where plausible statements are generated that are factually incorrect. These LLMs often confuse facts between two similar entities or make critical errors, in which a single incorrect token determines the accuracy of the response [3]. As a result, despite advancements in LLM technologies, LLMs are still limited to domain-specific tasks or knowledge-intensive applications, and generative AI in particular continues to face significant challenges in business applications [4]. This is primarily because most real business

The associate editor coordinating the review of this manuscript and approving it for publication was Gustavo Olague.

data are not publicly available and their access and usage are outside their organizations, preventing existing LLMs from training on such external knowledge. Given that retraining LLMs is both time-intensive and prohibitively expensive, there is increasing interest in approaches that mitigate hallucinations and produce more accurate outputs. One prominent solution is retrieval-augmented generation (RAG) [5].

In a typical RAG process, given an input query, the retriever identifies the most relevant information from external knowledge databases using a semantic similarity search and then provides this information as an additional context to the LLM to generate a response. This pipeline enables LLM to address the challenges associated with domain-specific tasks and improves its ability to accurately perform user-defined tasks [1]. Therefore, when applying RAG technology, LLMs can supplement their pre-trained knowledge with up-to-date information or detailed data from specific domains such as finance and law. As generative AI has become increasingly widespread, interest in RAG as a means of improving performance has grown, leading to rapid advancements in naive, advanced, and modular RAG [6]. However, this promising RAG technology has several limitations.

The limitations of RAG are its degraded performance when handling structured data, such as tabular data-based documents, which are frequently used and an important knowledge source [7], [8]. Tables composed of rows and columns along with schema metadata represent the structured information with interrelated attributes. The data typically have direct, clear meanings but often lack the nuanced contextual details present in textual datasets, which hinders RAG models from effectively retrieving information relevant to user queries. Even when a table is presented alongside textual information, LLMs often cannot determine a contextual interrelationship between the text and tabular information, frequently failing to retrieve the correct data segments and potentially generating hallucinated responses. Despite these limitations, structured tabular data remain a prevalent and essential data source across various industries, with estimates suggesting that over 70% of global data are stored in structured formats such as databases or spreadsheets [33]. This emphasizes the need for precise analysis and interpretation, which poses a fundamental challenge in RAG systems.

This study proposed an approach to efficiently process and integrate external table data, addressing the limitations of existing RAG techniques in handling various structured table sources. By incorporating classified types of table retrieval requiring domain-specific knowledge, such as finance, this research aims to investigate how such embedded information can enhance retrieval accuracy. The ultimate goal is to establish a practical and effective solution for improving generative AI services for business intelligence and other real-world applications. For this purpose, we developed a hierarchical node-based header representation when conducting data splitting and embedding processes in RAG systems. This approach is based on the heuristic yet plausible assumption that headers play a more significant role than indices when a table retrieves information from tables. However, in practice, many tables present complex information in their indices rather than in the headers. Therefore, this study does not rely solely on loading tables. Instead, it applies a preliminary transformation if a transposed table structure, achieved by rearranging headers and indices, is deemed more appropriate by the author. Furthermore, if domain-specific knowledge is embedded in indices or columns, this information is structured into a hierarchical node representation. Notably, the analysis results of this study confirm that presenting more complex and meaningful information in table headers enhances RAG performance. This improvement is attributed to the fact that retrievers utilize table header representations, which enable a more accurate identification of hierarchical node-based headers during the table retrieval process. This study contributes to the literature by proposing a method to enhance the performance of RAG system retrievers by representing hierarchical node structures for table headers embedded within simple two-dimensional tabular data consisting solely of rows and columns. This approach effectively addresses the limitations of the existing RAG systems when applied to tabular data.

The remainder of this paper is organized as follows: Section II discusses related work. Section III describes the experimental design, including the task and dataset description, proposed methodology, and baseline model. Section IV presents experimental results. Finally, Section V presents the results and limitations of this study, and discusses future work.

II. RELATED WORK

A. TYPICAL RAG APPLICATIONS

RAG techniques address several limitations inherent in LLMs, including outdated knowledge, insufficient coverage of long-tail information [9], and the risks associated with exposing private data sources [10] by leveraging a flexible external knowledge repository. Retrievable knowledge serves as a non-parametric memory that is easily updatable, capable of incorporating extensive long-tail information, and is suitable for encoding sensitive data without compromising privacy. In addition, retrieval-based approaches can reduce the computational costs associated with retraining LLMs, decrease model size requirements, extend the effective context window, and streamline generation processes by minimizing unnecessary computational steps [11].

A typical RAG process is shown in Figure 1. In RAG systems, queries directed to the LLM are used to search and retrieve pertinent documents from external sources, extending beyond the pre-trained database. By incorporating this external knowledge, RAG effectively addresses the problem of generating factually inaccurate responses and eliminates the need for retraining and adjusting weights, saving significant time and expense and resulting in considerable advancements and active research in this area [12].

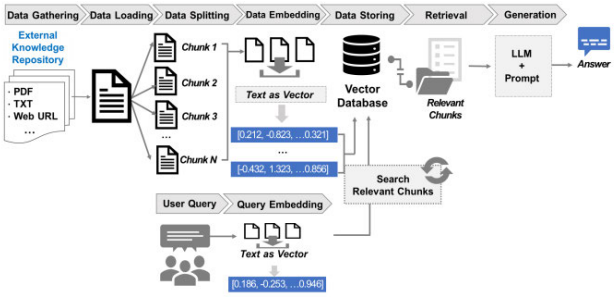


FIGURE 1. A generic RAG process.

The evolution of RAG in the context of LLMs demonstrates a distinct progression through several developmental stages. In its initial phase, RAG emerged alongside the rise of transformer-based architectures, primarily focusing on enhancing language models through external knowledge integration [13]. This foundational period emphasizes improving pre-training methodologies for more comprehensive knowledge retention. The introduction of ChatGPT [14] marked a significant milestone in this trajectory, as it showcased the in-context learning capabilities of LLMs. Consequently, research in RAG has shifted toward optimizing the retrieval process to support LLMs in handling more complex and knowledge-intensive queries during inference. As advancements continue, the scope of RAG extends beyond inference, incorporating fine-tuning strategies to further improve the performance and adaptability of LLMs in various knowledge-retrieval sources [6].

Despite its notable advancements, the effectiveness of RAG relies heavily on the quality of the retrieved knowledge [15], it is essential to assess the impact of various knowledge resources when considering the performance of RAG-based techniques. However, there is a limited body of research on tabular datasets as knowledge resources for RAG systems. Existing state-of-the-art (SOTA) LLMs are trained primarily on textual datasets, which results in suboptimal performance of RAG applications that require analytical tools or numeric data.

B. LEVERAGING TABULAR DATA AS KNOWLEDGE RESOURCES

As distinct forms of structured knowledge, tables can serve as effective external knowledge resources in retrieval-based systems. When properly leveraged, tabular data can significantly enhance the performance of open-domain QA (question answering) tasks by providing precise and contextually relevant information. However, one issue with employing LLMs for general-purpose tasks involving tabular data is that the unique characteristics of different applications may be overlooked, particularly in industries such as finance, law, and medicine. Most existing pre-training methods primarily utilize text or image data as training inputs because adapting these models to new tabular tasks requires an additional fine-tuning process. This adaptation is often challenging because of the limited accessibility of domain-specific tabular datasets, making data integration into pretraining pipelines less feasible [16].

Despite ongoing efforts to overcome these limitations through advanced generalization methods such as zero-shot learning for unseen tasks [17], [18] and in-context learning with newly presented tabular instances [19], [20], the resulting performance of these methods often reflects restricted generalization capabilities and limited practical applications. For instance, LIFT [19], which evaluates fine-tuning against in-context learning across six classification tasks, applies a narrow assessment framework that focuses primarily on accuracy, potentially overlooking the broader generalization potential of the model across diverse contexts. Similarly, TabPFN [20], which employs a prior data-fitted network designed for numerical feature classification datasets [21], lacks support for zero-shot inference, thereby limiting its adaptability. Another attempt aims to enhance the LLMs comprehension of tabular data by incorporating instruction-tuning datasets customized for specific tabular structures [22]. Moreover, models such as TabLLM [17], UniTabE [23], and TransTab [18] have demonstrated success in zero-shot scenarios; however, their applications remain confined to specific classification tasks, often neglecting in-context learning. These models also rely heavily on fine-tuning for task-specific adaptation. This constrained scope highlights the need for a more comprehensive tabular data-processing framework that enables more advanced generalization capabilities and broader, more adaptive applications. Additionally, all these approaches focus on utilizing tabular data as part of LLM training datasets rather than employing the data as external knowledge resources in RAG systems.

With the advancement of ''retrieve-and-read'' RAG frameworks, a growing body of research has incorporated tabular data into RAG systems. These studies address table QA as two separate sub-tasks: use of a retriever to select relevant table candidates from a corpus and use a reader to locate correct answers from table candidates. For example, the DTR [24] and CLTR [25] each adopt a two-stage framework in which a retriever first selects a set of candidate tables, followed by a reader that extracts answers from the retrieved tables. However, these approaches face inherent limitations owing to the independent training of the two components, leading to error propagation; incorrectly retrieved tables increase the likelihood of the reader identifying incorrect answers. To address this issue, more recent approaches, such as T-RAG [26], dual reader-parser [27], and OmniTab [28] integrate the retrieval and answer generation processes by jointly training both components. This end-to-end architecture allows for more effective answer generation by eliminating cascading errors typical of multistage pipelines. In addition, the CARP model [29] extracts a hybrid chain of retrieved tables and passages for prompt construction, while

the StructGPT [30] retrieves from multiple sources, including knowledge graphs (KGs), tables, and databases. A cTBLS (Conversational Tables) architecture [31] then forms prompt with ranked tables after retrieval. Next, ERATTA [32] and TableGPT1·2 [33], [34] generate SQL code to extract table information and integrate it into the prompt to minimize model hallucinations. These approaches integrate table information into prompts for query-based RAG, which is generally more challenging for end-to-end Table QA than executing SQL queries over relational database tables owing to the lack of schema information.

Although several methods have been explored to utilize tabular data, existing research has largely focused on relatively simple table structures. However, there remains a substantial gap in addressing the challenges posed by the diverse and complex table formats frequently encountered in real-world business applications. Complex tables, characterized by hierarchical headers and a multitude of values, pose significant challenges for generators in processing extensive datasets. Consequently, insufficient table comprehension can lead to incorrect answers. To address this gap, this study introduces a novel approach that incorporates the hierarchical relationships of tabular data columns and rows as a hierarchical node-based header representation, thereby enabling a more effective analysis of such complex and varied table structures.

III. EXPERIMENT SETUP

A. TASK DESCRIPTION

1) TASK: FACTOID QUESTION-ANSWERING

While there are a number of different question formats, this study is primarily concerned with factoid QA, as defined within the QA framework used at the Text REtrieval Conference (TREC). Factoid QA tasks are designed to provide responses to fact-based queries with concise and specific answers that are typically derived from structured or unstructured data sources [35]. These tasks are designed to answer questions with clear, factual answers, such as names, dates, locations, definitions, or numerical data. For example, "*Who is the wealthiest person in the world in 2024?", "What is the total number of ChatGPT users?"*, and "*Where is Google based?*" are examples of the factoid questions. Our study aimed to enhance the indexing and retrieval performance of existing RAG techniques to address domain-specific knowledge requirements, such as those in the finance and law sectors.

2) EVALUATION: ACCURACY

In this study, a prediction is considered correct if any substring of the prediction is an exact match of any of the gold answers. For factual knowledge generation, we established baseline models by applying an RAG system to retrieve the external table-text knowledge without data transposition and incorporating hierarchical nodes, and found that the proposed technique generated more correct responses than the baseline model.

B. BASELINE MODEL

The traditional RAG system follows a process that includes indexing, retrieval, and generation, and is also characterized as a "retrieve-read" framework. For indexing and retrieval, we employed Python-based frameworks, including LangChain and FlashRank, to process the information into chunks for embedding and retrieval. To optimize data retrieval, vector database techniques were utilized, with ChromaDB serving as the vector database solution, and cosine similarity applied for loss metrics. In this study, the LangChain framework was used to generate chunks and retrieval was performed using FlashRank. Finally, the embedding model "text-embedding-ada-002-v2 for GPT-3.5-turbo" is integrated for QA generation.

1) TWO-STAGE RETRIEVAL WITH RERANKING

In the retrieval process depicted in Figure 1, the most relevant documents are identified from the vector database collections before being transferred to the LLM. To achieve this, a similarity search was conducted to locate documents closely aligned with the user's query. This study incorporated an additional verification step to rank retrieved documents using similarity scores. Prior research [36] found that increasing the number of retrieved documents from the vector database can enhance retrieval recall; however, sending an excessive number of documents directly to the LLM can also negatively impact its recall performance. To address this challenge, the retrieval strategy is optimized by first retrieving a larger set of documents to maximize recall, and then filtering this set to ensure that only the most relevant documents are forwarded to the LLM. This is achieved through two-stage retrieval with a re-ranking model that reorders the retrieved documents and retains only those most relevant for effective LLM input. The re-ranking approach assigns scores to the retrieved documents, prioritizing those that are most pertinent before they are utilized by the response generation process. By emphasizing the most relevant information, reranking minimizes the inclusion of extraneous or less-useful content, thereby enhancing the efficiency and accuracy of the RAG system [37].

In search systems, re-rankers are commonly employed as a part of a two-stage retrieval process. In this approach, the initial stage uses an embedding model or retriever to identify a subset of relevant documents in a large dataset. Subsequently, in the second stage, a re-ranking model is applied to reorder the retrieved documents, ensuring that the most relevant documents are prioritized for downstream tasks (see Figure 2).

2) EMBEDDING MODEL

With the recent popularization of LLMs and embeddings-as-a-service (EaaS), we applied text embedding-ada-002

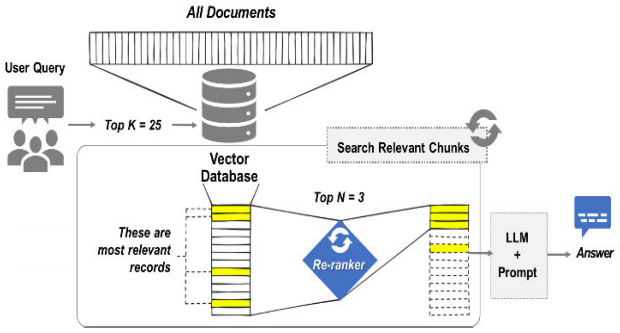


FIGURE 2. An example of reranking and two-stage retrieval processes.

(ADA-002) in this study. ADA-002 is OpenAI's second-generation EaaS API endpoint model, specifically adapted for text embedding, and includes multilingual support (including Korean). Since the release of ADA-002 as a proprietary commercial product in late 2022, no specific details regarding its architecture, training dataset, or underlying model have been made publicly available.

The following LLMs were used in the robustness testing of the proposed model: OpenAI's text-embedding-3-large and text-embedding-3-small. Information regarding the model and parameter sizes of text-embedding-3-large and text-embedding-3-small, similar to that of ADA-002, has not been disclosed. Their context lengths and embedding sizes are listed in Table 1.

TABLE 1. Embedding model descriptions.

Embedding Model	Context length	Embedding Size
text-embedding-ada-002	8,191	1,536
text-embedding-3-small	8,191	3,072
text-embedding-3-large	8,191	1,536

Model size and number of parameters are not available.

3) LARGE LANGUAGE MODEL

We chose GPT-3.5-turbo as an LLM baseline due to its recency, SOTA performance, and diverse architectures. The model represents a range of capabilities and architecture designs, from the versatile and widely adopted GPT series to the specialized mixture-of-experts approach in Mistral AI. GPT-3.5-turbo performed well due to its autoregressive architecture, large training data corpus, and number of parameters [38].

The robustness of the proposed method was further examined through the application of additional large language models, specifically OpenAI's GPT-4 Omni (GPT-4o) and GPT-4o-mini. OpenAI introduced GPT-4o in May 2024 as an upgraded model featuring a larger parameter set, enhanced performance, and faster processing. GPT-4o-mini, a more streamlined and cost-effective version of GPT-4o, was subsequently launched to address the increased efficiency demands. To optimize the cost and processing time, OpenAI's batch processing approach was employed, as it can simultaneously address multiple requests at reduced costs and with improved granularity [39].

4) HYPERPARAMETER SETTINGS

Hyperparameters are important in determining the efficiency, accuracy, and overall performance of a RAG-based QA task [40]. In our RAG system implementation, seven hyperparameters were utilized for data chunking, indexing, retrieval, and generation processes, as listed in Table 2.

TABLE 2. Hyperparameter descriptions.

Parameter	Description	Value
Chunk size[1]	The size of each text chunk generated during the preprocessing stage, typically in measure tokens. It ensures coherent and manageable text segments.	120 characters
Chunk overlap[1]	The number of overlapping characters between consecutive chunks. This helps in maintaining context across chunks.	50 characters
Retriever search K	Top relevant document chunks extracted based on their similarity to the query vector.	5
QA prompt template	The structure or format used to guide a language model to generate answers.	See Note 2
Embedding model	The model used to generate dense vector embedding from text chunks.	text-embedding-ada-002-v2
LLMs	The language model used for generating answers based on the retrieved document chunks and the user query.	GPT-3.5-turbo
Vector database	The type of database used to index and store vector embedding for efficient retrieval.	Chroma – open-source embedding vector database

1. The tabular text documents used in this study consisted of passages with brief descriptions of the tables. Therefore, considering the length of the sentences, the chunk size and overlap were set to small values.
2. You are a customer service agent that helps a customer with answering questions. Please answer the question based on the provided context below. Make sure not to make any changes to the context, if possible, when preparing answers so as to provide accurate responses. If the answer cannot be found in context, politely say that you do not know. Do not try to make up an answer.

C. PROPOSED METHODOLOGY

Figure 3 illustrates the workflow of the RAG system for tabular data incorporating the proposed method in this study. Complex tables often include hierarchical structures in headers or rows and contain a large volume of data, making it challenging for generators to embed and retrieve information effectively. In this study, various data representations were considered, and transformations were applied to ensure that the axis containing more significant information between the header and index was designated as the header. Furthermore, the domain-specific knowledge embedded in the table was structured as hierarchical nodes to ensure that key information was clearly represented in the table headers. To achieve

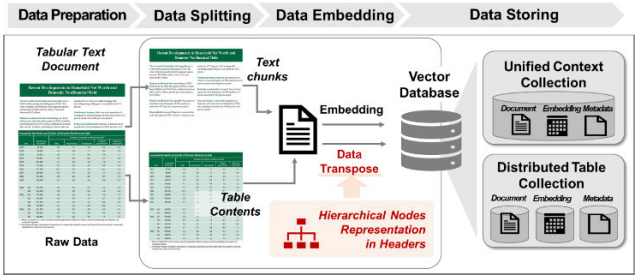


FIGURE 3. RAG system workflow for tabular data featuring the proposed method.

this, the data were first loaded by separating tables from the accompanying short text passages. The text was then segmented into fixed-size chunks, and each chunk underwent an encoding process to generate and store the embedding vectors. In contrast, the tables are not divided into chunks but are split into individual records. Each record was stored along with the corresponding header and index information from the table. The hierarchical nodes proposed in this study are then attached to each record.

1) TABLE RETRIEVER

In this study, we adopted the framework of the Dense Passage Retrieval (DPR) model outlined by [41]. Given M target tables T, the objective of our proposed model is to retrieve the *Top_K* candidate tables that contain information most relevant to the question Q. To achieve this, we employ two distinct encoders: a table header encoder (Enc_T) and a question encoder (Enc_Q). Enc_T processes the headers of the M target tables, generating representations t that are indexed for retrieval. The input x_t to Enc_T is formally defined in (1). When a query Q is provided, the question encoder Enc_Q computes a query representation q. Subsequently, the *Top_K* tables are retrieved by selecting the closest candidate table headers *t*from the indexed representations based on the similarity measure defined as the dot product in (2). The pseudocode illustrating the generation of hierarchical nodes for table headers is presented in Figure 4. The hierarchical nodes for an example target table, indexed in Figure 5, are initially constructed and subsequently utilized as representations of the table headers.

$$x_t = \{[CLS]content[SEP]header_{col}[SEP]header_{row}[SEP]\} \tag{1}$$

$$Sim(q, t) = Enc_Q(Q)^T \cdot Enc_T(x_t) \tag{2}$$

A key distinction between our retrieval approach and prior studies lies in the representation of the table headers. We apply a preliminary transformation if a transposed table structure achieved by rearranging headers and indices is deemed more appropriate by the author. Specifically, our method incorporates hierarchical nodes that capture the structure and contextual information of tables, which enhances the retrieval system by enabling a more nuanced understanding of contextual and detailed table information. Consequently, this significantly improves the identification of relevant table values in response to user queries. An illustration of the extracted headers is shown in Figure 6. In addition, metadata are appended to both textual and tabular data as the document's title (source) and data type (e.g., text or table).

The effectiveness of the hierarchical nodes in the header representation proposed in this research varies depending on the structural complexity of the tables and the level of domain knowledge required for their interpretation. To address this, we categorize the tables into four distinct types (I–IV), as shown in Figure 5. Type I and Type II tables differ in complexity, with Type I having a simpler structure for headers and indices and Type II having a more complex structure. Neither type requires domain-specific knowledge; therefore, the integration of hierarchical nodes into header representations was not applied. However, because the starting point of the proposed approach is to ensure that the header encapsulates more important information from the table, the data are transposed accordingly. Figure 6 presents examples of the actual data used for the analysis based on the criteria outlined in Figure 5.

Furthermore, this study analyzes and compares the effectiveness of the proposed method by examining how indexed information is stored in the vector database, comparing unified context collection (UCC) with distributed table collection (DTC), and assessing the impact of different vector database structures on performance. Although the UCC and DTC approaches were not developed in this study, it is important to examine how the suggested hierarchical node approach interacts with different structures to function more effectively for table retrievers. Several studies [42], [43] have integrated RAG with knowledge graphs (KGs), which represent structured data, and vector databases, which store unstructured information, to enhance domain-specific factual retrieval. By incorporating relevant subgraphs that preserve structured context, these approaches have demonstrated reductions in hallucinations, and achieved improved performance in specialized QA tasks. However, prior research has primarily focused on utilizing text data as the primary external knowledge source within RAG frameworks. Furthermore, these studies have primarily focused on enhancing RAG performance by utilizing external datasets structured as KGs, rather than employing related vector databases. As the RAG system tightly integrates indexing, retrieval, and generation, it is important to combine these components to enhance the performance of each individual element, including vector store [44].

2) VECTOR DATABASE: UCC VS. DTC

To accommodate the context limitations of LLMs, text is segmented into smaller and more easily processed chunks. These chunks were then encoded into vector representations using an embedding model and stored in a vector database. This allows efficient similarity searches in the subsequent retrieval

```
# Import the Table of Type IV
SET data = READ_CSV_FILE("Table_Type IV.csv")

# Transpose the Table to Place Key Information in the Header
SET data = TRANSPOSE(data)

# Define a Tree Node Structure
CLASS TreeNode:
  PROPERTIES:
    value
    children (empty list)
    parent (default = null)

  FUNCTION add_child(child_node):
    SET child_node.parent = this
    APPEND child_node to children

  FUNCTION print_tree(level = 0):
    SET indent = "    " repeated level times
    PRINT indent + value
    FOR each child in children:
      CALL child.print_tree(level + 1)

# Extract Level Mapping from Data
FUNCTION extract_level_mapping(data):
  SET header_row = data.row[0]
  SET level_mapping = empty dictionary

  FOR each column in data.columns (excluding first column):
    SET base_name = SPLIT(column, '.')[0] with trimmed whitespace
    IF base_name not in level_mapping:
      SET level_mapping[base_name] = empty list
    APPEND header_row[column] to level_mapping[base_name]

  RETURN level_mapping

# Build a Custom Hierarchy Tree Node
FUNCTION build_custom_tree(data):
  SET root = NEW TreeNode("Root: Outstanding credit guarantee balance")
  SET level_mapping = CALL extract_level_mapping(data)

  FOR each level1, level2_list in level_mapping:
    SET child_node = NEW TreeNode("Child: " + level1, parent = root)
    CALL root.add_child(child_node)

    FOR each level2 in level2_list:
      SET grandchild_node = NEW TreeNode("Grandchild: " + TRIM(level2), parent = child_node)
      CALL child_node.add_child(grandchild_node)

# Export the Hierarchy Tree Node
SET tree_root = CALL build_custom_tree(data)
```

FIGURE 4. A pseudocode for generating the hierarchical nodes of the table headers.

phase. The vector database for converting chunks into vectors also influences the efficiency of similarity retrieval. The most common approach to this task is the use of UCC. UCC stores all the information and metadata in a single data location along with the vectors used for the semantic search. This stored information is called collection. Another approach is to use DTC, in which information, metadata, and vectors are stored in separate collections according to the type of information. In this environment, there are multiple collections rather than one central location. UCC integrates documents, metadata, and embeddings for comprehensive

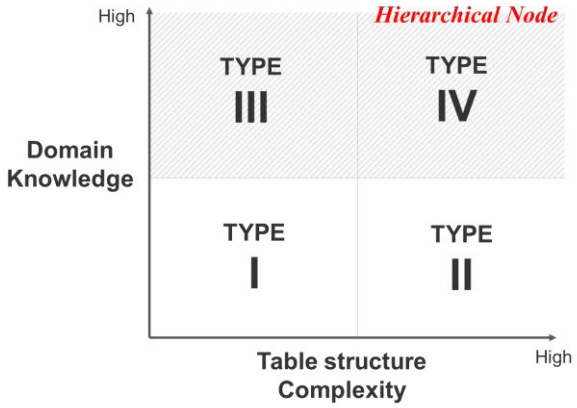


FIGURE 5. Criteria for classifying table types.

data representation, whereas DTC handles its respective components separately as documents, metadata, and embedding types.

Each method has advantages and limitations. For example, DTC may entail more complex management owing to the requirements of data synchronization across multiple nodes, inconsistency resolution, and distributed database efficiency. Nonetheless, DTCs offer significant scalability and fault tolerance, enabling the system to handle increased workloads and recover swiftly from node failure. By contrast, UCCs simplify data management by centralizing all information in a single repository, ensuring consistent and straightforward administration. However, they may face challenges related to scalability and may be susceptible to becoming a single point of failure. The following analysis evaluates both configurations to determine their impacts on the RAG performance.

D. DATASET DESCRIPTION

This study utilizes a dataset comprising passage-question-answer triples derived from professional documents in the finance domains, written in Korean and made publicly available by AI-Hub(https://www.aihub.or.kr/). The original dataset categorized data labeling tasks into five types based on the question-answer format: boundary extraction, short yes/no answers, table-based answer extraction, multiple choice, and procedural (method-based). As this study focused on analyzing tabular data, only the table-based answer extraction type was selected. This type consists of question-answer pairs related to the content of tables with rows and columns, where the answers can be identified within individual table cells. In this context, each table is paired with a brief text description to form a single set comprising the table and its corresponding description. The data structure of the table-based answer extraction method is shown in Figure 8.

As mentioned in Section III-C, this study categorized tables into four major types (see Figure 5), with ten tables of each type selected for experimentation. The table-based answer extraction dataset consisted of ten questions with corresponding answers for each table. Therefore,

Raw format of an example target table

Type Ⅳ ①

		End of			②	Composition Ratio
		2008	2009	2010	October 2011	
Total	Outstanding Credit Guarantee Balance	12,824	17,282	18,887	20,041	100
Institution	Credit Guarantee Fund	6,211	8,070	8,334	8,735	43.6
	Technology Credit Guarantee Fund	2,765	3,569	3,736	3,723	18.6
	Gangwon Credit Guarantee Foundation	3,848	5,643	6,817	7,583	37.8
Guarantee Type	Loan Guarantee	12,234	16,492	18,204	19,345	96.5
	Deposit Bank	9,048	11,279	12,034	12,724	63.5
	Non-Banking Institution	3,186	5,214	6,170	6,621	33
	Bill Guarantee	368	424	384	364	1.8
	Others / Miscellaneous	222	366	298	332	1.7
Industry	Manufacturing	4,338	5,473	5,679	5,829	29.1
	Construction	1,766	2,260	2,312	2,363	11.8
	Wholesale and Retail Trade	3,904	5,362	5,906	6,283	31.4
	Food and Accommodation Services	877	1,401	1,617	1,875	9.4
	Others / Miscellaneous	1,939	2,786	3,374	3,691	18.4

Hierarchical nodes of an example target table in indices

Root: Outstanding credit guarantee balance
- *Child: Total,*
- *Child: Institution, and*
  - *Grandchild: Credit Guarantee Fund*
  - *Grandchild: Technology Credit Guarantee Fund*
  - *Grandchild: Gangwon Credit Guarantee Foundation*
- *Child: Guarantee Type, and*
  - *Grandchild: Loan Guarantee*
  - *Grandchild: Loan Guarantee of Deposit Bank*
  - *Grandchild: Loan Guarantee of Non-Banking Institution*
  - *Grandchild: Bill Guarantee*
  - *Grandchild: Others / Miscellaneous*
- *Child: Industry, and*
  - *Grandchild: Manufacturing*
  - *Grandchild: Construction*
  - *Grandchild: Wholesale and Retail Trade*
  - *Grandchild: Food and Accommodation Services*
  - *Grandchild: Others / Miscellaneous*

Input x_t of an example target table ①, ②

① *{**'content'**: 'Period: End of 2008 Root: Outstanding credit guarantee balance, and Child: Total: 12,824 Root: Outstanding credit guarantee balance, and Child: Institution, and Grandchild: Credit Guarantee Fund:6,211 Root: Outstanding credit guarantee balance, and Child: Institution, and Grandchild: Grandchild: Technology Credit Guarantee Fund: 2,765 Root: Outstanding credit guarantee balance, and Child: Institution, and Grandchild: Gangwon Credit Guarantee Foundation: 3,848 Root: Outstanding credit guarantee balance, and Child: Guarantee Type, and Grandchild: Loan Guarantee: 12,234 Root: Outstanding credit guarantee balance, and Child: Guarantee Type, and Grandchild: Loan Guarantee of Deposit Bank: 9,048 Root: Outstanding credit guarantee balance, and Child: Guarantee Type, and Grandchild: Loan Guarantee of Non-Banking Institution: 3,186 Root: Outstanding credit guarantee balance, and Child: Guarantee Type, and Grandchild: Bill Guarantee: 368 Root: Outstanding credit guarantee balance, and Child: Guarantee Type, and Grandchild: Others / Miscellaneous: 222 Root: Outstanding credit guarantee balance, and Child: Industry, and Grandchild: Manufacturing: 4,338 Root: Outstanding credit guarantee balance, and Child: Industry, and Grandchild: Construction: 1,766 Root: Outstanding credit guarantee balance, and Child: Industry, and Grandchild: Wholesale and Retail Trade: 3,904 Root: Outstanding credit guarantee balance, and Child: Industry, and Grandchild: Food and Accommodation Services: 877 Root: Outstanding credit guarantee balance, and Child: Industry, and Grandchild: Others / Miscellaneous: 1,939', **'metadata'**: {'source': 'Current Status and Challenges of Financing for SMEs in the Gangwon Region','line': 0, 'type': 'table'}}*

FIGURE 6. An example of target table retrieval representation.

400 experimental data items were used, as the study involved analyzing four distinct types of tables.

②*{**'content'**: 'Period: End of October, 2011 composition ratio Root: Outstanding credit guarantee balance, and Child: Total: 100.0 Root: Outstanding credit guarantee balance, and Child: Institution, and Grandchild: Credit Guarantee Fund: 43.6 Root: Outstanding credit guarantee balance, and Child: Institution, and Grandchild: Grandchild: Technology Credit Guarantee Fund: 18.6 Root: Outstanding credit guarantee balance, and Child: Institution, and Grandchild: Gangwon Credit Guarantee Foundation: 37.8 Root: Outstanding credit guarantee balance, and Child: Guarantee Type, and Grandchild: Loan Guarantee: 96.5 Root: Outstanding credit guarantee balance, and Child: Guarantee Type, and Grandchild: Loan Guarantee of Deposit Bank: 63.5 Root: Outstanding credit guarantee balance, and Child: Guarantee Type, and Grandchild: Loan Guarantee of Non-Banking Institution: 33.0 Root: Outstanding credit guarantee balance, and Child: Guarantee Type, and Grandchild: Bill Guarantee: 1.8 Root: Outstanding credit guarantee balance, and Child: Guarantee Type, and Grandchild: Others / Miscellaneous: 1.7 Root: Outstanding credit guarantee balance, and Child: Industry, and Grandchild: Manufacturing: 29.1 Root: Outstanding credit guarantee balance, and Child: Industry, and Grandchild: Construction: 11.8 Root: Outstanding credit guarantee balance, and Child: Industry, and Grandchild: Wholesale and Retail Trade: 31.4 Root: Outstanding credit guarantee balance, and Child: Industry, and Grandchild: Food and Accommodation Services: 9.4 Root: Outstanding credit guarantee balance, and Child: Industry, and Grandchild: Others / Miscellaneous: 18.4', **'metadata'**: {'source': 'Current Status and Challenges of Financing for SMEs in the Gangwon Region', 'line': 4, 'type': 'table'}}*

FIGURE 6. ***(Continued.)*** An example of target table retrieval representation.

TABLE 3. Dataset size and composition.

Type	Number of tables	Number of questions	Total
I	10	10	100
II	10	10	100
III	10	10	100
IV	10	10	100
Total	40	10 (avg.)	400

IV. RESULTS

As the base model for comparison, this study employed a generation process utilizing an RAG framework, following a traditional retrieve-then-read approach to access relevant external knowledge sources. The performance of the RAG system with data transposition and integration of hierarchical nodes into header representations for retrieving table-text

Type I	Gangwon (City name)			Nationwide		
	2013	2014	2015	2013	2014	2015
Sales growth rate	4.3	1.3	3.4	0.5	-1.6	-3
Total asset growth rate	4.8	5.8	5.5	5.6	4	3.7
Tangible asset growth rate	3.1	4.1	5.2	4.8	3.2	5.2

Type II		Total	1st Quartile	2st Quartile	3st Quartile	4st Quartile
Jeju (City name)	2015.3(A)	26,258	1,109	8,372	23,853	76,192
	2021.3(B)	49,153	1,512	15,168	42,881	141,128
	Change(B-A)	22,896	402	6,796	19,029	64,935
Nationwide	2015.3(A)	30,325	2,267	11,290	25,825	84,036
	2021.3(B)	43,736	2,734	14,920	36,412	123,575
	Change(B-A)	13,411	467	3,630	10,587	39,540

Type III		2006	2007	2008	2009	2010	2011
Initial Public Offering (IPO)	Large Enterprise	9,906	13,953	2,141	3,606	27,726	13,147
	SMEs	21,948	101,217	10,783	51,303	29,180	44,836
Equity Capital Increase	Large Enterprise	1,261	1,300	1,063	1,121	1,660	-
	SMEs	1,947	3,582	3,900	5,296	2,207	344
Corporate Bonds Issuance	Large Enterprise	273,230	346,452	641,039	946,723	1,010,143	982,090
	SMEs	4,165	7,644	16,548	11,292	7,815	7,912
Subtotal	Large Enterprise (A)	284,397	361,705	644,243	951,450	1,039,529	995,237
Composition Ratio (A/C)		91.0	76.3	95.4	93.3	96.4	94.9
Subtotal	SMEs (B)	28,060	112,443	31,231	67,891	39,202	53,092
Composition Ratio (B/C)		9.0	23.7	4.6	6.7	3.6	5.1
Direct Financing Total (A)		312,457	474,148	675,474	1,019,341	1,078,731	1,048,329

Type IV		End of				Composition Ratio
		2008	2009	2010	October 2011	
Total	Outstanding Credit Guarantee Balance	12,824	17,282	18,887	20,041	100
Institution	Credit Guarantee Fund	6,211	8,070	8,334	8,735	43.6
	Technology Credit Guarantee Fund	2,765	3,569	3,736	3,723	18.6
	Gangwon Credit Guarantee Foundation	3,848	5,643	6,817	7,583	37.8
Guarantee Type	Loan Guarantee	12,234	16,492	18,204	19,345	96.5
	Deposit Bank	9,048	11,279	12,034	12,724	63.5
	Non-Banking Institution	3,186	5,214	6,170	6,621	33
	Bill Guarantee	368	424	384	364	1.8
	Others / Miscellaneous	222	366	298	332	1.7
Industry	Manufacturing	4,338	5,473	5,679	5,829	29.1
	Construction	1,766	2,260	2,312	2,363	11.8
	Wholesale and Retail Trade	3,904	5,362	5,906	6,283	31.4
	Food and Accommodation Services	877	1,401	1,617	1,875	9.4
	Others / Miscellaneous	1,939	2,786	3,374	3,691	18.4

FIGURE 7. Examples of table types depending on domain knowledge and table structure complexity.

<Title> 2024.07. Household Net Worth

The net worth of households and nonprofit organizations increased by $5.1 trillion to $160.8 trillion in the first quarter. The change in household net worth was primarily driven by an increase in the value of directly and indirectly held corporate equity, which rose by $3.8 trillion in the first quarter. The value of real estate on the household balance sheet rose by $0.9 trillion.

Household Balance Sheet Summary

Trillions of dollars

Description	2022	2023	2023:Q2	2023:Q3	2023:Q4	2024:Q1
Net Worth	144.68	155.73	153.18	151.77	155.73	160.84
Change in Net Worth	-5.62	11.05	5.29	-1.41	3.95	5.12
Equities	39.74	47.13	44.72	43.04	47.13	50.96
Change in Equities	-9.33	7.39	2.43	-1.68	4.09	3.83
Real Estate	42.59	44.90	44.76	45.27	44.90	45.84
Change in Real Estate	3.69	2.31	2.22	0.51	-0.36	0.93

<Sources >Board of Governors of the Federal Reserve System

✓ Question.1 : How much did the household net worth increase in the first quarter of 2024 compared to the previous quarter?

Answer : 5.12

...

✓ Question.10: What is the size of Real Estate in the second quarter of 2023?

Answer : 44.76

FIGURE 8. An example of tabular text documents in this study.

knowledge was assessed in comparison with the established baseline method.

As shown in Figure 9, the approach combining GPT-3.5 and ADA-002 with RAG, applied to cases where the data were transposed, exhibited superior performance across both

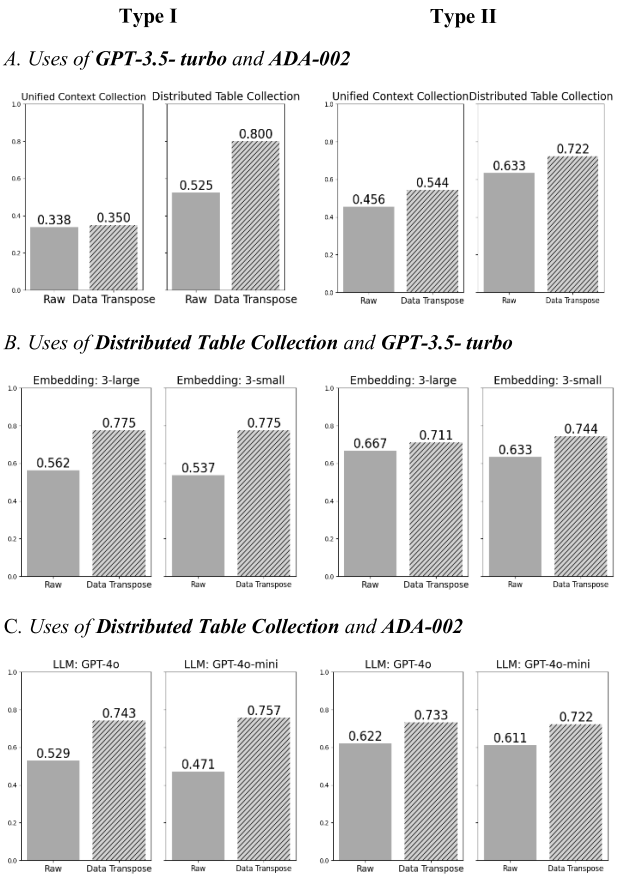


FIGURE 9. Accuracy of factoid QA RAG for types I and II.

Type I and II tables. This performance notably exceeded that observed in cases where the data were not transformed. Additionally, cases with a DTC vector database structure were more accurate than those with a UCC structure, regardless of whether the data were baseline or transposed. To further ensure the reliability of the evaluations, experiments were conducted by varying the embedding model and the LLM. The results demonstrated that the proposed method, which modifies the structure to position more critical information in the header, outperformed baseline methods. Consistent with the baseline configurations, the proposed approach provides more accurate responses in the DTC structure. These results indicate that the DTC structure is more suitable than UCC for vector database in table retrieval tasks.

Table types III and IV require domain-specific knowledge, with Type III characterized by a simple structure for indices and headers, and Type IV exhibiting a more complex structure. As discussed in Section III-C, this study represents domain-specific knowledge in the form of hierarchical nodes. Similar to Types I and II, Types III and IV include critical table information in their indices. Therefore, the proposed approach transposes the data and converts the knowledge embedded in the original indices into a hierarchical node format, setting these nodes as the headers.

TABLE 4. Accuracy of factoid QA RAG for types I.

Method	Embedding	LLM	UCC	DTC
Raw	text-embedding-ada-002	gpt-3.5-turbo	0.338	0.525
Transpose			**0.350**	**0.800**
Raw		gpt-4o	0.229	0.529
Transpose			**0.329**	**0.743**
Raw		gpt-4o-mini	0.271	0.471
Transpose			**0.329**	**0.757**
Raw	text-embedding-3-large	gpt-3.5-turbo	0.325	0.562
Transpose			**0.350**	**0.775**
Raw	text-embedding-3-small		0.313	0.537
Transpose			**0.338**	**0.775**

TABLE 5. Accuracy of factoid QA RAG for types II.

Method	Embedding	LLM	UCC	DTC
Raw	text-embedding-ada-002	gpt-3.5-turbo	0.456	0.633
Transpose			**0.544**	**0.722**
Raw		gpt-4o	0.478	0.622
Transpose			**0.556**	**0.733**
Raw		gpt-4o-mini	0.444	0.611
Transpose			**0.578**	**0.722**
Raw	text-embedding-3-large	gpt-3.5-turbo	0.456	0.667
Transpose			**0.556**	**0.711**
Raw	text-embedding-3-small		0.444	0.633
Transpose			**0.556**	**0.744**

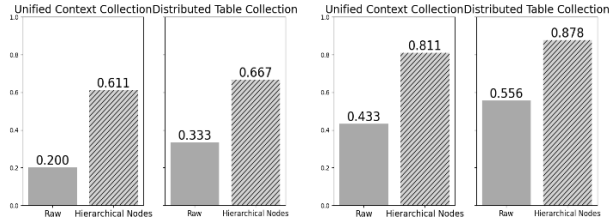
TABLE 6. Accuracy of FACTOID QA RAG for types III.

Method	Embedding	LLM	UCC	DTC
Raw	text-embedding-ada-002	gpt-3.5-turbo	0.200	0.333
Hierarchical Nodes[1]			**0.611**	**0.667**
Raw		gpt-4o	0.200	0.289
Hierarchical Nodes[1]			**0.544**	**0.633**
Raw		gpt-4o-mini	0.189	0.337
Hierarchical Nodes[1]			**0.539**	**0.663**
Raw	text-embedding-3-large	gpt-3.5-turbo	0.216	0.330
Hierarchical Nodes[1]			**0.528**	**0.656**
Raw	text-embedding-3-small		0.211	0.322
Hierarchical Nodes[1]			**0.544**	**0.667**

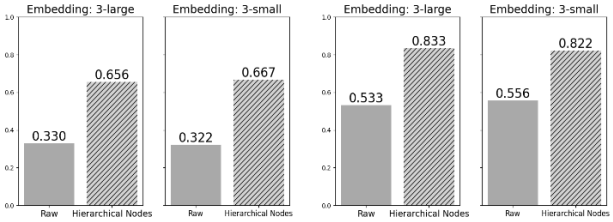
1. Hierarchical Nodes indicate that represents hierarchical nodes representation in the header after transposing the table.

Type III **Type IV**

*A. Uses of **GPT-3.5- turbo** and **ADA-002***



*B. Uses of **Distributed Table Collection** and **GPT-3.5- turbo***



*C. Uses of **Distributed Table Collection** and **ADA-002***

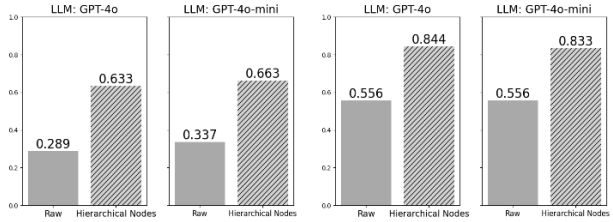


FIGURE 10. Accuracy of factoid QA RAG for types III and IV.

TABLE 7. Accuracy of factoid QA RAG for types IV.

Method	Embedding	LLM	UCC	DTC
Raw	text-embedding-ada-002	gpt-3.5-turbo	0.433	0.556
Hierarchical Nodes[1]			**0.811**	**0.878**
Raw		gpt-4o	0.456	0.556
Hierarchical Nodes[1]			**0.856**	**0.844**
Raw		gpt-4o-mini	0.444	0.556
Hierarchical Nodes[1]			**0.700**	**0.833**
Raw	text-embedding-3-large	gpt-3.5-turbo	0.444	0.533
Hierarchical Nodes[1]			**0.822**	**0.833**
Raw	text-embedding-3-small		0.433	0.556
Hierarchical Nodes[1]			**0.811**	**0.822**

1. Hierarchical Nodes indicate that represents hierarchical nodes representation in the header after transposing the table.

As a result, (shown in Figure 10) the table retriever demonstrated optimal performance when integrated into the RAG system with data transpose and headers represented in the hierarchical node representation. This outcome remained consistent even when the embedding model and LLMs varied. Additionally, vector database was more accurate with a DTC structure than UCC, consistent with the patterns observed in Types I and II. These findings provide meaningful evidence of the robustness of the proposed method in this study.

These results can be summarized as follows. Our approach demonstrates superior performance compared to the baseline model using tables without transformation. The dataset used in this study consists predominantly of tables in which complex information is embedded in the indices. By simply

transposing the index and header, this study shows that table retriever performance can be significantly improved by categorizing tables into two distinct types. Notably, for tables requiring domain-specific knowledge, structuring the data into hierarchical nodes and representing them as headers results in the highest accuracy. Our method proved to be highly effective in representing and retrieving complex tables by leveraging the table headers alone.

Furthermore, it outperforms existing approaches that incorporate all table values, thereby providing empirical evidence to support the hypothesis that headers are a key factor in enhancing table-retrieval performance. These findings collectively suggest that the DTC structure, in which information, metadata, and vectors are stored in separate collections according to the type of information, is superior to UCC for vector database when performing table-retrieval tasks. As the structure of the information extracted from text and tables can differ significantly, data separation and storage using the DTC structure is effective in improving the accuracy of table-specific responses.

V. CONCLUSION

In this study, we hypothesize that headers play a more significant role than indices when a table retrieves information from tables and propose a novel approach for tabular data applications that incorporates the representation of hierarchical nodes within table headers. This method utilizes a structured organization of table headers to enhance the retrieval and interpretation of complex tables. Domain-specific knowledge is encoded as hierarchical nodes, allowing table retrieval without requiring the division of data into smaller chunks. Importantly, the representation of hierarchical nodes in headers effectively addresses tables requiring intricate domain knowledge, and mitigates hallucination risks in LLM-generated outputs. This method seeks to streamline complex tables with diverse structures and formats.

The results of this study are as follows. The proposed method consistently outperformed the baseline across Table types I and II, particularly when employing the DTC vector database structure instead of UCC. The evaluation process incorporated variations in embedding models and LLMs to ensure robustness, confirming the advantage of positioning critical table information in headers. For more complex tables requiring domain-specific knowledge (Table types III and IV), where key content is frequently embedded within indices, the same strategy was applied—transposing data and converting index information into hierarchical nodes designated as headers. The results, presented in Figure 10, demonstrate optimal performance under this configuration, aligning with the findings for types I and II (Figure 9). The hierarchical node presentation for headers facilitates more accurate retrieval by relying solely on the headers, surpassing even methods that incorporate the entire table content. Overall, the proposed approach substantially improves table retrieval accuracy, particularly when using DTC, which organizes information, metadata, and vectors into distinct categories. The findings underscore the importance of high-quality header representation for effective RAG-based table retrieval and highlight the structural advantages of DTC over UCC. These results provide empirical support for the hypothesis that transposing table data and employing hierarchical node information for headers contribute to enhanced knowledge retrieval performance.

A limitation of this study is that, while various table types were considered, the dataset used was relatively small. Further validation is necessary to evaluate the effectiveness of the proposed method on larger-scale datasets.

Future work will focus on developing techniques for automatically identifying table headers in large-scale datasets. Additionally, we analyze how the relationship between short text passages associated with tables and their headers can enhance the performance of RAG.

REFERENCES

[1] P. Zhao, H. Zhang, Q. Yu, Z. Wang, Y. Geng, F. Fu, L. Yang, W. Zhang, J. Jiang, and B. Cui, "Retrieval-augmented generation for AI-generated content: A survey," 2024, *arXiv:2402.19473*.

[2] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2022, pp. 1–14.

[3] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, "On faithfulness and factuality in abstractive summarization," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, Jul. 2020, pp. 1906–1919.

[4] J. Kasai, K. Sakaguchi, Y. Takahashi, R. Le Bras, A. Asai, X. Yu, D. Radev, N. A. Smith, Y. Choi, and K. Inui, "RealTime QA: What's the answer right now?" 2022, *arXiv:2207.13332*.

[5] B. Saha, U. Saha, and M. Z. Malik, "QuIM-RAG: Advancing retrieval-augmented generation with inverted question matching for enhanced QA performance," *IEEE Access*, vol. 12, pp. 185401–185410, 2024.

[6] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," 2023, *arXiv:2312.10997*.

[7] K. Kim, M. Kim, H. Lee, S. Park, Y. Han, and B. K. Jeon, "THoRR: Complex table retrieval and refinement for RAG," in *Proc. Workshop Inf. Retrieval's Role RAG Syst. (IR-RAG)*, Washington, DC, USA, Jul. 2024, pp. 1–6.

[8] J. Pasaribu, N. Yudistira, and W. F. Mahmudy, "Tabular data classification and regression: XGBoost or deep learning with retrieval-augmented generation," *IEEE Access*, vol. 12, pp. 191719–191732, 2024.

[9] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi, "When not to trust language models: Investigating effectiveness of parametric and non-parametric memories," in *Proc. Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2023, pp. 1–19.

[10] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, and A. Oprea, "Extracting training data from large language models," presented at the USENIX Security Symp., 2021.

[11] Z. He, Z. Zhong, T. Cai, J. D. Lee, and D. He, "REST: Retrieval-based speculative decoding," 2023, *arXiv:2311.08252*.

[12] L. Zhang, K. Jijo, S. Setty, E. Chung, F. Javid, N. Vidra, and T. Clifford, "Enhancing large language model performance to answer questions and extract information more accurately," 2024, *arXiv:2402.01722*.

[13] M. Fateen, B. Wang, and T. Mine, "Beyond scores: A modular RAG-based system for automatic short answer scoring with feedback," *IEEE Access*, vol. 12, pp. 185371–185385, 2024.

[14] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 27730–27744.

[15] Y.-C. Lin, A. Kumar, N. Chang, W. Zhang, M. Zakir, R. Apte, H. He, C. Wang, and J.-S. Roger Jang, "Novel preprocessing technique for data embedding in engineering code generation using large language model," 2023, *arXiv:2311.16267*.
[16] X. Wen, H. Zhang, S. Zheng, W. Xu, and J. Bian, "From supervised to generative: A novel paradigm for tabular deep learning with large language models," in *Proc. 30th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2024, pp. 3323–3333.
[17] S. Hegselmann, A. Buendia, H. Lang, M. Agrawal, X. Jiang, and D. Sontag, "TabLLM: Few-shot classification of tabular data with large language models," in *Proc. AISTATS*, 2023, pp. 5549–5581.
[18] Z. Wang and J. Sun, "TransTab: Learning transferable tabular transformers across tables," in *Proc. NeurIPS*, 2022, pp. 1–14.
[19] T. Dinh, Y. Zeng, R. Zhang, Z. Lin, M. Gira, S. Rajput, J. Sohn, D. Papailiopoulos, and K. Lee, "LIFT: Language-interfaced fine-tuning for non-language machine learning tasks," in *Proc. NeurIPS*, 2022, pp. 1–22.
[20] N. Hollmann, S. Müller, K. Eggensperger, and F. Hutter, "TabPFN: A transformer that solves small tabular classification problems in a second," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2023, pp. 1–37.
[21] S. Müller, N. Hollmann, S. Pineda Arango, J. Grabocka, and F. Hutter, "Transformers can do Bayesian inference," 2021, *arXiv:2112.10510*.
[22] T. Zhang, X. Yue, Y. Li, and H. Sun, "TableLlama: Towards open large generalist models for tables," 2023, *arXiv:2311.09206*.
[23] Y. Yang, Y. Wang, G. Liu, L. Wu, and Q. Liu, "UniTabE: A universal pretraining protocol for tabular foundation model in data science," in *Proc. ICLR*, 2024, pp. 1–16.
[24] J. Herzig, T. Müller, S. Krichene, and J. Martin Eisenschlos, "Open domain question answering over tables via dense retrieval," 2021, *arXiv:2103.12011*.
[25] F. Pan, M. Canim, M. Glass, A. Gliozzo, and P. Fox, "CLTR: An end-to-end, transformer-based system for cell level table retrieval and table question answering," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process., Syst. Demonstrations*, vol. 2021, pp. 202–209.
[26] F. Pan, M. Canim, M. Glass, A. Gliozzo, and J. Hendler, "End-to-end table question answering via retrieval-augmented generation," 2022, *arXiv:2203.16714*.
[27] A. H. Li, P. Ng, P. Xu, H. Zhu, Z. Wang, and B. Xiang, "Dual reader-parser on hybrid textual and tabular evidence for open domain question answering," in *Proc. ACL/IJCNLP*, 2021, pp. 1–15.
[28] Z. Jiang, Y. Mao, P. He, G. Neubig, and W. Chen, "OmniTab: Pretraining with natural and synthetic data for few-shot table-based question answering," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics (NAACL)*, 2022, pp. 1–11.
[29] W. Zhong, J. Huang, Q. Liu, M. Zhou, J. Wang, J. Yin, and N. Duan, "Reasoning over hybrid chain for table-and-text open domain question answering," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2022, pp. 1–7.
[30] J. Jiang, K. Zhou, Z. Dong, K. Ye, W. X. Zhao, and J. R. Wen, "StructGPT: A general framework for large language model to reason over structured data," in *Proc. Empirical Methods Natural Language Process. (EMNLP)*, 2023, pp. 1–15.
[31] A. S. Sundar and L. Heck, "CTBLS: Augmenting large language models with conversational tables," 2023, *arXiv:2303.12024*.
[32] S. Roychowdhury, M. Krema, A. Mahammad, B. Moore, A. Mukherjee, and P. Prakashchandra, "ERATTA: Extreme RAG for table to answers with large language models," 2024, *arXiv:2405.03963*.
[33] L. Limongi, F. Martini, T. Ha Dao, A. Gaggero, H. Hasnaoui, I. Lopez-Gonzalez, F. Chiarello, F. de Matteis, A. Quaranta, A. Salamon, F. Mattioli, M. Bernard, and M. Lobino, "Linearly multiplexed photon number resolving single-photon detectors array," 2024, *arXiv:2408.12345*.
[34] L. Zha et al., "TableGPT: Towards unifying tables, nature language and commands into one GPT," 2023, *arXiv:2307.08674*.
[35] M. A. Greenwood, "Open-domain question answering," Ph.D. dissertation, Dept. Comput. Sci., Univ. Sheffield, Sheffield, U.K., 2005.
[36] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Trans. Assoc. Comput. Linguistics*, vol. 12, pp. 157–173, Feb. 2024.
[37] Q. Wang, L. I. Wu, and G. Li, "PIRTRE-C: A two-stage retrieval and reranking enhanced framework for improving Chinese psychological counseling," in *Proc. IEEE Int. Conf. Med. Artif. Intell. (MedAI)*, Nov. 2024, pp. 337–347.
[38] J. Shin, R. Aleithan, H. Hemmati, and S. Wang, "Retrieval-augmented test generation: How far are we?" 2024, *arXiv:2409.12682*.
[39] S. Shahriar, B. D. Lund, N. R. Mannuru, M. A. Arshad, K. Hayawi, R. V. K. Bevara, A. Mannuru, and L. Batool, "Putting GPT-4o to the sword: A comprehensive evaluation of language, vision, speech, and multimodal proficiency," *Appl. Sci.*, vol. 14, no. 17, p. 7782, Sep. 2024.
[40] H. N. Patel, A. Surti, P. Goel, and B. Patel, "A comparative analysis of large language models with retrieval-augmented generation based question answering system," in *Proc. 8th Int. Conf. I-SMAC (IoT Social, Mobile, Anal. Cloud) (I-SMAC)*, Oct. 2024, pp. 792–798.
[41] V. Karpukhin, B. Oguz, S. Min, P. S. H. Lewis, L. Wu, S. Edunov, D. Chen, and W. Yih, "Dense passage retrieval for open-domain question answering," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Nov. vol. 2020, pp. 6769–6781.
[42] R. C. Barron, V. Grantcharov, S. Wanna, M. E. Eren, M. Bhattarai, and N. Solovyev, "Domain-specific retrieval-augmented generation using vector stores, knowledge graphs, and tensor factorization," in *Proc. Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2024, pp. 1669–1676.
[43] B. Sarmah, D. Mehta, B. Hall, R. Rao, S. Patel, and S. Pasquali, "HybridRAG: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction," in *Proc. 5th ACM Int. Conf. AI Finance*, 2024, pp. 608–616.
[44] Y. Yang, H. Wu, T. Wang, J. Yang, H. Ma, and G. Luo, "Pseudo-knowledge graph: Meta-path guided retrieval and in-graph text for RAG-equipped LLM," 2025, *arXiv:2503.00309*.

MINCHAE SONG was born in Seoul, Republic of Korea. She received the B.A. and M.A. degrees in economics and the Ph.D. degree in big data analysis from Ewha Womans University, in 2007, 2010, and 2019, respectively. From 2010 to 2014, she was with the National Economic Research Institute. Since 2019, she has been with the Financial Research Institution, NongHyup Financial Group. She has expertise in artificial intelligence, natural language processing, and financial data science. Her research interests include NLP, LLMs, financial analysis, and digital finance.