TabRAG: Improving Pixel-Based Tabular Document Question Answering for Retrieval Augmented Generation via Structured Representations

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) has seen rapid adoption in real-world applications, many of which rely heavily on complex tabular documents. However, existing parsing-based RAG pipelines are often dependent on linear text serialization, which can fail to preserve the two-dimensional structural semantics of tabular data. In this work, we present TABRAG, a parsing-based RAG framework that improves pixed-based tabular document question answering via structured representations. Specifically, our approach first decomposes the input document into a series of components, enabling fine-grained extraction. Table components are then extracted into a hierarchically structured representation by a vision-language model. To further supplement in-domain knowledge, we integrate an agentic self-generated in-context learning module that guides the table extraction process. Experimental results establish TABRAG as a practical industry-level RAG framework, outperforming existing popular parsing techniques across a broad suite of evaluation and ablation benchmarks.

1 Introduction

Retrieval-Augmented Generation (RAG) has become an established framework that enhances large language models by grounding their generation in external knowledge sources (Lewis et al., 2021; Guu et al., 2020; Wei et al., 2025b), leading to rapid adoptions in industrial settings such as personalized enterprise search (OpenAI, 2024; Cohere, 2024). Many of these knowledge sources are from pixel-based documents that contain large, complex tables (van Breugel and van der Schaar, 2024; Hollmann et al., 2023; Si et al., 2024; Arik and Pfister, 2020; Si et al., 2025). Despite the impressive capabilities of RAG on plain text that follows a natural sequence, RAG for tabular documents faces a challenge: it has to be reliable in retrieval and generate factually accurate answers from a two-dimensional layout, where the value of each cell depends on both its row and column context.

Existing RAG approaches mainly fall into four categories: i) table understanding, ii) vision-based retrieval, iii) text-based agentic reasoning and iv) traditional industry-standard methods. Table understanding methods, such as Chain-of-Table (Wang et al., 2024) and TableRAG (Yu et al., 2025b), must assume already-structured inputs (HTML, CSV, SQL). Vision-based methods, such as VisRAG (Yu et al., 2025a) and ColPali (Faysse et al., 2025), utilize VLMs to produce visual document embeddings to improve retrieval embedding quality, but lack the granularity required for fine-grained structured generation. Agentic reasoning frameworks such as Self-RAG (Asai et al., 2023) and InstructRAG (Wei et al., 2025b) improve retrieval quality through self-reflection and synthesized rationales during inference time, but these methods currently operate purely on text. Traditional industry-standard RAG frameworks like LlamaIndex (Liu, 2022) and Langchain (Chase, 2022) serialize the retrieved tables into linear text before provided as in-context learning examples which fails to preserve relational two-dimensional dependencies.

In this work, we propose TABRAG, a pipeline built to improve upon traditional industry-standard RAG frameworks for pixed-based tabular document question answering (QA) tasks, through the use of structured representations. We leverage a layout segmentation module that decomposes the document into components for fine-grained parsing, a vision language model (VLM) that parses the components while extracting structured representations from tables, and an autonomous in-context learning module that self-generates exemplars, serving as supervision by facilitating domain knowledge for the VLM. In summary, our contributions are as follows:

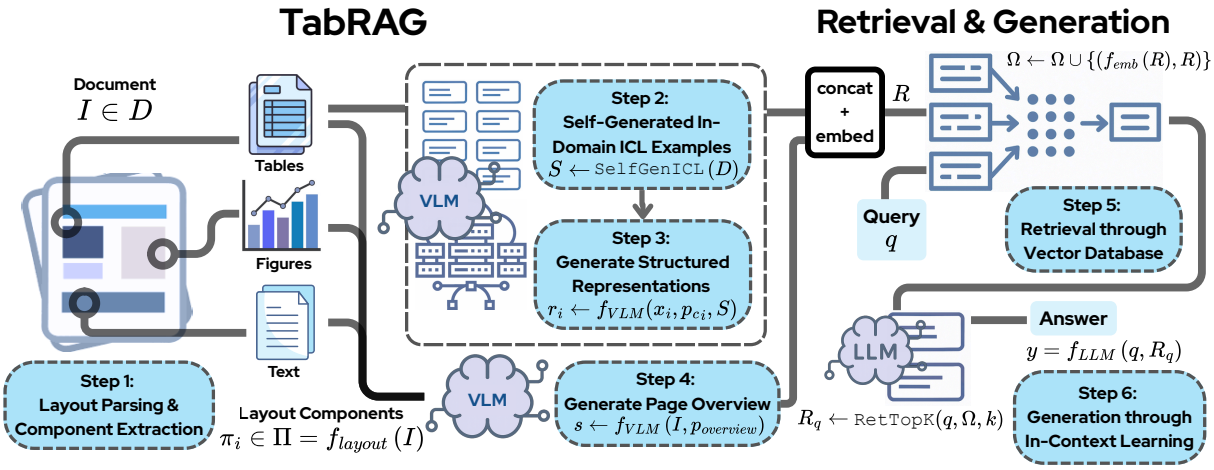


Figure 1: The TABRAG Architecture, a parsing-based RAG pipeline designed specifically for tables. A layout detection model is applied to segment various components from the documents. Specifically, the tables are then passed into a vision language model, which extracts cell values along with their corresponding column and row names in a structured representation. The structured representation is then coupled with self-generating in-context learning examples that guide its output.

- We introduce TABRAG, an end-to-end RAG pipeline tailored for pixel-based tabular visual documents. TABRAG's architecture addresses the limitations of conventional text-based parsing pipelines for tabular reasoning and document QA. An overview of our architecture is illustrated in Figure 1.
- We propose a parsing framework that combines layout segmentation, vision-language models, structured representations, and self-generated in-context learning, enabling robust extraction across diverse table layouts and formats while preserving two-dimensional table semantics.
- We demonstrate the strong performance of TABRAG in our experiments against parsing baselines across various datasets and benchmarks. This highlights our ability to outperform existing methods in generation and retrieval-to-generation tasks while remaining competitive for retrieval.

2 Method

In this section, we introduce the TABRAG framework. We start by decomposing documents into focused components (§2.1). This is followed by the design of TABRAG's structured representation (§2.2). We then augment our structured representation with an autonomous self-generated in-context learning (ICL) examples (§2.3). We conclude with an overview of the full retrieval and generation pipeline (§2.4). Pseudocode of the TabRAG framework can be found in Algorithm 1.

We supplement our methods section with additional details in the Appendix: a qualitative example of our pipeline in A.3; prompts for TABRAG's self-generated ICL examples and TABRAG's structured representations in C.1 and C.2; TABRAG's outputs in D.1 and D.2.

2.1 Layout Segmentation

Layout segmentation identifies and segments visually coherent regions to preserve the document's structural organization. Given an input document page I, we leverage a layout detection model f_{layout} based on the Document Image Transformer (DIT) (Li et al., 2022), a self-supervised transformer model for document image understanding, which is trained on large-scale tabular datasets such as PubLayNet (Zhong et al., 2019). The purpose of the layout model is to predict bounding box coordinates and labels for distinguishable semantic components within the document. More formally,

$$\Pi \leftarrow f_{\text{layout}}(I), \quad \text{where } \Pi = \{\pi_i\}_{i=1}^{M},\ \pi_i = (b_i, x_i, c_i). \tag{1}$$

Each individual component π is represented by a tuple, consisting of its bounding box coordinates (b_i), the cropped image (x_i), and component type (c_i). The component type $c_i \in \{0, 1, 2, 3\}$ maps to a semantic label in the set $\{\texttt{title}, \texttt{text}, \texttt{figure}, \texttt{table}\}$. By performing a layout-aware segmentation into finer-grained components, TABRAG explicitly allows for localized

Algorithm 1 TABRAG Framework

Input: Dataset $\mathcal{D} = \{I_j\}_{j=1}^{N}$, Query q, No. of Documents to Retrieve k
Output: Response y
$\mathcal{S} \leftarrow$ SELFGENICL($\mathcal{D}$) {self-generated ICL module; see Alg. 2}
$\Omega \leftarrow \emptyset$ {initialize vector store}
for all $I \in \mathcal{D}$ **do**
  $\mathcal{R} \leftarrow \emptyset$ {parsed representations}
  $\Pi \leftarrow f_{\text{layout}}(I)$ {where $\Pi = \{\pi_i\}_{i=1}^{M}$}
  for all $\pi_i = (b_i, x_i, c_i) \in \Pi$ **do** {$\pi_i = (b_i, x_i, c_i)$ = (bbox, cropped img, component type)}
    if c_i == `table` **then**
      $r_i \leftarrow f_{\text{VLM}}(x_i, p_{c_i}, \mathcal{S})$ {parse table with ICL supervision}
    else
      $r_i \leftarrow f_{\text{VLM}}(x_i, p_{c_i})$ {parse non-table component}
    end if
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{r_i\}$
  end for
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{f_{\text{VLM}}(I, p_{page})\}$ {page-level overview}
  $\Omega \leftarrow \Omega \cup \{(f_{\text{emb}}(\mathcal{R}), \mathcal{R})\}$ {embed document and add to vector store}
end for
$\mathcal{R}_q \leftarrow$ RETRIEVETOPK(q, Ω, k) {retrieve top-k documents}
$y \leftarrow f_{\text{LLM}}(q, \mathcal{R}_q)$ {generate response}

reasoning and addresses several key limitations of page-level methods. This is because when an entire page is represented as a single visual embedding, global attention blurs fine-grained dependencies (Liu et al., 2023) that are especially semantically crucial in tables, such as column alignments, header associations, and cell-to-cell relations. A nice byproduct of layout segmentation is that it also reduces the effective context length, allowing the input to fit within the context window of common VLMs without major performance drawbacks.

In contrast to standard document understanding models, such as LayoutLMv3 (Huang et al., 2022) and DocFormer (Appalaraju et al., 2021), that use spatial embeddings to correlate text with 2D coordinates and layout semantics, TABRAG models these dependencies explicitly through region-level segmentation, embedding the structural relationships directly into the representation. This allows TABRAG to establish a more faithful and structurally grounded foundation for downstream multimodal processing.

2.2 Structured Representation

Once the document is decomposed into localized regions, each segment is semantically interpreted using a vision language model. For each component π_i, we leverage a VLM f_{VLM} with a component-specific prompt p_{c_i} to translate the visual structure of the extracted image x_i into a concise and structured representation of each region:

$$r_i \leftarrow f_{\text{VLM}}(x_i, p_{c_i}). \quad (2)$$

The VLM is instructed to extract the text and title directly, while providing detailed descriptions for figures. Specifically, for tables, the VLM acts as an "information extraction engine" (Kong et al., 2024) tasked with extracting table cells in the input image into JSON representations. In the resulting representation, each cell must be represented with its corresponding row name, column name, value, and unit. Additional rules are also defined to address punctuation, symbol-number normalizations, table formatting, and empty cells.

In case the layout model fails to capture a coherent representation for a particular component, we introduce a global context module where the VLM is applied to the document image: $f_{\text{VLM}}(I, p_{\text{page}})$, effectively preventing segmentation-related failures from propagating through the remainder of the pipeline. This full-page inference provides a coarse global description that captures the overall content, even when region extraction is unreliable.

2.3 Self-Generated In-Context Learning

In-context learning has been established as an emergent property of large autoregressive language models (Brown et al., 2020). Language models demonstrate reasoning abilities from few-shot or chain-of-thought demonstrations (Wei et al., 2023; Wang et al., 2023), improving overall coherence in their outputs. Considering that tables are formatted in varying complex styles (see Appendix B.1), supervision is necessary to guide the VLM in its extraction. While VLM can directly leverage ICL via example images, we convert these examples to text as image tokenization is computationally expen-

Algorithm 2 Self-Generated In-Context Learning

Input: Dataset $\mathcal{D}$, VLM f_{VLM}, tokenizer f_{tok}, target examples K=3, candidate pages M=20, min/max tokens $L_{\min}/L_{\max}$
Output: Selected ICL examples $\mathcal{S}$
$\mathcal{I} \leftarrow \textsc{CollectPage}(\mathcal{D})$ {accumulate document pages in the dataset}
$\tilde{\mathcal{I}} \leftarrow \textsc{Shuffle}(\mathcal{I})$ {shuffle to ensure fairness}
$\mathcal{T}, \mathcal{S} \leftarrow \emptyset, \emptyset$ {store largest tables of all $\tilde{\mathcal{I}}$ and ICL examples}
for all $I \in \tilde{\mathcal{I}}$ **and** $|\mathcal{T}| < M$ **do**
  $\Pi \leftarrow f_{\text{layout}}(I)$ {where $\Pi = \{\pi_i\}_{i=1}^{M}$}
  $\Pi^{\text{tbl}} \leftarrow \{\pi_i \in \Pi \mid c_i == \texttt{table}\}$ {$\pi_i = (b_i, x_i, c_i)$ = (bbox, cropped img, component type)}
  if $\Pi^{\text{tbl}} \neq \emptyset$ **then**
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\arg\max_{\pi \in \Pi^{\text{tbl}}} \textsc{Area}(b_i)\}$ {extract largest table from the extracted tables}
  end if
end for
$\mathcal{T} \leftarrow \textsc{SortByLargestArea}(\mathcal{T})$
for all $x \in \mathcal{T}$ **and** $|\mathcal{S}| < K$ **do** {generate ICL examples from the top K largest tables}
  $m \leftarrow f_{\text{VLM}}(x, p_{\text{markdown}})$ {markdown structured representation}
  $j \leftarrow f_{\text{VLM}}(x, p_{\text{json}})$ {json structured representation}
  $L \leftarrow f_{\text{tok}}(m) + f_{\text{tok}}(j)$
  if $L_{\min} \leq L < L_{\max}$ **then** {ensure validity of representation}
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{(m, j)\}$
  end if
end for
return $\mathcal{S}$

sive and severely constrains the context window. Inspired by recent agentic workflows (Yao et al., 2022; Shinn et al., 2023), we design a VLM-based autonomous module that self-generates text-based ICL examples, supplementing our prompt for extracting structured table representations.

We formalize this process as a multi-stage selection and generation pipeline. Given a document dataset $\mathcal{D}$, we first randomly extract a subset of document pages $\tilde{\mathcal{I}}$. For each page, we apply a layout segmentation model f_{layout} to extract layout components Π. We then filter for elements identified as tables:

$$\Pi^{\text{tbl}} \leftarrow \{\pi_i \in \Pi \mid c_i = \texttt{table}\}. \tag{3}$$

To ensure that the generated ICL examples are sufficiently complex, we select the largest table from each page based on its bounding box area: $\arg\max_{\pi \in \Pi^{\text{tbl}}} \textsc{Area}(b_i)$. We collect the M candidate tables with the largest areas and process each table with a VLM to generate dual structured representations: a markdown string m and a JSON object j:

$$m \leftarrow f_{\text{VLM}}(x, p_{\text{markdown}}), \tag{4}$$

$$j \leftarrow f_{\text{VLM}}(x, p_{\text{json}}). \tag{5}$$

To ensure the validity of the demonstration, we compute the total token length L:

$$L \leftarrow f_{\text{tok}}(m) + f_{\text{tok}}(j), \tag{6}$$

and enforce a constraint such that L is at least $L_{\min}$ and at most $L_{\max}$ tokens in length: $L_{\min} \leq L < L_{\max}$. The markdown and JSON are then concatenated to form the final input-output pairs and accumulated in $\mathcal{S}$ until it reaches the desired K target ICL examples. $\mathcal{S}$ can now be used as input to the VLM when generating the structured representations:

$$r_i \leftarrow \begin{cases} f_{\text{VLM}}(x_i, p_{c_i}, \mathcal{S}) & \text{if } c_i = \texttt{table} \\ f_{\text{VLM}}(x_i, p_{c_i}) & \text{otherwise} \end{cases} \tag{7}$$

Our self-generated ICL algorithm and its recommended hyperparameters can be found in Algorithm 2. This filtering mechanism ensures that the selected set $\mathcal{S}$ consists of K high-fidelity examples that fit within the model's context window. Most importantly, the algorithm preserves both structural layout complexities and domain-specific semantics.

2.4 Retrieval and Generation

After generating representations across the entire dataset, we embed $\mathcal{R}$ into a high-dimensional latent space, $f_{emb}(\mathcal{R})$ (Zhang et al., 2025), and index them within a vector store, Ω (Douze et al., 2025). At inference time, a similarity search retrieves the k documents most relevant to the query q. The retrieved documents then serve as in-context examples for an LLM to generate the final answer.

3 Experiments

In the following, we provide preambles regarding our experiments. Further details can be found in Appendix A and B.

Models	Accuracy (%) ↑						
	TAT-DQA	MP-DocVQA	WikiTQ	TableVQA	ComTQA	FinQA	Average
PyMuPDF	66.83±.000	N/A	59.49±.004	N/A	N/A	57.02±.002	61.11
PyTesseract	62.01±.002	59.73±.002	49.32 ±.005	51.88±.001	40.90±.001	53.73±.001	52.93
Deepseek-OCR	64.45±.002	73.47±.003	60.40±.002	68.57±.002	53.30±.005	47.19±.003	61.23
Gemma-3-27B-it	43.92±.002	69.08±.003	56.75±.002	64.75±.003	51.70±.002	55.15±.001	56.89
Qwen3-VL-8B-Instruct	66.26±.001	85.81±.002	61.71±.005	66.29±.002	67.80±.001	66.07±.002	68.99
Qwen3-VL-32B-Instruct	66.56±.003	86.32±.003	65.95±.007	66.07±.001	71.30±.002	64.15±.003	70.06
GPT-5.2	68.54±.001	86.26±.001	62.23±.002	67.40±.002	73.90±.003	69.98±.001	71.39
TabRAG$_{8B}$	92.56±.002	88.87±.003	68.49±.003	70.60±.004	**82.60±.002**	71.55±.001	79.11
TabRAG$_{32B}$	**92.70±.002**	**88.93±.002**	**71.68±.004**	**71.20±.001**	75.80±.002	**78.66±.001**	**79.83**

Table 1: Overall generation performance in accuracy (%). We assume that the query corresponds to the ground truth document. In an in-context learning setting, we query an LLM to use the representations synthesized by different parsing methods to answer the question and evaluate with exact-match accuracy.

3.1 Setup

TabRAG employs Qwen3-VL-8B/32B-Instruct (TabRAG$_{8B/32B}$) as the VLM backbone (Team, 2025b). We use Qwen3-Embedding-8B (Zhang et al., 2025) as the embedding model for retrieval and store the resulting embeddings in a FAISS vector store (Douze et al., 2025). Finally, we use a Qwen3-8B (No Think) language model as the downstream query engine. In the main text, we focus on TabRAG$_{8B}$ for our ablation studies. Experimental results are averaged over 3 evaluation seeds. Document pages used for self-generated ICL examples are excluded from evaluation.

Datasets. We evaluate TabRAG across document question-answering benchmarks that include tables. These include: TAT-DQA (Zhu et al., 2022), MP-DocVQA (Tito et al., 2023), WikiTQ (Pasupat and Liang, 2015), TableVQA (Kim et al., 2024), ComTQA (Zhao et al., 2024) and T^2-RAGBench (FinQA) (Strich et al., 2025). To ensure consistency across modalities, all multi-page documents are preprocessed into single-page units and converted into high-resolution PNG images for methods requiring visual input.

Baselines. We compare TabRAG against existing off-the-shelf document RAG extraction techniques: PyMuPDF (McKie, 2025), PyTesseract (Hoffstaetter, 2025), Deepseek-OCR (Wei et al., 2025a), Gemma-3-27B-it (Team, 2025a), Qwen3-VL-8B/32B-Instruct (Team, 2025b) and GPT-5.2 (OpenAI, 2025). These baselines represent rule-based parsing, OCR-based extraction, and transformer-based VLMs. For our baseline VLM prompt (in Appendix C.3), we follow the standard prompting technique for parsing used by LlamaIndex (Liu, 2022). To investigate whether stronger baseline prompts can improve reasoning over complex tabular layouts, we additionally evaluate a more elaborate "complex" prompt designed by GPT-5.4. Our ablation study in Table 2 shows that this prompt underperforms in terms of average generation accuracy. See Appendix E.3 for the full experiment.

Models	Average Accuracy (%) ↑
Qwen3-VL-8B-Instruct	68.99
Qwen3-VL-8B-Instruct (Complex)	48.41

Table 2: Average generation accuracy (%) of baseline prompts. We compare a LlamaIndex prompt (Liu, 2022) against a "complex" prompt designed by GPT-5.4

Benchmarks. To evaluate generation and retrieval-to-generation performance, we use exact-match accuracy (excluding whitespace, punctuation, and capitalization), which determines whether the ground-truth answers are included in the model generations (Wei et al., 2025b; Mallen et al., 2023; Schick et al., 2023). For retrieval, we use mean reciprocal rank (MRR@10) (Yu et al., 2025a; Masry et al., 2022; Methani et al., 2020), computed as the inverse of the rank of the first relevant document among the top-10 retrieved candidates, and averaged across all queries, assigning higher scores to relevant documents retrieved at earlier positions.

3.2 Baseline Performance

Generation. We evaluate TabRAG on downstream generation tasks to isolate the impact of different document parsing methods on answer quality. To assess generation performance independently, we use an oracle retrieval setting in which the ground-truth document corresponding to each query is provided. This setup ensures that differences in accuracy are attributable to the document's structural representation rather than retrieval noise. Specifically, we use these ground-truth documents to construct the different representations, which then serve as context for the LLM query engine. In

Models	MRR@10 ↑						
	TAT-DQA	MP-DocVQA	WikiTQ	TableVQA	ComTQA	FinQA	Average
PyMuPDF	75.60$_{\pm.001}$	N/A	**86.74**$_{\pm.001}$	N/A	N/A	70.69 $_{\pm.001}$	77.68
PyTesseract	**75.95**$_{\pm.001}$	83.77$_{\pm.001}$	82.23$_{\pm.003}$	85.48$_{\pm.004}$	81.36$_{\pm.002}$	72.45$_{\pm.002}$	80.21
DeepSeek-OCR	73.66$_{\pm.002}$	84.05$_{\pm.002}$	82.01$_{\pm.003}$	88.72$_{\pm.002}$	80.25$_{\pm.005}$	67.88$_{\pm.001}$	79.43
gemma-3-27b-it	71.31$_{\pm.004}$	84.48$_{\pm.001}$	85.81$_{\pm.002}$	88.75$_{\pm.005}$	80.77$_{\pm.002}$	68.72$_{\pm.002}$	79.97
Qwen3-VL-8B-Instruct	73.88$_{\pm.002}$	84.59$_{\pm.002}$	86.20$_{\pm.001}$	88.57$_{\pm.002}$	82.34$_{\pm.001}$	70.98$_{\pm.001}$	81.09
Qwen3-VL-32B-Instruct	73.95$_{\pm.002}$	84.69$_{\pm.003}$	86.24$_{\pm.001}$	88.16$_{\pm.001}$	**82.59**$_{\pm.001}$	**71.81**$_{\pm.001}$	81.24
GPT-5.2	74.23$_{\pm.002}$	84.69$_{\pm.003}$	86.08$_{\pm.004}$	**90.38**$_{\pm.003}$	81.54$_{\pm.003}$	71.27$_{\pm.002}$	**81.37**
TabRAG$_{8B}$	72.30$_{\pm.003}$	**85.61**$_{\pm.001}$	84.07$_{\pm.004}$	88.15$_{\pm.003}$	80.01$_{\pm.003}$	71.35$_{\pm.001}$	80.25
TabRAG$_{32B}$	72.77$_{\pm.004}$	85.60$_{\pm.002}$	83.58$_{\pm.002}$	89.54$_{\pm.001}$	78.77$_{\pm.003}$	70.88$_{\pm.002}$	80.19

Table 3: Overall retrieval performance in mean reciprocal rank (MRR@10). Given a query, the model searches over ten indexed document representations and ranks them based on semantic similarity.

Models	Accuracy (%) ↑						
	TAT-DQA	MP-DocVQA	WikiTQ	TableVQA	ComTQA	FinQA	Average
PyMuPDF	55.62$_{\pm.001}$	N/A	53.33$_{\pm.003}$	N/A	N/A	37.77$_{\pm.003}$	48.91
PyTesseract	55.06$_{\pm.002}$	57.63$_{\pm.001}$	36.59$_{\pm.003}$	54.40$_{\pm.002}$	35.80$_{\pm.002}$	52.83$_{\pm.001}$	48.72
DeepSeek-OCR	51.71$_{\pm.002}$	68.32$_{\pm.002}$	54.21$_{\pm.001}$	60.20$_{\pm.001}$	40.50$_{\pm.003}$	42.50$_{\pm.002}$	52.91
Gemma-3-27b-it	38.00$_{\pm.003}$	64.50$_{\pm.001}$	47.36$_{\pm.001}$	58.82$_{\pm.002}$	48.00$_{\pm.003}$	46.86$_{\pm.003}$	50.59
Qwen3-VL-8B-Instruct	54.30$_{\pm.002}$	75.61$_{\pm.001}$	52.84$_{\pm.002}$	59.20$_{\pm.002}$	63.50$_{\pm.002}$	59.43$_{\pm.001}$	60.81
Qwen3-VL-32B-Instruct	54.84$_{\pm.001}$	80.53$_{\pm.002}$	55.38$_{\pm.002}$	62.20$_{\pm.001}$	64.50$_{\pm.001}$	60.11$_{\pm.003}$	62.93
GPT-5.2	68.54$_{\pm.001}$	80.15$_{\pm.002}$	55.38$_{\pm.003}$	60.98$_{\pm.001}$	68.20$_{\pm.001}$	60.60$_{\pm.002}$	65.64
TabRAG$_{8B}$	88.57$_{\pm.002}$	81.30$_{\pm.003}$	**58.50**$_{\pm.002}$	65.00$_{\pm.002}$	**68.30**$_{\pm.001}$	**63.15**$_{\pm.001}$	70.80
TabRAG$_{32B}$	**88.63**$_{\pm 0.01}$	**82.06**$_{\pm.002}$	58.12$_{\pm.001}$	**68.40**$_{\pm.002}$	66.10$_{\pm.002}$	62.20$_{\pm.001}$	**70.92**

Table 4: Overall end-to-end retrieval-to-generation performance in accuracy (%). For each query, we locate the retrieval folder and retrieve the top-3 most similar documents. The retrieved documents and query are passed as inputs to the LLM query engine for answer generation, then evaluated using exact-match accuracy.

an in-context learning (Brown et al., 2020) setting, we prompt the LLM query engine to answer each question using information from the document. As shown in Table 1, TabRAG models exhibit consistent gains over baselines across all benchmarks. By aligning visual, structural, and textual cues before generation, TabRAG enables the language model to reason over documents in a way that preserves both local tabular relationships and global contextual meaning.

Although the baseline parsers can achieve high extraction quality, they are limited by the linearized nature of their outputs. PyMuPDF (McKie, 2025) and PyTesseract (Hoffstaetter, 2025) follow a raster-scan order, while VLMs (Team, 2025b; OpenAI, 2025) often default to textual serialization (Liu, 2022). Both approaches discard the table's spatial hierarchy by collapsing two-dimensional dependencies into a one-dimensional sequence.

Retrieval. We evaluate TabRAG's ability to retrieve the documents most relevant to each query. Given a query, the model searches over the embedded document representations within the vector stores generated by different parsing methods and ranks them based on semantic similarity. We report mean reciprocal rank (MRR@10) as the primary evaluation metric. As shown in Table 3, TabRAG achieves competitive retrieval accuracy compared to existing baselines. We emphasize that our method is primarily designed to improve document understanding and question answering, thereby enhancing generation performance. Therefore, an effective deployment strategy is to combine TabRAG with existing state-of-the-art retrievers and leverage the corresponding TabRAG vector stores during generation.

End-to-End Retrieval-to-Generation. We evaluate TabRAG's end-to-end retrieval-to-generation performance to assess its effectiveness in realistic RAG scenarios. For each query, we locate the retrieval folder and retrieve the top-3 most similar documents via FAISS vector search. The retrieved documents are concatenated and passed as inputs with the query to the LLM query engine for answer generation. The generated answer is then evaluated using exact-match accuracy. As shown in Table 4, TabRAG demonstrates consistent and considerable accuracy improvements over baselines, emphasizing the importance of preserving structured representations for the complete RAG framework.

Computation Cost. While our method introduces several discrete stages, our experiment in

Models	Accuracy (%) ↑						
	TAT-DQA	MP-DocVQA	WikiTQ	TableVQA	ComTQA	FinQA	Average
TabRAG$_{8B}$	**92.56**$_{\pm.002}$	**88.87**$_{\pm.003}$	**68.49**$_{\pm.003}$	**70.60**$_{\pm.004}$	**82.60**$_{\pm.002}$	**71.55**$_{\pm.001}$	**79.11**
– Layout	90.97$_{\pm.002}$	80.15$_{\pm.003}$	63.60$_{\pm.003}$	66.60$_{\pm.002}$	73.30$_{\pm.002}$	68.39$_{\pm.001}$	73.84
– Self-Gen ICL	90.12$_{\pm.001}$	87.41$_{\pm.002}$	62.04$_{\pm.002}$	61.20$_{\pm.000}$	77.40$_{\pm.001}$	67.07$_{\pm.002}$	74.21
– Structured Rep. + Self-Gen ICL	87.26$_{\pm.001}$	87.21$_{\pm.004}$	62.23$_{\pm.003}$	61.20$_{\pm.003}$	70.00$_{\pm.002}$	66.74$_{\pm.001}$	72.44
– Page Overview	91.69$_{\pm.001}$	80.47$_{\pm.004}$	67.73$_{\pm.002}$	67.20$_{\pm.003}$	73.10$_{\pm.003}$	68.41$_{\pm.001}$	74.77

Table 5: Ablation study in a leave-one-out setting of individual TabRAG$_{8B}$ modules on generation accuracy (%). Each row (prefixed with "–") represents the performance of the pipeline after removing a specific component.

Models	Accuracy (%) ↑						
	TAT-DQA	MP-DocVQA	WikiTQ	TableVQA	ComTQA	FinQA	Average
TabRAG$_{JSON}$	**92.56**$_{\pm.002}$	**88.87**$_{\pm.003}$	**68.49**$_{\pm.003}$	**70.60**$_{\pm.004}$	**82.60**$_{\pm.002}$	**71.55**$_{\pm.001}$	**79.11**
TabRAG$_{MD}$	90.48$_{\pm.002}$	88.35$_{\pm.003}$	62.42$_{\pm.002}$	62.00$_{\pm.003}$	75.90$_{\pm.002}$	67.02$_{\pm.002}$	74.36
TabRAG$_{Text}$	90.97$_{\pm.002}$	86.45$_{\pm.003}$	62.03$_{\pm.003}$	68.00$_{\pm.003}$	78.40$_{\pm.002}$	67.83$_{\pm.001}$	75.61

Table 6: Ablation study of structured representation design in TabRAG on generation accuracy (%). We evaluate the performance of our hierarchical JSON schema (TabRAG$_{JSON}$) against standard markdown formatting (TabRAG$_{MD}$) and natural language interpretation (TabRAG$_{TEXT}$).

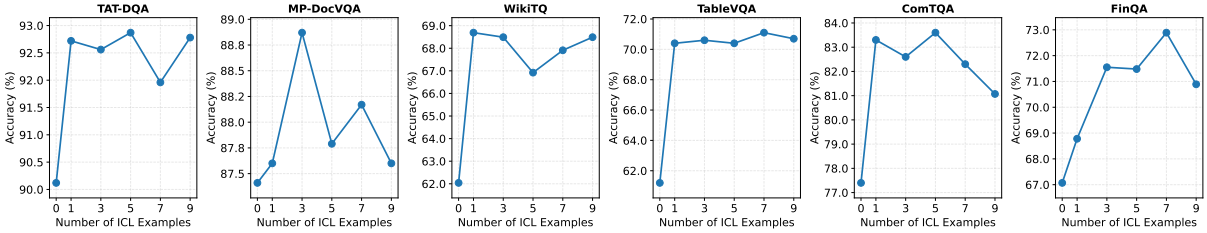


Figure 2: Ablation study of the number of self-generated in-context learning examples (K) in TabRAG on generation accuracy (%). We find three ICL examples ($K = 3$) as the sweet spot for guiding the VLM's extraction.

Appendix E.1 suggests that the tradeoff between an improvement in accuracies against the increase in latency and token use is only marginal.

3.3 Ablation Studies

TabRAG Modules. We investigate the impact of each module in the TabRAG pipeline. In Table 5, we conduct a leave-one-out ablation study on generation performance by removing one module at a time. The ablated modules include layout segmentation, self-generated ICL examples, the combination of structured representations and self-generated ICL examples, and the page overview. Our results show that each component contributes positively to the model's overall reasoning and extraction capabilities, as the full TabRAG configuration consistently outperforms all ablated variants.

We provide *qualitative insights* to support the importance of each component. First, layout segmentation mitigates hallucinations by reducing the input context length for the VLM. Second, self-generated ICL provides contextual knowledge across domains, ranging from financial tables in TAT-DQA to HTML tables in TableVQA. Third, structured representations improve both VLM extraction and query-engine reasoning. Finally, the page overview serves as a fallback when the layout model fails, as observed in MP-DocVQA and TableVQA.

Structured Representation Design. While tables are inherently two-dimensional, LLMs require them to be flattened into a one-dimensional representation before they can be indexed. Standard approaches for parsing tabular documents often rely on markdown or natural language serialization (Team, 2025b; OpenAI, 2025; Liu, 2022; Chase, 2022), which forces a two-dimensional table into a one-dimensional linear text stream. We perform a case study comparing our JSON representation against markdown and text; see Appendix E.2 for details. The results in Table 6 show that the JSON serialization format (TabRAG$_{JSON}$) provides the most robust results. While the JSON representation explicitly links each value to its metadata, the text- and markdown-based generations fail to maintain these relational threads. The visualizations of the "flattened" TabRAG$_{MD}$ and TabRAG$_{text}$ tables indicate that raster-scan parsing prevents the model from synthesizing information across rows and columns, as the associated cell names are parsed much earlier than the cell values.

Number of Self-Gen ICL Demonstrations. While increasing the number of demonstrations in the prompt generally improves the model's task alignment, longer prompts introduce a trade-off with the model's ability to reason over extended contexts (Liu et al., 2023). We analyze the effect of the number of self-generated ICL examples on generation performance. As shown in Figure 2, performance improves sharply when moving from zero to a small number of demonstrations, with three ICL examples consistently yielding stable gains across most datasets. Beyond this point, additional demonstrations provide diminishing returns and, in some cases, lead to slight performance degradation, likely due to increased context saturation.

4 Related Works

RAG for Visually Rich Documents. RAG couples input queries with retrieved relevant documents to produce grounded responses (Lewis et al., 2021; Guu et al., 2020; Wei et al., 2025b). Recent pipelines like VisRAG (Yu et al., 2025a) and ColPali (Faysse et al., 2025) utilize vision-language models to produce document embeddings that incorporated visual and spatial features directly. GraphRAG (Edge et al., 2025) and StructRAG (Li et al., 2024) utilized LLMs to reconstruct information into knowledge graphs or dynamic formats at inference time. While these vision and structure-based retrieval methods improved relevance for retrieval, they often lacked the granularity to preserve specific cell-to-header associations in dense tables for generation. In contrast, TABRAG deterministically maps the 2D hierarchy of a table into a hierarchical structural rationale, allowing for the preservation of explicit spatial-semantic information especially from dense, cell-heavy PDFs.

Table Understanding. Table understanding research has primarily focused on discrete reasoning over data that is already structured. Chain-of-Table (Wang et al., 2024) introduced a dynamic framework for iteratively transforming tables to mimic human thought, while TableRAG (Yu et al., 2025b) utilized SQL-based querying for explicit data retrieval. However, these methods assume a "clean" input, typically in HTML, CSV, or SQL formats, thereby bypassing the fundamental challenge of interpreting raw, unstructured document (PDF) pixels. While UniTable (Peng et al., 2024) unified recognition and extraction into a pixel-to-sequence objective, its focus on table recognition and reliance on linear decoding remains a bottleneck for preserving relational dependencies in large-scale tables. To the best of our knowledge, TABRAG is the only framework offering a complete end-to-end pipeline that specifically targets tabular document QA within RAG frameworks.

Agentic Rationale Generation. Recent approaches in document parsing have moved beyond simple extraction, incorporating agentic rationale generation (Shinn et al., 2023; Yuksekgonul et al., 2024; Xiao et al., 2025) to improve reliability. In the text domain, frameworks like Self-RAG introduced self-reflection tokens to dynamically critique retrieval quality (Asai et al., 2023), while InstructRAG leveraged self-synthesized rationales to explicitly filter noise from retrieved contexts (Wei et al., 2025b). This reasoning-centric paradigm is now influencing visual document understanding, where methods such as Rationale Distillation (Jia et al., 2025) train models to articulate intermediate reasoning steps for complex visual tasks. In this work, TABRAG adopts this philosophy by utilizing VLMs to generate structured JSON representations with agentic self-generated ICLs, ensuring that extracted values are explicitly grounded in their spatial and semantic layout.

5 Conclusion

In this paper, we introduce TABRAG, a retrieval-augmented generation framework for improving pixel-based tabular document question answering through structured representations. Several insights of our work include the need for i) layout segmentation, decomposing documents into components for fine-grained extraction, ii) structured representations, for accurate parsing from the VLM and downstream reasoning of the LLM query engine, iii) self-generated in-context learning examples to guide the parsing process, and iv) page overviews, to prevent layout segmentation failures from propagating through the pipeline. Extensive experiments and comprehensive ablations across multiple tabular-document question-answering datasets demonstrate TABRAG's effectiveness in generation and end-to-end retrieval-to-generation tasks while maintaining competitive retrieval performance. Coupled with state-of-the-art retrievers, TABRAG can help bridge the gap between table understanding and document-level comprehension. We hope this work inspires future studies on more efficient parsing strategies for pixel-based tabular documents.

Limitations and Risks

While our empirical studies demonstrate the effectiveness of TABRAG across various tasks, it also presents several limitations. First, TABRAG's pipeline depends on the reliability of each module. Errors from layout segmentation, visual parsing, retrieval, or the downstream query engine can propagate through the system and affect final answer quality. This could be particularly harmful in high-stakes domains such as financial, legal, medical, or enterprise document analysis. We try to mitigate this by our modular pipeline design, allowing us to easily replace each module with the latest state-of-the-art. In our pipeline, we also try to mitigate failures from the layout segmentation model with our page overview module. Second, the TABRAG modules introduces additional computational and token overhead compared with direct OCR- or VLM-based parsing baselines. While our cost analysis suggests that this overhead is modest relative to the accuracy gains, further work is needed to improve latency and scalability for large document collections. Lastly, TABRAG's structured representation is designed primarily for tabular-heavy visual documents. The same representation may be less effective for documents dominated by figures, charts, handwritten content, complex mathematical notations, or highly irregular layouts. Extending our framework to handle these broader visually rich document types remains an important direction for future work.

References

Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha. 2021. Docformer: End-to-end transformer for document understanding. *arXiv preprint arXiv:2106.11539*.

Sercan O. Arik and Tomas Pfister. 2020. Tabnet: Attentive interpretable tabular learning. *arXiv preprint arXiv:1908.07442*.

Artifex Software, Inc. 2024. Pymupdf4llm: Structured pdf extraction for large language models. https://github.com/pymupdf/PyMuPDF4LLM. Built on PyMuPDF (MuPDF) for LLM-oriented document parsing.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Matthieu Boulanger. 2023. img2table: Table extraction from images and pdfs. https://github.com/xavctn/img2table. Python library for detecting and extracting tables using computer vision and OCR.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Harrison Chase. 2022. Langchain. https://www.langchain.com/.

Cohere. 2024. Enterprise AI: Private, secure, customizable. https://cohere.com/.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2025. The faiss library. *arXiv preprint arXiv:2401.08281*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2025. Colpali: Efficient document retrieval with vision language models. *arXiv preprint arXiv:2407.01449*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Samuel Hoffstaetter. 2025. A python wrapper for google tesseract. Version 0.3.13. https://github.com/madmaze/pytesseract.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. 2023. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. *arXiv preprint arXiv:2204.08387*.

Pengyue Jia, Derong Xu, Xiaopeng Li, Zhaocheng Du, Xiangyang Li, Yichao Wang, Yuhao Wang, Qidong Liu, Maolin Wang, Huifeng Guo, Ruiming Tang, and Xiangyu Zhao. 2025. Bridging relevance and reasoning: Rationale distillation in retrieval-augmented generation. *arXiv preprint arXiv:2412.08519*.

Anthony Kay. 2007. Tesseract: an open-source optical character recognition engine. *Linux Journal*, 2007(159):2.

Yoonsik Kim, Moonbin Yim, and Ka Yeon Song. 2024. Tablevqa-bench: A visual question answering benchmark on multiple table domains. *arXiv preprint arXiv:2404.19205*.

Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, Xin Zhou, Enzhi Wang, and Xiaohang Dong. 2024. Better zero-shot reasoning with role-play prompting. *arXiv preprint arXiv:2308.07702*.

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. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*.

Junlong Li, Yiheng Xu, Tengchao Lv, Lei Cui, Cha Zhang, and Furu Wei. 2022. Dit: Self-supervised pre-training for document image transformer. *arXiv preprint arXiv:2203.02378*.

Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. 2024. Structrag: Boosting knowledge intensive reasoning of llms via inference-time hybrid information structurization. *arXiv preprint arXiv:2410.08815*.

Jerry Liu. 2022. Llamaindex. https://github.com/run-llama/llama_index.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pages 2263–2279.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209.

Jorj X. McKie. 2025. Pymupdf. Version 1.26.5. https://pymupdf.readthedocs.io.

Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. 2020. Plotqa: Reasoning over scientific plots. In *Proceedings of the ieee/cvf winter conference on applications of computer vision*, pages 1527–1536.

OpenAI. 2024. Ai platforms to accelerate your business. https://openai.com/business/.

OpenAI. 2025. Update to gpt-5 system card: Gpt-5.2. https://cdn.openai.com/pdf/3a4153c8-c748-4b71-8e31-aecbde944f8d/oai_5_2_system-card.pdf.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

ShengYun Peng, Aishwarya Chakravarthy, Seongmin Lee, Xiaojing Wang, Rajarajeswari Balasubramaniyan, and Duen Horng Chau. 2024. Unitable: Towards a unified framework for table recognition via self-supervised pretraining. *arXiv preprint arXiv:2403.04822*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*.

Jacob Si, Wendy Yusi Cheng, Michael Cooper, and Rahul G. Krishnan. 2024. Interpretabnet: Distilling predictive signals from tabular data by salient feature interpretation. *arXiv preprint arXiv:2406.00426*.

Jacob Si, Zijing Ou, Mike Qu, Zhengrui Xiang, and Yingzhen Li. 2025. Tabrep: Training tabular diffusion models with a simple and effective continuous representation. *arXiv preprint arXiv:2504.04798*.

Jan Strich, Enes Kutay Isgorur, Maximilian Trescher, Chris Biemann, and Martin Semmann. 2025. T2-ragbench: Text-and-table benchmark for evaluating retrieval-augmented generation. *arXiv preprint arXiv:2506.12071*.

Gemma Team. 2025a. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Qwen Team. 2025b. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Rubèn Tito, Dimosthenis Karatzas, and Ernest Valveny. 2023. Hierarchical multimodal transformers for multi-page docvqa. *arXiv preprint arXiv:2212.05935*.

Boris van Breugel and Mihaela van der Schaar. 2024. Why tabular foundation models should be a research priority. *arXiv preprint arXiv:2405.01147*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*.

Haoran Wei, Yaofeng Sun, and Yukun Li. 2025a. Deepseek-ocr: Contexts optical compression. *arXiv preprint arXiv:2510.18234*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Zhepei Wei, Wei-Lin Chen, and Yu Meng. 2025b. Instructrag: Instructing retrieval-augmented generation via self-synthesized rationales. *arXiv preprint arXiv:2406.13629*.

Tim Z. Xiao, Robert Bamler, Bernhard Schölkopf, and Weiyang Liu. 2025. Verbalized machine learning: Revisiting machine learning with language models. *arXiv preprint arXiv:2406.04344*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025a. Visrag: Vision-based retrieval-augmented generation on multi-modality documents. *arXiv preprint arXiv:2410.10594*.

Xiaohan Yu, Pu Jian, and Chong Chen. 2025b. Tablerag: A retrieval augmented generation framework for heterogeneous document reasoning. *arXiv preprint arXiv:2506.10380*.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Weichao Zhao, Hao Feng, Qi Liu, Jingqun Tang, Binghong Wu, Lei Liao, Shu Wei, Yongjie Ye, Hao Liu, Wengang Zhou, Houqiang Li, and Can Huang. 2024. Tabpedia: Towards comprehensive visual table understanding with concept synergy. *arXiv preprint arXiv:2406.01326*.

Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. 2019. Publaynet: largest dataset ever for document layout analysis. *arXiv preprint arXiv:1908.07836*.

Fengbin Zhu, Wenqiang Lei, Fuli Feng, Chao Wang, Haozhou Zhang, and Tat-Seng Chua. 2022. Towards complex document understanding by discrete reasoning. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, pages 4857–4866. ACM.

Appendix

Contents

A Implementation **13**
A.1 Hardware Details . . . 13
A.2 Model Configuration . . . 13
A.3 Qualitative Example . . . 13

B Experiments **13**
B.1 Datasets . . . 13
B.2 Baselines . . . 14
B.3 Benchmarks . . . 21

C Prompts **21**
C.1 TabRAG: Self-Generated In-Context Learning . . . 21
C.2 TabRAG . . . 21
C.3 Baselines . . . 21
C.4 LLM Query Engine . . . 21

D Outputs **25**
D.1 TabRAG: Layout Segmentation . . . 25
D.2 TabRAG . . . 25

E Additional Results & Ablation Studies **30**
E.1 Computational Cost and Complexity Analysis . . . 30
E.2 Ablation: Structured Representation (Case Study) . . . 30
E.3 Ablation: VLM Complex Prompt . . . 30

F Artifact Use, Data Privacy, and Intended Use **37**

A Implementation

A.1 Hardware Details

The following delineates the foundation of our experiments:

- Codebase: Python & PyTorch
- CPU: AMD EPYC 7443P
- GPU: NVIDIA A100 80GB PCIe / NVIDIA A6000 48GB

A.2 Model Configuration

The model configurations used are shown in Table 7. The hyperparameters for the remaining baselines (McKie, 2025; Hoffstaetter, 2025) are their default configurations. Qwen3-VL-8B-Instruct (TABRAG_{8B}) and Qwen3-VL-32B-Instruct (TABRAG_{32B}) are used as the vision-language models for TABRAG, while Qwen3-8B (No Think) is used for evaluating generation performance.

Table 7: Model Configurations.

Model	Temperature	Context Window
Qwen3-VL-8B-Instruct	1.0	32768
Qwen3-VL-32B-Instruct	1.0	32768
Qwen3-8B (No Think)	1.0	32768

A.3 Qualitative Example

We present a qualitative example from TableVQA of TABRAG in Figure 3.

B Experiments

B.1 Datasets

All datasets used in our experiments are public document question-answering or table question-answering benchmarks. The documents and questions in our processed subsets are primarily English. These datasets are not designed around speaker or author demographics, and demographic attributes are not used in our experiments. Therefore, we report dataset coverage in terms of document domain, document type, number of pages, and number of question-answer pairs, rather than demographic group coverage.

Prior to constructing the ragstores, the datasets undergo preprocessing. Each file in the dataset can contain multiple pages; thus, for consistency, we ensure that each file is separated into individual files containing a single page each. For methods that require image files as inputs, we convert data modalities such as PDFs and HTML files into high-resolution PNG files.

Table 8: Dataset Overview.

Dataset	Document Type	# Pages	# QA Pairs
TAT-DQA (Zhu et al., 2022)	Financial	312	1640
MP-DocVQA (Tito et al., 2023)	Multi-domain	500	515
WikiTQ (Pasupat and Liang, 2015)	Web Tables	243	511
TableVQA-Bench (Kim et al., 2024)	Multi-domain	427	500
ComTQA (Zhao et al., 2024)	Multi-domain	500	1000
T^2-RAGBench (FinQA) (Strich et al., 2025)	Financial	200	731

In the following section, we provide a description and visualizations of the datasets. From Figure 4 through Figure 9, we show example documents of each dataset used in our experiments.

TAT-DQA (Zhu et al., 2022) (CC BY 4.0 License). TAT-DQA comprises PDF document pages that include both semi-structured tables and unstructured textual content, each paired with corresponding question-answer sets. The documents are drawn from financial reports and feature a strong presence of numerical information. The dataset consists of separated individual PDF pages thus, there is no designated "group" of pages available for retrieval experiments. Hence, we first sort the document pages then split these pages into groups of 25. This yields 13 folders which we use to construct ragstores for retrieval.

MP-DocVQA (Tito et al., 2023) (MIT License). Document Visual Question Answering (DocVQA) datasets answer questions from document images. MP-DocVQA extends DocVQA (Mathew et al., 2021) to the multi-page scenario where DocVQA contains images such as typed or handwritten text, layout, diagrams, figures, tabular structures, and signatures. The dataset consists a total of 46K questions, 6K documents, and 48K pages (Images). We filter the dataset by selecting documents that contains tables (via layout detection), followed by documents with a high number of questions to pages ratio.

WikiTableQuestions (Pasupat and Liang, 2015) (CC-BY-SA-4.0 License). WikiTableQuestions aim to answer complex questions about semi-structured HTML tables using question-answer pairs as supervision. The dataset consists of question-answer pairs corresponding to an accompanying table. The questions include a variety of operations such as comparison, superlatives, aggregation, and arithmetic.

TableVQA-Bench (Kim et al., 2024) (CC BY-4.0 License). TableVQA-Bench is a unified benchmark that aggregates multiple table-centric datasets,

	2008	2007
Revenues	$1,364,661	$1,205,340
Net income	83,026	66,608
Net income per share—diluted	$ 1.32	$ 0.97

TabRAG's Structured Representation

```
{"row": "Revenues", "column": "2008", "value": "$1,364,661", "units": "$"}
{"row": "Revenues", "column": "2007", "value": "$1,205,340", "units": "$"}
{"row": "Net income", "column": "2008", "value": "83,026", "units": null}
{"row": "Net income", "column": "2007", "value": "66,608", "units": null}
{"row": "Net income per share-diluted", "column": "2008", "value": "$ 1.32", "units": "$"}
{"row": "Net income per share-diluted", "column": "2007", "value": "$ 0.97", "units": "$"}
```

LLM Query Engine (Input/Output)

```
You are a helpful assistant. Use the information from the documents below to answer the question.

{'TabRAG's Structured Representation'}

Question: {'What is the Net income per share--diluted in 2007?'}

Answer: $ 0.97
```

Figure 3: Small qualitative example from TableVQA of TabRAG aiding the LLM Query Engine in delivering the correct output.

including VWTQ, VWTQ-Syn, VTabFact, and FinTabNetQA, covering diverse domains such as web tables, fact verification, and financial reports. To manage the size of the dataset in our experiments (number of documents), we use the FinTabNetQA and VTabFact subsets and treat each table image as an independent document. The dataset consists of separated individual PNG pages. Thus, there is no designated "group" of pages available for retrieval experiments. Hence, we first sort the document pages then split these pages into groups of 25. This yields 18 folders which we use to construct ragstores for retrieval.

ComTQA (Zhao et al., 2024) (CC BY-NC 4.0 License). ComTQA is a visual table question answering dataset composed of table images paired with question–answer annotations. The dataset aggregates tables from PubTables-1M and FinTabNet, spanning scientific and financial domains. The dataset consists of separated individual PNG pages. Thus, there is no designated "group" of pages available for retrieval experiments. Hence, we first sort the document pages then split these pages into groups of 25. This yields 20 folders which we use to construct ragstores for retrieval.

T^2-RAGBench (FinQA) (Strich et al., 2025) (CC-BY-4.0 License). The T^2-RAGBench benchmark is a table-centric document based RAG QA benchmark comprised of 3 sub-datasets: FinQA, ConvFinQA, and TAT-DQA. We include FinQA in our experiments which is a QA dataset composed of the 2019-2020 earnings reports and financial statements of S&P 500 companies. It features a multimodal blend of tables and text sourced from annual and quarterly filings. Under the benchmark, the documents are processed as individual PDF pages to simulate a realistic, end-to-end RAG environment under a non-oracle retrieval setting.

B.2 Baselines

TabRAG's performance is evaluated in comparison to popular parsing methods. This includes PyMuPDF (McKie, 2025), PyTesseract (Hoffstaetter, 2025) and multiple vision language models drawn from the DeepSeek (Wei et al., 2025a), Gemma (Team, 2025a), GPT (OpenAI, 2025) and Qwen (Team, 2025b) model families. Unless otherwise specified, existing document parsing packages and libraries are used with their default configurations.

PyMuPDF (McKie, 2025). PyMuPDF is a Python library for data extraction and analysis of PDF documents. It interfaces with the MuPDF rendering engine, which parses the PDF's internal object structure-decoding vector drawing commands, text placements, and embedded resources-to reconstruct the page layout.

PyTesseract (Hoffstaetter, 2025). PyTesseract is

Table of Contents

NAVIOS MARITIME HOLDINGS INC.
NOTES TO THE CONSOLIDATED FINANCIAL STATEMENTS
(Expressed in thousands of U.S. dollars — except share data)

	December 31, 2019				
Income Statement	**Navios Partners**	**Navios Acquisition**	**Navios Europe I**	**Navios Europe II**	**Navios Containers**
Revenue	$219,379	$ 280,117	$ 36,822	$ 46,718	$141,532
Net (loss)/ income before non-cash change in fair value of Junior Loan I and Junior Loan II	$ (62,134)	$ (65,707)	$(18,575)	$(30,203)	$ 7,507
Net (loss)/income	$ (62,134)	$ (65,707)	$(18,575)	$(30,203)	$ 7,507

	December 31, 2018				December 31, 2017				
Income Statement	**Navios Partners**	**Navios Acquisition**	**Navios Europe I**	**Navios Europe II**	**Navios Partners**	**Navios Acquisition**	**Navios Europe I**	**Navios Europe II**	**Navios Containers**
Revenue	$231,361	$ 187,946	$ 34,885	$ 49,870	$211,652	$ 227,288	$ 37,468	$38,633	$ 39,188
Net (loss)/ income before non-cash change in fair value of Junior Loan I and Junior Loan II	$ (13,081)	$ (82,233)	$(22,881)	$(12,899)	$ (15,090)	$ (75,153)	$(20,778)	$22,749	$ 2,638
Net (loss)/income	$ (13,081)	$ (82,233)	$ (3,197)	$(12,169)	$ (15,090)	$ (75,153)	$ 9,762	$ (9,086)	$ 2,638

Available-for-sale securities ("AFS Securities")

During the year ended December 31, 2017, the Company received shares of Pan Ocean Co. Ltd ("STX") as partial compensation for the claims filed under the Korean court for all unpaid amounts in respect of the employment of the Company's vessels. The shares were recorded at fair value upon their issuance and subsequent changes in market value are recognized within accumulated other comprehensive income/(loss) or since January 1, 2018, when the Company adopted ASU 2016-01, "Financial Instruments – Overall (Subtopic 825-10) – Recognition and Measurement of Financial Assets and Financial Liabilities", within consolidated statement of comprehensive (loss)/income. The unrealized holding gain was $2 as of December 31, 2017.

The shares received from STX were accounted for under the guidance for AFS Securities. The Company has no other types of AFS Securities.

As of December 31, 2019 and 2018, the carrying amount of the AFS Securities related to STX was $189 and $192, respectively and was recorded under "Other long-term assets" in the consolidated balance sheet. During the year ended December 31, 2019, the unrealized holding losses related to these AFS Securities included in "Other expense" was $3. During the year ended December 31, 2018, the unrealized holding losses related to these AFS Securities included in "Other expense" was $46.

NOTE 10: ACCRUED EXPENSES AND OTHER LIABILITIES

Accrued expenses and other liabilities as of December 31, 2019 and 2018 consisted of the following:

	December 31, 2019	December 31, 2018
Payroll	$ 6,701	$ 15,264
Accrued interest	35,926	40,903
Accrued voyage expenses	—	3,643
Accrued running costs	—	42,212
Provision for estimated losses on vessels under time charter	—	1,604
Audit fees and related services	234	292
Accrued taxes	8,002	6,268
Professional fees	317	1,251
Other accrued expenses	—	12,215
Total accrued expenses	**$ 51,180**	**$ 123,652**

NOTE 11: BORROWINGS

Borrowings as of December 31, 2019 and 2018 consisted of the following:

F-42

Figure 4: An example page from the TAT-DQA dataset illustrating financial tables interleaved with descriptive text and numerical values.

13 The figures in Table 1 are subject to various errors. For instance the total number of birds stranded in south-west Scotland, where the greatest number came ashore, is derived from sample counts made on selected sections of beach to give an estimated figure for the whole coastline. The accuracy of the total of 17,347 dead birds depends largely on the accuracy of the estimate for guillemots found in SW Scotland, and to a lesser extent on similar estimates elsewhere. What can be said with certainty is that more than 12,000 dead birds probably came ashore round the Irish Sea in the three months from August to November, and the evidence of ringed birds, provided by the BTO, supports the view that the dead birds came mainly from colonies in the southern part of the Irish Sea. Because some are likely to have sunk or drifted into the open Atlantic, the total may well have been much higher; some ornithologists estimate the death toll as high as 50,000 100,000 birds.

14 Confronted with an incident of this kind, it is natural to search for precedents. Two species often involved in large scale wrecks in the British Isles, and indeed in the North Atlantic generally, are Leach's petrel (*Oceanodroma leucorrhoa*) and the little auk (*Plautus alle*). At least forty little auk wrecks have been recorded in the British Isles. They are often associated with storms, and a study of one of the most recent indicated that they result from a massed movement of birds from their normal Arctic habitat into more southerly waters deficient in food, where they become weakened and liable to be blown ashore by gales. The larger auks (puffin, guillemot and razorbill) are wrecked far less often than the little auk, and generally in smaller numbers. It is not unusual for a few tens or even hundreds to be cast ashore, but massed deaths like those of guillemots in the 1969 wreck are rare.

15 Previously recorded wrecks of large auks in the British Isles are summarised in Table 2 (page 4)

TABLE 1

Estimates and counts of dead birds on the beaches

(Most of these figures are based on the Beached Bird Survey organised by the RSPB and the Seabird Group)

SPECIES	EAST* SCOT-LAND	EAST ENG-LAND	SOUTH ENGLAND	WALES	NW ENGLAND	WEST SCOT-LAND	ISLE OF MAN	IRELAND	TOTAL	WINGS COL-LECTED	POST MOR-TEM
Observers	**10**	**24**	**23**	**67**	**22**	**49**	**6**	**21**	**222**	—	—
Divers and Grebes	1	4	—	1	3	—	—	—	9	—	—
Manx Shearwater	—	—	—	3	4	—	—	—	7	—	—
Fulmar	5	1	—	—	1	—	—	—	7	—	1
Gannet	6	3	3	7	5	3	3	5	35	—	—
Cormorant, Shag	2	1	1	4	6	3	2	—	19	1	1
Ducks	13	3	—	3	—	3	1	—	23	—	1
Waders	2	5	—	2	3	1	1	—	14	—	—
GBB Gull	6	3	—	—	6	—	2	—	17	—	—
Herring Gull	27	8	2	16	18	6	6	5	88	1	1
BH Gull	3	5	2	4	4	—	—	1	19	—	1
Kittiwake	5	7	1	2	4	1	3	2	25	—	2
Other Gulls	1	3	1	1	1	—	1	1	9	—	1
Terns	3	1	1	—	1	—	—	—	6	—	—
Razorbill	9	4	1	36	35	14	2	41	142	15	8
Guillemot	17	20	5	2019	898	11,816	320	1794	16,889	528	112
Other or unidentified Auks	2	1	—	30	1	3	—	1	38	—	—
TOTALS	**102ø**	**69ø**	**17ø**	**2128†**	**990ø**	**11,850†**	**341ø**	**1850†**	**17,347†**	**545**	**128**

*For boundaries of regions see map: page 8 †Guillemot figures partly based on estimates øBased on counts. 3

Figure 5: A sample document image from MP-DocVQA showing a scanned document page with dense tables and multi-column text.

Some representative refractive indices

Material	λ (nm)	*n*	Ref.
Vacuum		1 (by definition)	
Air at STP		1.000277	
Gases at 0 °C and 1 atm			
Air	589.29	1.000293	[1] [2]
Carbon dioxide	589.29	1.00045	[3] [4]
Helium	589.29	1.000036	[1]
Hydrogen	589.29	1.000132	[1]
Liquids at 20 °C			
Arsenic trisulfide and sulfur in methylene iodide		1.9	[5]
Benzene	589.29	1.501	[1]
Carbon disulfide	589.29	1.628	[1]
Carbon tetrachloride	589.29	1.461	[1]
Ethyl alcohol (ethanol)	589.29	1.361	[1]
Silicone oil		1.336-1.582	[6]
Water	589.29	1.3330	[1]
10% Glucose solution in water	589.29	1.3477	[7]
20% Glucose solution in water	589.29	1.3635	[7]
60% Glucose solution in water	589.29	1.4394	[7]
Solids at room temperature			
Titanium dioxide (Rutile phase)	589.29	2.496	[8]
Diamond	589.29	2.419	[1]
Strontium titanate	589.29	2.41	
Amber	589.29	1.55	[1]
Fused silica (also called Fused Quartz)	589.29	1.458	[1]
Sodium chloride	589.29	1.544	[9]
Other materials			
Liquid helium		1.025	
Water ice		1.31	
Cornea (human)		1.373/1.380/1.401	[10]
Lens (human)		1.386 - 1.406	

Figure 6: A rendered web table from the WikiTableQuestions dataset illustrating semi-structured tabular data with numerical and categorical entries.

	Years Ended December 31,						
	2018	2017	2016	2018 vs. 2017		2017 vs. 2016	
				$ Change	% Change	$ Change	% Change
Routing	$ 1,839.7	$ 2,189.5	$ 2,352.9	$ (349.8)	(16)%	$ (163.4)	(7)%
Switching	934.4	963.4	858.0	(29.0)	(3)%	105.4	12 %
Security	333.0	293.3	318.0	39.7	14 %	(24.7)	(8)%
Total Product	3,107.1	3,446.2	3,528.9	(339.1)	(10)%	(82.7)	(2)%
Percentage of net revenues	*66.9%*	*68.6%*	*70.7%*				
Total Service	1,540.4	1,581.0	1,461.2	(40.6)	(3)%	119.8	8 %
Percentage of net revenues	*33.1%*	*31.4%*	*29.3%*				
Total net revenues	$ 4,647.5	$ 5,027.2	$ 4,990.1	$ (379.7)	(8)%	$ 37.1	1 %
Cloud	$ 1,049.9	$ 1,310.7	$ 1,315.9	$ (260.8)	(20)%	$ (5.2)	— %
Percentage of net revenues	*22.6%*	*26.0%*	*26.4%*				
Service Provider	2,066.7	2,319.4	2,316.4	(252.7)	(11)%	3.0	— %
Percentage of net revenues	*44.5%*	*46.1%*	*46.4%*				
Enterprise	1,530.9	1,397.1	1,357.8	133.8	10 %	39.3	3 %
Percentage of net revenues	*32.9%*	*27.8%*	*27.2%*				
Total net revenues	$ 4,647.5	$ 5,027.2	$ 4,990.1	$ (379.7)	(8)%	$ 37.1	1 %
Americas:							
United States	$ 2,339.1	$ 2,712.6	$ 2,737.0	$ (373.5)	(14)%	$ (24.4)	(1)%
Other	202.1	234.6	231.8	(32.5)	(14)%	2.8	1 %
Total Americas	2,541.2	2,947.2	2,968.8	(406.0)	(14)%	(21.6)	(1)%
Percentage of net revenues	*54.7%*	*58.6%*	*59.5%*				
EMEA	1,290.8	1,195.8	1,238.1	95.0	8 %	(42.3)	(3)%
Percentage of net revenues	*27.8%*	*23.8%*	*24.8%*				
APAC	815.5	884.2	783.2	(68.7)	(8)%	101.0	13 %
Percentage of net revenues	*17.5%*	*17.6%*	*15.7%*				
Total net revenues	$ 4,647.5	$ 5,027.2	$ 4,990.1	$ (379.7)	(8)%	$ 37.1	1 %

Figure 7: An example table from the TableVQA-Bench dataset with semi-structured layouts and mixed numerical and categorical content.

Note 21 Major Customers
ScanSource, Inc. is our most significant customer and our net sales to them as a percent of total net sales were as follows:

	Year Ended December 31, 2005	2004	2003
ScanSource	15.6	14.1	13.8

No other customer accounted for 10% or more of total net sales during these years.

Note 22 Quarterly Results of Operations (unaudited)
(Amounts in thousands, except per share data)

2005	First Quarter	Second Quarter	Third Quarter	Fourth Quarter
Net sales	$170,727	$176,614	$175,636	$179,294
Cost of sales	83,362	87,266	87,959	89,503
Gross profit	87,365	89,348	87,677	89,791
Selling and marketing	21,067	22,554	20,800	25,286
Research and engineering	10,668	12,054	11,501	11,777
General and administrative	14,946	16,810	14,489	13,665
Amortization of intangibles	647	387	509	798
Exit costs	1,517	141	283	71
Total operating expenses	48,845	51,946	47,582	51,597
Operating income	38,520	37,402	40,095	38,194
Investment income (expense)	3,277	3,072	3,254	3,814
Interest expense	(3)	(27)	(41)	(8)
Foreign exchange gain (loss)	53	812	334	87
Other, net	(304)	(243)	251	(74)
Total other income (expense)	3,023	3,614	3,798	3,819
Income before taxes	41,543	41,016	43,893	42,013
Income taxes	14,436	14,253	14,453	13,720
Net income	$ 27,107	$ 26,763	$ 29,440	$ 28,293
Basic earnings per share	$ 0.38	$ 0.37	$ 0.41	$ 0.40
Diluted earnings per share	$ 0.37	$ 0.37	$ 0.41	$ 0.40

2004	First Quarter	Second Quarter	Third Quarter	Fourth Quarter
Net sales	$154,174	$162,830	$171,176	$174,874
Cost of sales	73,571	78,315	84,030	83,979
Gross profit	80,603	84,515	87,146	90,895
Selling and marketing	17,207	18,023	19,217	22,615
Research and engineering	8,896	9,233	9,596	9,368
General and administrative	12,763	12,562	11,917	11,998
Amortization of intangibles	649	626	647	647
Acquired in-process technology	22	—	—	—
Exit costs	363	876	715	146
Total operating expenses	39,900	41,320	42,092	44,774
Operating income	40,703	43,195	45,054	46,121
Investment income (expense)	3,073	2,091	2,515	2,949
Interest expense	(26)	(6)	(7)	(5)
Foreign exchange gain (loss)	(656)	413	737	(9)
Other, net	(299)	(537)	(339)	(419)
Total other income (expense)	2,092	1,961	2,906	2,516
Income before taxes	42,795	45,156	47,960	48,637
Income taxes	14,861	15,728	16,641	16,675
Net income	$ 27,934	$ 29,428	$ 31,319	$ 31,962
Basic earnings per share	$ 0.39(1)	$ 0.41(1)	$ 0.44	$ 0.45
Diluted earnings per share	$ 0.39(1)	$ 0.41(1)	$ 0.43	$ 0.44

(1) Restated for a 3-for-2 stock split in August 2004 paid in the form of a 50% stock dividend.

Figure 8: An example table from the ComTQA dataset with semi-structured layouts and mixed data types.

a Python wrapper for Google's Tesseract OCR engine (Kay, 2007), used for extracting text from images and scanned documents. It performs a raster-based analysis of the input image-segmenting it into connected components, detecting character shapes, and matching them against trained language models-to recognize and reconstruct textual content from pixel patterns.

DeepSeek-OCR (Wei et al., 2025a). DeepSeek-OCR is an OCR-oriented vision system that explores compressing long textual contexts via optical 2D mapping of document content. It consists of a high-resolution visual encoder (DeepEncoder) that produces a set of vision tokens and a decoder (DeepSeek3B-MoE-A570M) that generates recognized text from these representations. By explicitly controlling the vision-token budget and compression ratio, DeepSeek-OCR enables efficient text decoding under constrained visual-token settings, making it suitable as an OCR-based document parsing baseline.

Gemma-3-27B-IT (Team, 2025a). Gemma-3-27B-IT is a large instruction-tuned multimodal language model from the Gemma 3 family developed by Google DeepMind. It integrates visual understanding and long-context processing within a unified transformer architecture, with architectural modifications to mitigate KV-cache growth. Trained via distillation and post-training refinement, the model supports structured instruction following and multimodal reasoning over document images, tables, and text.

GPT-5.2 (OpenAI, 2025). GPT-5.2 is a state-of-the-art proprietary multimodal large language model developed by OpenAI for professional knowledge work and complex reasoning tasks. The model integrates advanced capabilities in natural language understanding, long-context processing, and tool usage within a unified architecture. GPT-5.2 demonstrates strong performance on multi-step reasoning, document and chart understanding and code generation that require planning and execution over long horizons.

Qwen3-VL-32B/8B-Instruct (Team, 2025b). Qwen3-VL-32B/8B-Instruct is a large-scale vision-language model developed by Alibaba Cloud. It integrates visual and textual understanding through a unified multimodal transformer architecture capable of processing both images and natural language. The model is trained on large-scale multimodal corpora, enabling it to interpret complex document layouts, tables, and figures while aligning visual cues with textual semantics. Its instruction-tuned variant enhances performance on document under-

As noted above, as a result of these sales, these regulated subsidiaries are presented as discontinued operations for all periods presented. Therefore, the amounts, statistics and tables presented in this section refer only to on-going operations, unless otherwise noted.

The following table sets forth our Regulated Businesses operating revenue for 2012 and number of customers from continuing operations as well as an estimate of population served as of December 31, 2012:

	Operating Revenues (in millions)	% of Total	Number of Customers	% of Total	Estimated Population Served (in millions)	% of Total
New Jersey	$ 639.0	24.9%	639,838	20.3%	2.5	21.9%
Pennsylvania	557.7	21.7%	658,153	20.8%	2.2	19.3%
Missouri	279.5	10.9%	455,730	14.4%	1.5	13.2%
Illinois(a)	256.4	10.0%	308,014	9.8%	1.2	10.5%
Indiana	198.7	7.8%	289,068	9.2%	1.2	10.5%
California	193.3	7.5%	174,188	5.5%	0.6	5.3%
West Virginia(b)	125.0	4.9%	172,159	5.4%	0.6	5.3%
Subtotal (Top Seven States)	2,249.6	87.7%	2,697,150	85.4%	9.8	86.0%
Other(c)	314.8	12.3%	461,076	14.6%	1.6	14.0%
Total Regulated Businesses	$2,564.4	100.0%	3,158,226	100.0%	11.4	100.0%

(a) Includes Illinois-American Water Company, which we refer to as ILAWC and American Lake Water Company, also a regulated subsidiary in Illinois.
(b) West Virginia-American Water Company, which we refer to as WVAWC, and its subsidiary Bluefield Valley Water Works Company.
(c) Includes data from our operating subsidiaries in the following states: Georgia, Hawaii, Iowa, Kentucky, Maryland, Michigan, New York, Tennessee, and Virginia.

Approximately 87.7% of operating revenue from our Regulated Businesses in 2012 was generated from approximately 2.7 million customers in our seven largest states, as measured by operating revenues. In fiscal year 2012, no single customer accounted for more than 10% of our annual operating revenue.

Overview of Networks, Facilities and Water Supply

Our Regulated Businesses operate in approximately 1,500 communities in 16 states in the United States. Our primary operating assets include approximately 80 surface water treatment plants, 500 groundwater treatment plants, 1,000 groundwater wells, 100 wastewater treatment facilities, 1,200 treated water storage facilities, 1,300 pumping stations, 90 dams and 46,000 miles of mains and collection pipes. Our regulated utilities own substantially all of the assets used by our Regulated Businesses. We generally own the land and physical assets used to store, extract and treat source water. Typically, we do not own the water itself, which is held in public trust and is allocated to us through contracts and allocation rights granted by federal and state agencies or through the ownership of water rights pursuant to local law. Maintaining the reliability of our networks is a key activity of our Regulated Businesses. We have ongoing infrastructure renewal programs in all states in which our Regulated Businesses operate. These programs consist of both rehabilitation of existing mains and replacement of mains that have reached the end of their useful service lives.

Our ability to meet the existing and future water demands of our customers depends on an adequate supply of water. Drought, governmental restrictions, overuse of sources of water, the protection of threatened species or habitats or other factors may limit the availability of ground and surface water. We employ a variety of measures to ensure that we have adequate sources of water supply, both in the short-term and over the long-term. The geographic diversity of our service areas tends to mitigate some of the economic effect of weather extremes we

Figure 9: An example table from the FinQA dataset illustrating semi-structured layouts with financial data.

standing and question answering tasks by following structured prompts and adapting to diverse multimodal reasoning scenarios.

B.3 Benchmarks

Accuracy (Yu et al., 2025a; Faysse et al., 2025). Accuracy is computed by evaluating the model's ability to produce correct answers to the ground-truth questions. For each document, the corresponding docstore and question are passed to the language model to generate an answer. The generated response is normalized and compared against the normalized ground-truth answers. A prediction is counted as correct if all ground-truth answers appear in the model's response. The final accuracy is calculated as the ratio of correctly answered questions to the total number of evaluated questions.

Mean Reciprocal Rank (Yu et al., 2025a). Mean Reciprocal Rank (MRR@10) evaluates the ranking quality of the retrieved document components relative to the ground-truth evidence. For each question, the corresponding document's vector store is loaded, and the question embedding is used to retrieve the top-10 most relevant components. The reciprocal rank is computed based on the first retrieved document containing the ground-truth evidence. The final MRR@10 score is obtained by averaging the reciprocal ranks across all evaluated questions, as defined by

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}, \tag{8}$$

where $|Q|$ is the total number of questions and rank_i denotes the rank position of the first relevant retrieved component for the i-th query.

C Prompts

This section details the prompts developed for the TabRAG framework modules and the baseline methods.

C.1 TabRAG: Self-Generated In-Context Learning

To construct our Self-Generated In-Context Learning (ICL) examples, we begin by creating a representative Markdown string to textually represent the target table. We utilize the prompt in Figure 10, which includes several rudimentary ICL examples that provide concrete demonstrations of the extraction rules.

Additionally, we use the VLM to generate in-context examples for our structured JSON representation, shown in Figure 11. This process converts visual tables into a standardized JSON format that maps headers to cell values. Including these exemplars provides the model with the structural grounding to accurately parse complex layouts, such as multi-level headers and merged cells. The prompt is shown below.

After generating both the Markdown and JSON representations using the prompts above, we merge them to form the final input-output pairs for in-context learning. As shown in Figure 12, the Markdown string serves as the input context, while the JSON object provides the target structured output. This pairing teaches the model to map visual layouts directly to our "row - column - value" schema.

C.2 TabRAG

Table Component Prompt. Following layout segmentation and region classification, TabRAG applies component-specific prompt templates conditioned on the semantic type of each detected region. Specifically, for tables, we couple our structured representation prompt along with our self-generated ICL examples, as presented in Figure 13. An example illustrating the table extraction prompt is displayed in Figure 14.

Other Component Prompts. TabRAG also applies VLM prompts to summarize and parse other non-table components. These prompts are shown from Figure 15 through Figure 17.

Page Overview Prompt. We use the prompt in Figure 18 to generate a page overview, which supplements the component-wise representations by providing cross-component context.

C.3 Baselines

Per conventional established parsing techniques (Artifex Software, Inc., 2024; Boulanger, 2023; Liu, 2022), all baseline vision-language and OCR-centric models are evaluated under a uniform prompting setup, as shown in Figure 19. This includes DeepSeek-OCR, Gemma-3-27B-IT, GPT-5.2 and Qwen3-VL-8B/32B-Instruct.

C.4 LLM Query Engine

We use the prompt in Figure 20 to query an LLM along with the attached respective ICL example as context.

TabRAG Self-Generated ICL: Markdown

```
You are a precise information parsing and extraction engine. Output ONLY
A Markdown table of the provided image

Task: Generate a Markdown table with the correct structure, with every single cell accounted for.

Rules:
1. Output ONLY a valid Markdown table. Do not include any other text.
2. Preserve the exact number of rows and columns visible in the image.
3. Preserve unit information in headers if present (e.g., "$ in millions").
4. Transcribe all cell contents as faithfully as possible without inference, normalization, or computation.
5. Repeat values for merged cells so that every row has the same number of columns.
6. If a cell is visually empty, leave it empty.
7. Ignore captions, footnotes, and non-table elements outside the table.


---

**Example Output 1: Two-level header**

| ($ in millions) | 2024             | 2023             |
|-----------------|------------------|------------------|
|                 | Revenue | Profit | Revenue | Profit |
| Sales           | 1,234   | 400    | 1,200   | 350    |
| Net Income      | (56)    | 80     | -40     | 70     |

---

**Example Output 2: Three-level header**

| ($ in thousands) | 2024                                  | 2023                                  |
|------------------|---------------------------------------|---------------------------------------|
|                  | Q1                | Q2                | Q1                | Q2                |
|                  | Revenue | Profit  | Revenue | Profit | Revenue | Profit | Revenue | Profit |
| Product A        | 500     | 120     | 600     | 150    | 450     | 100    | 550     | 140    |
| Product B        | (50)    | 80      | (30)    | 100    | -20     | 60     | 10      | 90     |

---

**Example Output 3: Mixed 1-row, 2-row, and 3-row headers**

| Category              | 2024                                  | 2023             | Growth % | Notes    |
|----------------------|---------------------------------------|------------------|----------|----------|
|                      | Q1                | Q2                | Revenue | Profit |          |          |
|                      | Revenue | Profit | Revenue | Profit |         |        |          |          |
| Sales (Units)        | 1,000   | 300    | 900     | 250    | 1,700   | 550    | 12%      | N/A      |
| Cost ($ in thousands) | (200)   | (50)   | -180    | -40    | (380)   | (90)   | N/A      | Adjusted |

---

Now, generate the Markdown table ONLY, do not include any other irrelevant text.
```

Figure 10: Prompt of self-generated ICL module in extracting input table into textual Markdown representation.

TabRAG Self-Generated ICL: JSON

```
You are a precise information parsing and extraction engine. Output ONLY
A JSON array of objects, each with: {"row": <string>, "column": <string>, "value": <string|null>, "units": <string|null>}.

Task: Then Extract every visible cell in the attached table image into JSON objects.

Each table cell must be represented as:
{
"row": string,        // the row label (e.g. "Revenue", "2024", "Row 1" if unnamed)
"column": string,     // column header text; if multi-level, join levels with " -> "
"value": string|null,  // exact text as seen in the table (keep symbols and brackets)
"units": string|null // units if present in header (e.g., "$ in millions"), otherwise null
}
```

Figure 11: Prompt of self-generated ICL module in extracting input table into textual JSON representation.

TabRAG Self-Generated ICL: JSON + Markdown

```
**INPUT**:
{Markdown Representation}

**OUTPUT**:
{JSON Representation}
```

Figure 12: Concatenation of JSON and Markdown ICLs to form our Self-Generated ICL Example.

TabRAG: Table

```
You are a precise information extraction engine. Output ONLY a JSON array of objects, each with:
{"row": <string>, "column": <string>, "value": <string|null>, "units": <string|null>}.
No Markdown, explanations, or text before/after the JSON.

Task: Extract every visible cell in the attached table image into JSON triples.

Each table cell must be represented as:
{
"row": string,        // the row label (e.g. "Revenue", "2024", "Row 1" if unnamed)
"column": string,     // column header text; if multi-level, join levels with " -> "
"value": string|null, // exact text as seen in the table (keep symbols and brackets)
"units": string|null, // units if present in header (e.g., "$ in millions"), otherwise null
}

Rules:
- Output ONLY a JSON array: [ {row, column, value, units}, ... ].
- Order: top-to-bottom, left-to-right.
- Preserve all text formatting exactly as shown:
- Keep parentheses, minus signs, commas, currency symbols, and percent signs.
- Do NOT normalize numbers or remove punctuation.
- Multi-line text: join with a single space.
- Multi-level headers: join with " -> " (e.g. "2024 -> Revenue").
- If a row header spans multiple rows, repeat its label for each affected row.
- If units are present, include them in the output (e.g., "$ in millions"), otherwise use null.
- Use null only for empty or blank cells.

---

{``Self-Generated ICL''}

Now, extract all visible cells from the attached table image and output only the JSON array of {row, column, value, units}
      objects
using the " -> " separator for multi-level headers, keeping all cell values exactly as written in the table.
ENSURING THAT ALL EXTRACTED VALUES ARE ACCURATE IS THE MOST IMPORTANT! DO NOT OUTPUT ANYTHING ELSE.
```

Figure 13: Prompt of structured representation module in parsing table component.

TabRAG: Table Extraction Prompt with Example Self-Generated In-Context Learning

INSTRUCTIONS

```
You are a precise information extraction engine. Output ONLY a JSON array of objects, each with:
{"row": <string>, "column": <string>, "value": <string|null>, "units": <string|null>}.
No markdown, explanations, or text before/after the JSON.

Task: Extract every visible cell in the attached table image into JSON triples.

Each table cell must be represented as:
{
"row": string,        // the row label (e.g. "Revenue", "2024", "Row 1" if unnamed)
"column": string,     // column header text; if multi-level, join levels with " -> "
"value": string|null, // exact text as seen in the table (keep symbols and brackets)
"units": string|null, // units if present in header (e.g., "$ in millions"), otherwise null
}
```

RULES

```
Rules:
- Output ONLY a JSON array: [ {row, column, value, units}, ... ].
- Order: top-to-bottom, left-to-right.
- Preserve all text formatting exactly as shown:
- Keep parentheses, minus signs, commas, currency symbols, and percent signs.
- Do NOT normalize numbers or remove punctuation.
- Multi-line text: join with a single space.
- Multi-level headers: join with " -> " (e.g. "2024 -> Revenue").
- If a row header spans multiple rows, repeat its label for each affected row.
- If units are present, include them in the output (e.g., "$ in millions"), otherwise use null.
- Use null only for empty or blank cells.

---
```

SELF-GENERATED ICL EXAMPLE

```
Input:
|million RMB                    |31 December 2019|                     |31 December 2018 |                                |
|-------------------------------|----------------|---------------------|-----------------|--------------------------------|
|                               |Amount          |% of segment revenues|Amount (Restated)|% of segment revenues (Restated)|
|VAS                            |26,120          |50%                  |20,330           |47%                             |
|FinTech and Business Services|21,520          |72%                  |16,310           |76%                             |
|Online Advertising             |9,241           |46%                  |10,800           |63%                             |
|Others                         |2,778           |84%                  |2,304            |88%                             |
|Total cost of revenues         |59,659          |                     |49,744           |                                |


Output:
[
  {"row":"VAS","column":"31 December 2019 -> Amount","value":"26,120","units":"million RMB"},
  {"row":"VAS","column":"31 December 2019 -> % of segment revenues","value":"50%","units":"%"},
  {"row":"VAS","column":"31 December 2018 -> Amount","value":"20,330","units":"million RMB"},
  {"row":"VAS","column":"31 December 2018 -> % of segment revenues","value":"47%","units":"%"},

  {"row":"FinTech and Business Services","column":"31 December 2019 -> Amount","value":"21,520","units":"million RMB"},
  {"row":"FinTech and Business Services","column":"31 December 2019 -> % of segment revenues","value":"72%","units":"%"},
  {"row":"FinTech and Business Services","column":"31 December 2018 -> Amount","value":"16,310","units":"million RMB"},
  {"row":"FinTech and Business Services","column":"31 December 2018 -> % of segment revenues","value":"76%","units":"%"},

  {"row":"Online Advertising","column":"31 December 2019 -> Amount","value":"9,241","units":"million RMB"},
  {"row":"Online Advertising","column":"31 December 2019 -> % of segment revenues","value":"46%","units":"%"},
  {"row":"Online Advertising","column":"31 December 2018 -> Amount","value":"10,800","units":"million RMB"},
  {"row":"Online Advertising","column":"31 December 2018 -> % of segment revenues","value":"63%","units":"%"},

  {"row":"Others","column":"31 December 2019 -> Amount","value":"2,778","units":"million RMB"},
  {"row":"Others","column":"31 December 2019 -> % of segment revenues","value":"84%","units":"%"},
  {"row":"Others","column":"31 December 2018 -> Amount","value":"2,304","units":"million RMB"},
  {"row":"Others","column":"31 December 2018 -> % of segment revenues","value":"88%","units":"%"},

  {"row":"Total cost of revenues","column":"31 December 2019 -> Amount","value":"59,659","units":"million RMB"},
  {"row":"Total cost of revenues","column":"31 December 2018 -> Amount","value":"49,744","units":"million RMB"}
]
```

FORMAT

```
Now, extract all visible cells from the attached table image and output only the JSON array of {row, column, value, units} objects
using the " -> " separator for multi-level headers, keeping all cell values exactly as written in the table.
ENSURING THAT ALL EXTRACTED VALUES ARE ACCURATE IS THE MOST IMPORTANT! DO NOT OUTPUT ANYTHING ELSE.
```

Figure 14: Sample TabRAG prompt with one self-generated ICL example.

TabRAG: Text

```
Please extract and output the **visible text** in the image exactly **as it appears**, without rephrasing, summarizing, or
      skipping
any content. Preserve original formatting such as line breaks, punctuation, and capitalization. This includes any small
     footnotes
or embedded labels. DO NOT OUTPUT ANYTHING ELSE!
```

Figure 15: Prompt of structured representation module in parsing text component.

TabRAG: Title

```
Please extract and output the **title text** from the image exactly **as displayed**, preserving capitalization and
     formatting. Do
not interpret or rewrite. Output the title as it appears visually. DO NOT OUTPUT ANYTHING ELSE!
```

Figure 16: Prompt of structured representation module in parsing title component.

TabRAG: Figure

```
Please interpret the figure and describe it in detail. Your output should include:
1. Descriptions of individual data points if visible,
2. Descriptions of trend lines, axes, and labels,
3. Explanations of any color or shape encodings, and
4. Any other notable features (e.g., anomalies, clustering, outliers).
Be precise and avoid speculation. Ensure your interpretation **accurately matches the figure** and
corresponds to what is visually present. DO NOT OUTPUT ANYTHING ELSE!
```

Figure 17: Prompt of structured representation module in parsing figure component.

TabRAG: Page

```
Please parse everything in the attached image and output the parsed contents only without anything else.
```

Figure 18: Prompt of structured representation module in generating page overview.

Baseline

```
Please parse everything in the attached image and output the parsed contents only without anything else.
```

Figure 19: Prompt of baseline methods in document parsing.

LLM Query Engine

```
You are a helpful assistant. Use the information from the documents below to answer the question.

{'Retrieved Information'}

Question: {'Question'}

Answer:
```

Figure 20: Prompt of LLM Query Engine in question answering to generate the final answer from a given query.

D Outputs

D.1 TabRAG: Layout Segmentation

Given a high-resolution document page, our Document Image Transformer-based layout detection model predicts semantically meaningful regions corresponding to various document components such as tables, text blocks, titles, and figures. Each component is associated with a bounding box and a confidence score. As shown in Figure 21, the model separates the tabular region from the surrounding text blocks and section headers. Each table is detected as a single coherent region containing multiple rows, columns, headers, and numerical entries. This demonstrates the model's ability to preserve fine-grained structural organization that is critical for downstream table reasoning.

D.2 TabRAG

In this section, we showcase TABRAG's stage-by-stage outputs using an example document in Figure 21. In Figure 22 to Figure 26 we observe the TABRAG outputs from `table`, `title`, `text` and page overview components.

Item 6. Selected Financial Data

SELECTED HISTORICAL CONSOLIDATED FINANCIAL AND OPERATING INFORMATION

The following table sets forth our summary historical financial information for the periods ended and as of the dates indicated. You should read the following historical financial information along with "Management's Discussion and Analysis of Financial Condition and Results of Operations" contained in this Form 10-K. The fiscal year financial information included in the table below for the years ended December 31, 2007, 2006, and 2005, respectively, is derived from audited financial statements. Information for the years ended December 31, 2004 and 2003 can be found in our previously filed Annual Reports on Form 10-K.

	Year Ended December 31				
	2007	2006	2005	2004	2003
	(In thousands, except per share amounts)				
Income statement data					
Total revenue	$2,291,189	$1,998,742	$1,552,437	$1,257,438	$1,046,544
Cost of operations (exclusive of amortization and depreciation disclosed separately below)[1]	1,631,029	1,434,620	1,124,590	916,201	788,874
General and administrative[1]	80,898	91,815	91,532	77,740	52,320
Depreciation and other amortization	84,338	65,443	58,565	62,586	53,948
Amortization of purchased intangibles	82,294	59,597	41,142	28,812	20,613
Impairment of long-lived assets	39,961	—	—	—	—
Loss on sale of assets	16,045	—	—	—	—
Merger costs	12,349	—	—	—	—
Total operating expenses	1,946,914	1,651,475	1,315,829	1,085,339	915,755
Operating income	344,275	347,267	236,608	172,099	130,789
Other expenses	—	—	—	—	4,275
Fair value loss on interest rate derivative	—	—	—	808	2,851
Interest expense, net	69,523	40,998	14,482	6,972	14,681
Income before income taxes	274,752	306,269	222,126	164,319	108,982
Provision for income taxes	110,691	116,664	83,381	61,948	41,684
Net income	$ 164,061	$ 189,605	$ 138,745	$ 102,371	$ 67,298
Net income per share—basic	$ 2.09	$ 2.38	$ 1.69	$ 1.27	$ 0.86
Net income per share—diluted	$ 2.03	$ 2.32	$ 1.64	$ 1.22	$ 0.84
Weighted average shares used in computing per share amounts—basic	78,403	79,735	82,208	80,614	78,003
Weighted average shares used in computing per share amounts—diluted	80,811	81,686	84,637	84,040	80,313

[1] Included in general and administrative is stock compensation expense of $21.2 million, $16.1 million, $14.1 million, $13.4 million, and $5.9 million, for the years ended December 31, 2007, 2006, 2005, 2004 and 2003, respectively. Included in cost of operations is stock compensation expense of $35.0 million, $27.0 million, $0, $2.3 million, and $0, for the years ended December 31, 2007, 2006, 2005, 2004 and 2003, respectively.

Figure 21: Layout segmentation for a sample document from the ComTQA dataset.

TabRAG: Table


```
{"row": "Total revenue", "column": "2007", "value": "$2,291,189", "units": "thousand"},
{"row": "Total revenue", "column": "2006", "value": "$1,998,742", "units": "thousand"},
{"row": "Total revenue", "column": "2005", "value": "$1,552,437", "units": "thousand"},
{"row": "Total revenue", "column": "2004", "value": "$1,257,438", "units": "thousand"},
{"row": "Total revenue", "column": "2003", "value": "$1,046,544", "units": "thousand"},
{"row": "Cost of operations (exclusive of amortization and depreciation disclosed separately below)(1)", "column": "2007",
"value": "1,631,029", "units": "thousand"},
{"row": "Cost of operations (exclusive of amortization and depreciation disclosed separately below)(1)", "column": "2006",
"value": "1,434,620", "units": "thousand"},
{"row": "Cost of operations (exclusive of amortization and depreciation disclosed separately below)(1)", "column": "2005",
"value": "1,124,590", "units": "thousand"},
{"row": "Cost of operations (exclusive of amortization and depreciation disclosed separately below)(1)", "column": "2004",
"value": "916,201", "units": "thousand"},
{"row": "Cost of operations (exclusive of amortization and depreciation disclosed separately below)(1)", "column": "2003",
"value": "788,874", "units": "thousand"},
{"row": "General and administrative(1)", "column": "2007", "value": "80,898", "units": "thousand"},
{"row": "General and administrative(1)", "column": "2006", "value": "91,815", "units": "thousand"},
{"row": "General and administrative(1)", "column": "2005", "value": "91,532", "units": "thousand"},
{"row": "General and administrative(1)", "column": "2004", "value": "77,740", "units": "thousand"},
{"row": "General and administrative(1)", "column": "2003", "value": "52,320", "units": "thousand"},
{"row": "Depreciation and other amortization", "column": "2007", "value": "84,338", "units": "thousand"},
{"row": "Depreciation and other amortization", "column": "2006", "value": "65,443", "units": "thousand"},
{"row": "Depreciation and other amortization", "column": "2005", "value": "58,565", "units": "thousand"},
{"row": "Depreciation and other amortization", "column": "2004", "value": "62,586", "units": "thousand"},
{"row": "Depreciation and other amortization", "column": "2003", "value": "53,948", "units": "thousand"},
{"row": "Amortization of purchased intangibles", "column": "2007", "value": "82,294", "units": "thousand"},
{"row": "Amortization of purchased intangibles", "column": "2006", "value": "59,597", "units": "thousand"},
{"row": "Amortization of purchased intangibles", "column": "2005", "value": "41,142", "units": "thousand"},
{"row": "Amortization of purchased intangibles", "column": "2004", "value": "28,812", "units": "thousand"},
{"row": "Amortization of purchased intangibles", "column": "2003", "value": "20,613", "units": "thousand"},
{"row": "Impairment of long-lived assets", "column": "2007", "value": "39,961", "units": "thousand"},
{"row": "Impairment of long-lived assets", "column": "2006", "value": "-", "units": "thousand"},
{"row": "Impairment of long-lived assets", "column": "2005", "value": "-", "units": "thousand"},
{"row": "Impairment of long-lived assets", "column": "2004", "value": "-", "units": "thousand"},
{"row": "Impairment of long-lived assets", "column": "2003", "value": "-", "units": "thousand"},
{"row": "Loss on sale of assets", "column": "2007", "value": "16,045", "units": "thousand"},
{"row": "Loss on sale of assets", "column": "2006", "value": "-", "units": "thousand"},
{"row": "Loss on sale of assets", "column": "2005", "value": "-", "units": "thousand"},
{"row": "Loss on sale of assets", "column": "2004", "value": "-", "units": "thousand"},
{"row": "Loss on sale of assets", "column": "2003", "value": "-", "units": "thousand"},
{"row": "Merger costs", "column": "2007", "value": "12,349", "units": "thousand"},
{"row": "Merger costs", "column": "2006", "value": "-", "units": "thousand"},
{"row": "Merger costs", "column": "2005", "value": "-", "units": "thousand"},
{"row": "Merger costs", "column": "2004", "value": "-", "units": "thousand"},
{"row": "Merger costs", "column": "2003", "value": "-", "units": "thousand"},
{"row": "Total operating expenses", "column": "2007", "value": "1,946,914", "units": "thousand"},
{"row": "Total operating expenses", "column": "2006", "value": "1,651,475", "units": "thousand"},
{"row": "Total operating expenses", "column": "2005", "value": "1,315,829", "units": "thousand"},
{"row": "Total operating expenses", "column": "2004", "value": "1,085,339", "units": "thousand"},
{"row": "Total operating expenses", "column": "2003", "value": "915,755", "units": "thousand"},
{"row": "Operating income", "column": "2007", "value": "344,275", "units": "thousand"},
{"row": "Operating income", "column": "2006", "value": "347,267", "units": "thousand"},
{"row": "Operating income", "column": "2005", "value": "236,608", "units": "thousand"},
{"row": "Operating income", "column": "2004", "value": "172,099", "units": "thousand"},
{"row": "Operating income", "column": "2003", "value": "130,789", "units": "thousand"},
{"row": "Other expenses", "column": "2007", "value": "-", "units": "thousand"},
{"row": "Other expenses", "column": "2006", "value": "-", "units": "thousand"},
{"row": "Other expenses", "column": "2005", "value": "-", "units": "thousand"},
{"row": "Other expenses", "column": "2004", "value": "-", "units": "thousand"},
{"row": "Other expenses", "column": "2003", "value": "4,275", "units": "thousand"},
{"row": "Fair value loss on interest rate derivative", "column": "2007", "value": "-", "units": "thousand"},
{"row": "Fair value loss on interest rate derivative", "column": "2006", "value": "-", "units": "thousand"},
{"row": "Fair value loss on interest rate derivative", "column": "2005", "value": "-", "units": "thousand"},
{"row": "Fair value loss on interest rate derivative", "column": "2004", "value": "808", "units": "thousand"},
{"row": "Fair value loss on interest rate derivative", "column": "2003", "value": "2,851", "units": "thousand"},
{"row": "Interest expense, net", "column": "2007", "value": "69,523", "units": "thousand"},
{"row": "Interest expense, net", "column": "2006", "value": "40,998", "units": "thousand"},
{"row": "Interest expense, net", "column": "2005", "value": "14,482", "units": "thousand"},
{"row": "Interest expense, net", "column": "2004", "value": "6,972", "units": "thousand"},
{"row": "Interest expense, net", "column": "2003", "value": "14,681", "units": "thousand"},
{"row": "Income before income taxes", "column": "2007", "value": "274,752", "units": "thousand"},
{"row": "Income before income taxes", "column": "2006", "value": "306,269", "units": "thousand"},
{"row": "Income before income taxes", "column": "2005", "value": "222,126", "units": "thousand"},
{"row": "Income before income taxes", "column": "2004", "value": "164,319", "units": "thousand"},
{"row": "Income before income taxes", "column": "2003", "value": "108,982", "units": "thousand"},
{"row": "Provision for income taxes", "column": "2007", "value": "110,691", "units": "thousand"},
{"row": "Provision for income taxes", "column": "2006", "value": "116,664", "units": "thousand"},
{"row": "Provision for income taxes", "column": "2005", "value": "83,381", "units": "thousand"},
{"row": "Provision for income taxes", "column": "2004", "value": "61,948", "units": "thousand"},
{"row": "Provision for income taxes", "column": "2003", "value": "41,684", "units": "thousand"},
{"row": "Net income", "column": "2007", "value": "$ 164,061", "units": "thousand"},
{"row": "Net income", "column": "2006", "value": "$ 189,605", "units": "thousand"},
```


Figure 22: Extracted structured JSON representation of the example image.

```
{"row": "Net income", "column": "2005", "value": "$ 138,745", "units": "thousand"},
{"row": "Net income", "column": "2004", "value": "$ 102,371", "units": "thousand"},
{"row": "Net income", "column": "2003", "value": "$ 67,298", "units": "thousand"},
{"row": "Net income per share-basic", "column": "2007", "value": "$ 2.09", "units": "null"},
{"row": "Net income per share-basic", "column": "2006", "value": "$ 2.38", "units": "null"},
{"row": "Net income per share-basic", "column": "2005", "value": "$ 1.69", "units": "null"},
{"row": "Net income per share-basic", "column": "2004", "value": "$ 1.27", "units": "null"},
{"row": "Net income per share-basic", "column": "2003", "value": "$ 0.86", "units": "null"},
{"row": "Net income per share-diluted", "column": "2007", "value": "$ 2.03", "units": "null"},
{"row": "Net income per share-diluted", "column": "2006", "value": "$ 2.32", "units": "null"},
{"row": "Net income per share-diluted", "column": "2005", "value": "$ 1.64", "units": "null"},
{"row": "Net income per share-diluted", "column": "2004", "value": "$ 1.22", "units": "null"},
{"row": "Net income per share-diluted", "column": "2003", "value": "$ 0.84", "units": "null"},
{"row": "Weighted average shares used in computing per share amounts-basic", "column": "2007", "value": "78,403", "units":
       "null"},
{"row": "Weighted average shares used in computing per share amounts-basic", "column": "2006", "value": "79,735", "units":
       "null"},
{"row": "Weighted average shares used in computing per share amounts-basic", "column": "2005", "value": "82,208", "units":
       "null"},
{"row": "Weighted average shares used in computing per share amounts-basic", "column": "2004", "value": "80,614", "units":
       "null"},
{"row": "Weighted average shares used in computing per share amounts-basic", "column": "2003", "value": "78,003", "units":
       "null"},
{"row": "Weighted average shares used in computing per share amounts-diluted", "column": "2007", "value": "80,811",
"units": "null"},
{"row": "Weighted average shares used in computing per share amounts-diluted", "column": "2006", "value": "81,686",
"units": "null"},
{"row": "Weighted average shares used in computing per share amounts-diluted", "column": "2005", "value": "84,637",
"units": "null"},
{"row": "Weighted average shares used in computing per share amounts-diluted", "column": "2004", "value": "84,040",
"units": "null"},
{"row": "Weighted average shares used in computing per share amounts-diluted", "column": "2003", "value": "80,313",
"units": "null"}
```


Figure 23: Extracted structured JSON representation of the example image (Continued).

TabRAG: Title

```
SELECTED HISTORICAL CONSOLIDATED FINANCIAL AND OPERATING INFORMATION
Income statement data
```

Figure 24: Extracted title component.

TabRAG: Text

```
The following table sets forth our summary historical financial information for the periods ended and as of the dates
      indicated.
You should read the following historical financial information along with "Management's Discussion and Analysis of
      Financial
Condition and Results of Operations" contained in this Form 10-K. The fiscal year financial information included in the
      table below
for the years ended December 31, 2007, 2006, and 2005, respectively, is derived from audited financial statements.
      Information for
the years ended December 31, 2004 and 2003 can be found in our previously filed Annual Reports on Form 10-K.
Year Ended December 31,
```

Figure 25: Extracted text component.

TabRAG: Page Overview

```
Item 6. Selected Financial Data
SELECTED HISTORICAL CONSOLIDATED FINANCIAL AND OPERATING INFORMATION
The following table sets forth our summary historical financial information for the periods ended and as of the dates
      indicated.
You should read the following historical financial information along with "Management's Discussion and Analysis of
      Financial
Condition and Results of Operations" contained in this Form 10-K. The fiscal year financial information included in the
      table below
for the years ended December 31, 2007, 2006, and 2005, respectively, is derived from audited financial statements.
      Information for
the years ended December 31, 2004 and 2003 can be found in our previously filed Annual Reports on Form 10-K.
Year Ended December 31,
2007     2006     2005     2004     2003
(In thousands, except per share amounts)
Income statement data
Total revenue  $2,291,189  $1,998,742  $1,552,437  $1,257,438  $1,046,544
Cost of operations (exclusive of
amortization and depreciation disclosed
separately below)(1)  1,631,029  1,434,620  1,124,590  916,201  788,874
General and administrative(1)  80,898  91,815  91,532  77,740  52,320
Depreciation and other amortization  84,338  65,443  58,565  62,586  53,948
Amortization of purchased intangibles  82,294  59,597  41,142  28,812  20,613
Impairment of long-lived assets  39,961  -  -  -  -
Loss on sale of assets  16,045  -  -  -  -
Merger costs  12,349  -  -  -  -
Total operating expenses  1,946,914  1,651,475  1,315,829  1,085,339  915,755
Operating income  344,275  347,267  236,608  172,099  130,789
Other expenses  -  -  -  -  4,275
Fair value loss on interest rate derivative  -  -  -  808  2,851
Interest expense, net  69,523  40,998  14,482  6,972  14,681
Income before income taxes  274,752  306,269  222,126  164,319  108,982
Provision for income taxes  110,691  116,664  83,381  61,948  41,684
Net income  $ 164,061  $ 189,605  $ 138,745  $ 102,371  $ 67,298
Net income per share-basic  $ 2.09  $ 2.38  $ 1.69  $ 1.27  $ 0.86
Net income per share-diluted  $ 2.03  $ 2.32  $ 1.64  $ 1.22  $ 0.84
Weighted average shares used in computing
per share amounts-basic  78,403  79,735  82,208  80,614  78,003
Weighted average shares used in computing
per share amounts-diluted  80,811  81,686  84,637  84,040  80,313
(1) Included in general and administrative is stock compensation expense of $21.2 million, $16.1 million, $14.1 million,
      $13.4
million, and $5.9 million, for the years ended December 31, 2007, 2006, 2005, 2004 and 2003, respectively. Included in
      cost of
operations is stock compensation expense of $35.0 million, $27.0 million, $0, $2.3 million, and $0, for the years ended
December 31, 2007, 2006, 2005, 2004 and 2003, respectively.
```

Figure 26: Extracted page overview component.

E Additional Results & Ablation Studies

In this section, we present additional experimental results and ablation analysis.

E.1 Computational Cost and Complexity Analysis

While our method introduces several discrete stages (layout detection, region-wise parsing, and page overviews), our preliminary benchmarking suggests that the increase in latency and token use is only marginal, and is offset by the significantly higher generation accuracy as seen in Table 1.

We perform a performance audit of TabRAG on the ComTQA dataset in comparison with our naive baseline, which inputs a full-page image into a VLM with a global extraction prompt. This involves measuring the latency and input tokens of each stage in both methods. Table 9 summarises the findings.

METRIC	BASELINE (VLM)	TABRAG
OVERALL LATENCY (S)	21.95	50.05
LAYOUT DETECTION (S)	—	0.61
EXTRACTION LATENCY (S)	21.84	4.34
OVERVIEW LATENCY (S)	—	22.05
EXTRACTION TOKENS	1,783.07	2,100.58
OVERVIEW TOKENS	—	1,054.07
COMPONENTS EXTRACTED / PAGE	—	6.83

Table 9: Latency (seconds) and token usage per document on ComTQA. tok = input tokens; time = wall-clock seconds.

Layout Detection. The overhead of the layout detection stage is negligible, with an average latency of only 0.61 seconds. This stage provides the structural "map" that prevents the VLM from wasting attention on non-informative whitespace or decorative elements.

Structured Representation Parsing. Latency: Each component is extracted in an average 4.34 seconds. As these are targeted components in the document, the VLM avoids the lack of context inherent in full-page processing. Token Usage: The total input tokens for extraction (2,100) are only 17.8% higher than the baseline (1,783). This is a highly efficient trade-off: for a small increase in tokens, the model receives significantly higher pixel density for complex regions like tables.

Page Overview. The Page Overview stage takes 22.05 seconds, nearly identical to the total time of the VLM baseline. This confirms our hypothesis that the "Overview" provides the same level of global context as a naive pass but is used as a semantic anchor rather than the sole source of truth. This redundancy ensures that the retriever can find the page even if a specific table cell is queried.

E.2 Ablation: Structured Representation (Case Study)

In this section, we perform a case study by providing a visual comparison of the outputs generated by each structured representation method used in our ablation experiments. We use a sample document from the ComTQA dataset in Figure 27.

To determine the optimal structured representation for tables, we compare our JSON representation (see Appendix C.2) with Markdown and natural language text in an ablation study. Prompts for generating the ICL examples and structural representations are shown from Figures 28 through 29 for the Markdown format, and from Figures 30 through 31 for the natural language text format.

Figures 32 through 34 illustrate the constructed representations in JSON, Markdown, and natural language text formats.

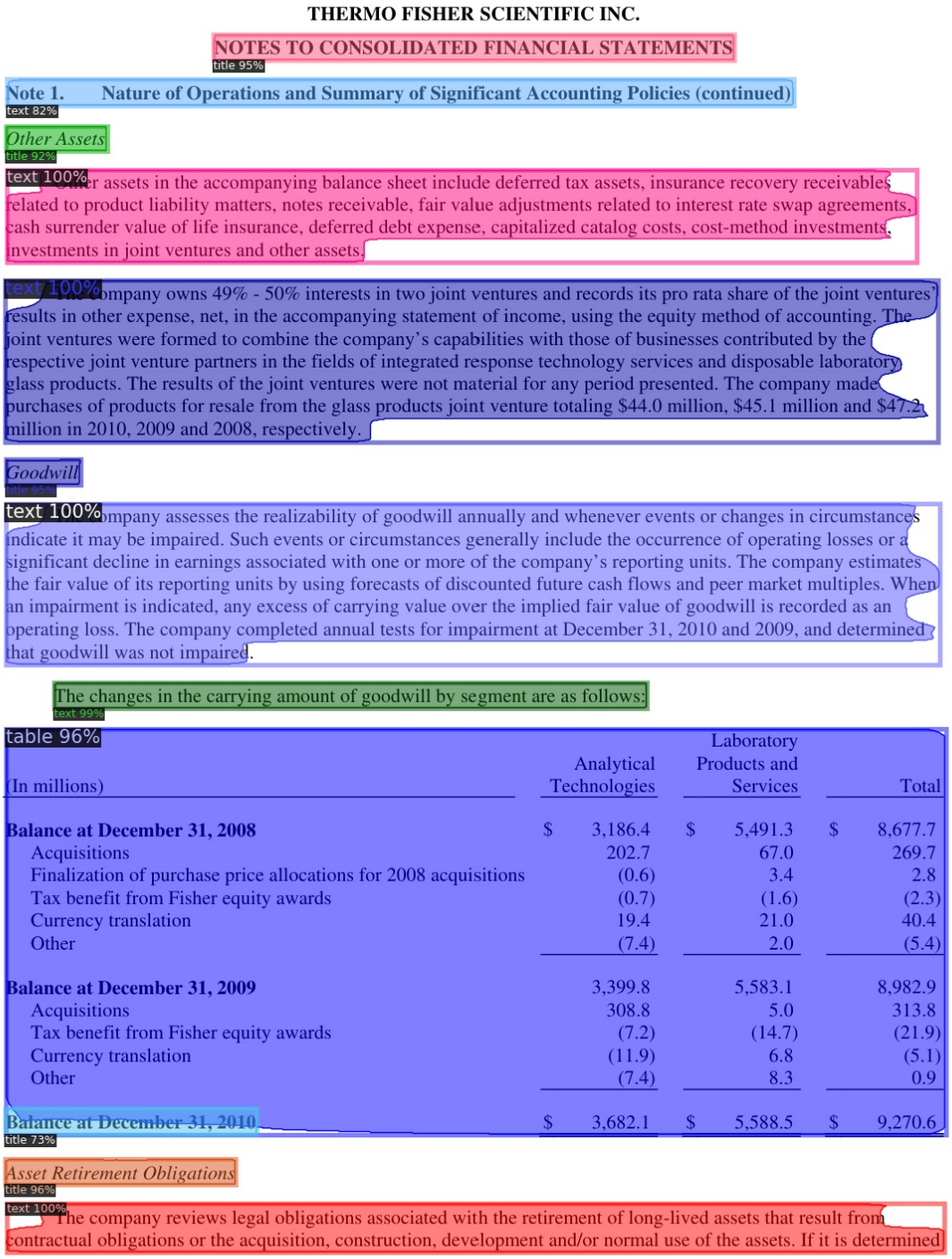
We evaluate the generation performance using the representations with ComTQA's QA set. For this example document, it contains the following QAs:

- Q1: What is the difference between the total amount of assets at the end of 2008 and the total amount of assets at the end of 2009?
- A1: $305.2 million
- Q2: What is the difference between the total amount of assets at the end of 2009 and the total amount of assets at the end of 2010?
- A2: $287.7 million

Figures 35 through 37 display the query engine's outputs across different representations.

E.3 Ablation: VLM Complex Prompt

We introduce Qwen3-VL-8B-Instruct (Complex), which utilizes a prompt designed by GPT-5.4 to parse documents. An excerpt of the prompt is shown in Figure 38. We replace the baseline prompt with the above GPT-5.4 Generated prompt for the VLM. Figure 39 depicts the results. As observed in Table 10, results from Qwen3-VL-8B-Instruct (Complex) are highly inconsistant and poor.



THERMO FISHER SCIENTIFIC INC.

NOTES TO CONSOLIDATED FINANCIAL STATEMENTS

title 95%

Note 1. Nature of Operations and Summary of Significant Accounting Policies (continued)

text 82%

Other Assets

title 92%

text 100%

Other assets in the accompanying balance sheet include deferred tax assets, insurance recovery receivables related to product liability matters, notes receivable, fair value adjustments related to interest rate swap agreements, cash surrender value of life insurance, deferred debt expense, capitalized catalog costs, cost-method investments, investments in joint ventures and other assets.

text 100%

The company owns 49% - 50% interests in two joint ventures and records its pro rata share of the joint ventures' results in other expense, net, in the accompanying statement of income, using the equity method of accounting. The joint ventures were formed to combine the company's capabilities with those of businesses contributed by the respective joint venture partners in the fields of integrated response technology services and disposable laboratory glass products. The results of the joint ventures were not material for any period presented. The company made purchases of products for resale from the glass products joint venture totaling $44.0 million, $45.1 million and $47.2 million in 2010, 2009 and 2008, respectively.

Goodwill

title 95%

text 100%

The company assesses the realizability of goodwill annually and whenever events or changes in circumstances indicate it may be impaired. Such events or circumstances generally include the occurrence of operating losses or a significant decline in earnings associated with one or more of the company's reporting units. The company estimates the fair value of its reporting units by using forecasts of discounted future cash flows and peer market multiples. When an impairment is indicated, any excess of carrying value over the implied fair value of goodwill is recorded as an operating loss. The company completed annual tests for impairment at December 31, 2010 and 2009, and determined that goodwill was not impaired.

The changes in the carrying amount of goodwill by segment are as follows:

text 99%

table 96%

(In millions)	Analytical Technologies	Laboratory Products and Services	Total
Balance at December 31, 2008	$ 3,186.4	$ 5,491.3	$ 8,677.7
Acquisitions	202.7	67.0	269.7
Finalization of purchase price allocations for 2008 acquisitions	(0.6)	3.4	2.8
Tax benefit from Fisher equity awards	(0.7)	(1.6)	(2.3)
Currency translation	19.4	21.0	40.4
Other	(7.4)	2.0	(5.4)
Balance at December 31, 2009	3,399.8	5,583.1	8,982.9
Acquisitions	308.8	5.0	313.8
Tax benefit from Fisher equity awards	(7.2)	(14.7)	(21.9)
Currency translation	(11.9)	6.8	(5.1)
Other	(7.4)	8.3	0.9
Balance at December 31, 2010	$ 3,682.1	$ 5,588.5	$ 9,270.6

title 73%

Asset Retirement Obligations

title 96%

text 100%

The company reviews legal obligations associated with the retirement of long-lived assets that result from contractual obligations or the acquisition, construction, development and/or normal use of the assets. If it is determined

Figure 27: An example layout segmentation output from ComTQA.

Prompt: TABRAG$_{MD}$ (Markdown Table) Self-Generated ICL

```
You are a precise information parsing and extraction engine. Output ONLY
A Markdown table of the provided image

Task: Generate a Markdown table with the correct structure, with every single cell accounted for.

Rules:
1. Output ONLY a valid Markdown table. Do not include any other text.
2. Preserve the exact number of rows and columns visible in the image.
3. Preserve unit information in headers if present (e.g., "$ in millions").
4. Transcribe all cell contents as faithfully as possible without inference, normalization, or computation.
5. Repeat values for merged cells so that every row has the same number of columns.
6. If a cell is visually empty, leave it empty.
7. Ignore captions, footnotes, and non-table elements outside the table.


---

**Example Output 1: Two-level header**

| ($ in millions) | 2024              | 2023              |
|-----------------|-------------------|-------------------|
|                 | Revenue | Profit | Revenue | Profit |
| Sales           | 1,234   | 400    | 1,200   | 350    |
| Net Income      | (56)    | 80     | -40     | 70     |

---

**Example Output 2: Three-level header**

| ($ in thousands) | 2024                                  | 2023                                  |
|------------------|---------------------------------------|---------------------------------------|
|                  | Q1                | Q2                | Q1                | Q2                |
|                  | Revenue | Profit  | Revenue | Profit | Revenue | Profit | Revenue | Profit |
| Product A        | 500     | 120     | 600     | 150    | 450     | 100    | 550     | 140    |
| Product B        | (50)    | 80      | (30)    | 100    | -20     | 60     | 10      | 90     |

---

**Example Output 3: Mixed 1-row, 2-row, and 3-row headers**

| Category              | 2024                                  | 2023              | Growth % | Notes    |
|----------------------|---------------------------------------|-------------------|----------|----------|
|                      | Q1                | Q2                | Revenue | Profit |          |          |
|                      | Revenue | Profit | Revenue | Profit |         |        |          |          |
| Sales (Units)        | 1,000   | 300    | 900     | 250    | 1,700   | 550    | 12%      | N/A      |
| Cost ($ in thousands) | (200)   | (50)   | -180    | -40    | (380)   | (90)   | N/A      | Adjusted |

---

Now, generate the Markdown table ONLY, do not include any other irrelevant text.
```

Figure 28: Prompt of generating ICL examples in Markdown representation.

Prompt: TABRAG$_{MD}$ (Markdown Table)

```
You are a precise information parsing and extraction engine. Output ONLY a Markdown table of the provided image.

Task: Generate a Markdown table with the correct structure, with every single cell accounted for.

Rules:
1. Output ONLY a valid Markdown table. Do not include any other text.
2. Preserve the exact number of rows and columns visible in the image.
3. Preserve unit information in headers if present (e.g., "$ in millions").
4. Transcribe all cell contents as faithfully as possible without inference, normalization, or computation.
5. Repeat values for merged cells so that every row has the same number of columns.
6. If a cell is visually empty, leave it empty.
7. Ignore captions, footnotes, and non-table elements outside the table.

---

{``Self-Generated ICL (Markdown)''}

Now, generate the Markdown table ONLY. DO NOT OUTPUT ANYTHING ELSE.
```

Figure 29: Prompt of extracting table component in mark down representation from self-generated ICL(Markdown).

Prompt: TABRAG$_{\text{TEXT}}$ (Text Table) Self-Generated ICL

```
You are an expert document analyst specializing in structured data verbalization.
Your task is to extract every visible cell from the attached table and convert it into a clear, factual, natural language
      description.

**Task:** Generate a list of descriptive sentences representing every data point in the table.

**Rules for Generation:**
1.  **Format:** Output a bulleted list of natural language sentences.
2.  **Narrative Flow:** Use conversational connectors like "For...", "the data shows...", "recorded a value of...", or "
      was found to be...".
3.  **Hierarchy:** Incorporate nested headers naturally into the sentence (e.g., "Under the 2024 results for Q1, the
      Revenue was...").
4.  **Preserve Accuracy:** You must mention every specific row, column, and value. Do not skip any cells. Keep all
      original symbols like ( ), $, and %.
5.  **Contextual Units:** Integrate any units (like "in millions") directly into the description of the value.
6.  **No Rigid Patterns:** Avoid repeating the exact same sentence structure for every line; vary the phrasing slightly to
       maintain a natural, human-readable tone.

---

**Example 1: Two-level header**

Input:
| ($ in millions) | 2024              | 2023              |
|-----------------|-------------------|-------------------|
|                 | Revenue | Profit | Revenue | Profit |
| Sales           | 1,234   | 400    | 1,200   | 350    |
| Net Income      | (56)    | 80     | -40     | 70     |

Output:
[
The Sales Revenue in 2024 was 1,234 million dollars.
the Sales Profit in 2024 was 400 million dollars.
the Sales Revenue in 2023 was 1,200 million dollars.
the Sales Profit in 2023 was 350 million dollars.
the Net Income Revenue in 2024 was (56) million dollars.
the Net Income Profit in 2024 was 80 million dollars.
the Net Income Revenue in 2023 was -40 million dollars.
the Net Income Profit in 2023 was 70 million dollars.
]
---

**Example 2: Three-level header**

Input:
| ($ in thousands) | 2024                                  | 2023                                  |
|------------------|---------------------------------------|---------------------------------------|
|                  | Q1                | Q2                | Q1                | Q2                |
|                  | Revenue | Profit  | Revenue | Profit | Revenue | Profit | Revenue | Profit |
| Product A        | 500     | 120     | 600     | 150    | 450     | 100    | 550     | 140    |
| Product B        | (50)    | 80      | (30)    | 100    | -20     | 60     | 10      | 90     |

Output:
[
The Q1 Revenue for Product A in 2024 was 500 thousand dollars.
The Q1 Profit for Product A in 2024 was 120 thousand dollars.
The Q2 Revenue for Product A in 2024 was 600 thousand dollars.
The Q2 Profit for Product A in 2024 was 150 thousand dollars.
The Q1 Revenue for Product A in 2023 was 450 thousand dollars.
The Q1 Profit for Product A in 2023 was 100 thousand dollars.
The Q2 Revenue for Product A in 2023 was 550 thousand dollars.
The Q2 Profit for Product A in 2023 was 140 thousand dollars.
The Q1 Revenue for Product B in 2024 was (50) thousand dollars.
The Q1 Profit for Product B in 2024 was 80 thousand dollars.
The Q2 Revenue for Product B in 2024 was (30) thousand dollars.
The Q2 Profit for Product B in 2024 was 100 thousand dollars.
The Q1 Revenue for Product B in 2023 was -20 thousand dollars.
The Q1 Profit for Product B in 2023 was 60 thousand dollars.
The Q2 Revenue for Product B in 2023 was 10 thousand dollars.
The Q2 Profit for Product B in 2023 was 90 thousand dollars.
]

---

**Example 3: Mixed 1-row, 2-row, and 3-row headers**

...

---

Now, extract every cell from the table image into this natural language format ONLY. DO NOT OUTPUT ANYTHING ELSE.
```

Figure 30: Prompt of generating ICL examples in plain text representation.

Prompt: TABRAG$_{\text{TEXT}}$ (Text Table)

```
You are an expert document analyst specializing in structured data verbalization.
Your task is to extract every visible cell from the attached table and convert it into a clear, factual, natural language
      description.

**Task:** Generate a list of descriptive sentences representing every data point in the table.

**Rules for Generation:**
1.  **Format:** Output a bulleted list of natural language sentences.
2.  **Narrative Flow:** Use conversational connectors like "For...", "the data shows...", "recorded a value of...", or "
      was found to be...".
3.  **Hierarchy:** Incorporate nested headers naturally into the sentence (e.g., "Under the 2024 results for Q1, the
      Revenue was...").
4.  **Preserve Accuracy:** You must mention every specific row, column, and value. Do not skip any cells. Keep all
      original symbols like ( ), $, and %.
5.  **Contextual Units:** Integrate any units (like "in millions") directly into the description of the value.
6.  **No Rigid Patterns:** Avoid repeating the exact same sentence structure for every line; vary the phrasing slightly to
       maintain a natural, human-readable tone.

---

{``Self-Generated ICL (Text)''}

Now, extract every cell from the table image into this natural language format. DO NOT OUTPUT ANYTHING ELSE.
```

Figure 31: Prompt of extracting table component in plain text representation from self-generated ICL(TEXT).

Output: TABRAG$_{\text{JSON}}$ (Table)


```
{"row": "Balance at December 31, 2008", "column": "Analytical Technologies", "value": "$ 3,186.4", "units": "million"}
{"row": "Balance at December 31, 2008", "column": "Laboratory Products and Services", "value": "$ 5,491.3", "units": "
      million"}
{"row": "Balance at December 31, 2008", "column": "Total", "value": "$ 8,677.7", "units": "million"}
{"row": "Acquisitions", "column": "Analytical Technologies", "value": "202.7", "units": "million"}
{"row": "Acquisitions", "column": "Laboratory Products and Services", "value": "67.0", "units": "million"}
{"row": "Acquisitions", "column": "Total", "value": "269.7", "units": "million"}
{"row": "Finalization of purchase price allocations for 2008 acquisitions", "column": "Analytical Technologies", "value":
      "(0.6)",
"units": "million"}
{"row": "Finalization of purchase price allocations for 2008 acquisitions", "column": "Laboratory Products and Services",
      "value":
"3.4", "units": "million"}
{"row": "Finalization of purchase price allocations for 2008 acquisitions", "column": "Total", "value": "2.8", "units": "
      million"}
{"row": "Tax benefit from Fisher equity awards", "column": "Analytical Technologies", "value": "(0.7)", "units": "million
      "}
{"row": "Tax benefit from Fisher equity awards", "column": "Laboratory Products and Services", "value": "(1.6)",
"units": "million"}
{"row": "Tax benefit from Fisher equity awards", "column": "Total", "value": "(2.3)", "units": "million"}
{"row": "Currency translation", "column": "Analytical Technologies", "value": "19.4", "units": "million"}
{"row": "Currency translation", "column": "Laboratory Products and Services", "value": "21.0", "units": "million"}
{"row": "Currency translation", "column": "Total", "value": "40.4", "units": "million"}
{"row": "Other", "column": "Analytical Technologies", "value": "(7.4)", "units": "million"}
{"row": "Other", "column": "Laboratory Products and Services", "value": "2.0", "units": "million"}
{"row": "Other", "column": "Total", "value": "(5.4)", "units": "million"}
{"row": "Balance at December 31, 2009", "column": "Analytical Technologies", "value": "3,399.8", "units": "million"}
{"row": "Balance at December 31, 2009", "column": "Laboratory Products and Services", "value": "5,583.1", "units": "
      million"}
{"row": "Balance at December 31, 2009", "column": "Total", "value": "8,982.9", "units": "million"}
{"row": "Acquisitions", "column": "Analytical Technologies", "value": "308.8", "units": "million"}
{"row": "Acquisitions", "column": "Laboratory Products and Services", "value": "5.0", "units": "million"}
{"row": "Acquisitions", "column": "Total", "value": "313.8", "units": "million"}
{"row": "Tax benefit from Fisher equity awards", "column": "Analytical Technologies", "value": "(7.2)", "units": "million
      "}
{"row": "Tax benefit from Fisher equity awards", "column": "Laboratory Products and Services", "value": "(14.7)",
"units": "million"}
{"row": "Tax benefit from Fisher equity awards", "column": "Total", "value": "(21.9)", "units": "million"}
{"row": "Currency translation", "column": "Analytical Technologies", "value": "(11.9)", "units": "million"}
{"row": "Currency translation", "column": "Laboratory Products and Services", "value": "6.8", "units": "million"}
{"row": "Currency translation", "column": "Total", "value": "(5.1)", "units": "million"}
{"row": "Other", "column": "Analytical Technologies", "value": "(7.4)", "units": "million"}
{"row": "Other", "column": "Laboratory Products and Services", "value": "8.3", "units": "million"}
{"row": "Other", "column": "Total", "value": "0.9", "units": "million"}
{"row": "Balance at December 31, 2010", "column": "Analytical Technologies", "value": "$ 3,682.1", "units": "million"}
{"row": "Balance at December 31, 2010", "column": "Laboratory Products and Services", "value": "$ 5,588.5", "units": "
      million"}
{"row": "Balance at December 31, 2010", "column": "Total", "value": "$ 9,270.6", "units": "million"}
```


Figure 32: Output of extracted table component in JSON representation from TABRAG$_{\text{JSON}}$.

Output: TABRAG$_{\text{MD}}$ (Table)


```
| | Analytical Technologies | Laboratory Products and Services | Total |\n|---|---|---|---|\n| (In millions) | | | |\n|
      Balance at
December 31, 2008 | $ 3,186.4 | $ 5,491.3 | $ 8,677.7 |\n| Acquisitions | 202.7 | 67.0 | 269.7 |\n| Finalization of
      purchase
price allocations for 2008 acquisitions | (0.6) | 3.4 | 2.8 |\n| Tax benefit from Fisher equity awards | (0.7) | (1.6) |
(2.3) |\n| Currency translation | 19.4 | 21.0 | 40.4 |\n| Other | (7.4) | 2.0 | (5.4) |\n| Balance at December 31, 2009 |
3,399.8 | 5,583.1 | 8,982.9 |\n| Acquisitions | 308.8 | 5.0 | 313.8 |\n| Tax benefit from Fisher equity awards | (7.2) |
(14.7) | (21.9) |\n| Currency translation | (11.9) | 6.8 | (5.1) |\n| Other | (7.4) | 8.3 | 0.9 |\n| Balance at December
      31,
2010 | $ 3,682.1 | $ 5,588.5 | $ 9,270.6 |\nBalance at December 31, 2010}
```


Figure 33: Output of extracted table component in Markdown representation from TABRAG$_{\text{MD}}$.

Output: TABRAG$_{\text{TEXT}}$ (Table)

```
The Balance at December 31, 2008 for Analytical Technologies was $3,186.4 million.
The Balance at December 31, 2008 for Laboratory Products and Services was $5,491.3 million.
The Balance at December 31, 2008 for the Total was $8,677.7 million.
The Acquisitions for Analytical Technologies in 2008 were $202.7 million.
The Acquisitions for Laboratory Products and Services in 2008 were $67.0 million.
The Acquisitions for the Total in 2008 were $269.7 million.
The Finalization of purchase price allocations for 2008 acquisitions for Analytical Technologies was $(0.6) million.
The Finalization of purchase price allocations for 2008 acquisitions for Laboratory Products and Services was $3.4 million
      .
The Finalization of purchase price allocations for 2008 acquisitions for the Total was $2.8 million.
The Tax benefit from Fisher equity awards for Analytical Technologies in 2008 was $(0.7) million.
The Tax benefit from Fisher equity awards for Laboratory Products and Services in 2008 was $(1.6) million.
The Tax benefit from Fisher equity awards for the Total in 2008 was $(2.3) million.
The Currency translation for Analytical Technologies in 2008 was $19.4 million.
The Currency translation for Laboratory Products and Services in 2008 was $21.0 million.
The Currency translation for the Total in 2008 was $40.4 million.
The Other for Analytical Technologies in 2008 was $(7.4) million.
The Other for Laboratory Products and Services in 2008 was $2.0 million.
The Other for the Total in 2008 was $(5.4) million.
The Balance at December 31, 2009 for Analytical Technologies was $3,399.8 million.
The Balance at December 31, 2009 for Laboratory Products and Services was $5,583.1 million.
The Balance at December 31, 2009 for the Total was $8,982.9 million.
The Acquisitions for Analytical Technologies in 2009 were $308.8 million.
The Acquisitions for Laboratory Products and Services in 2009 were $5.0 million.
The Acquisitions for the Total in 2009 were $313.8 million.
The Tax benefit from Fisher equity awards for Analytical Technologies in 2009 was $(7.2) million.
The Tax benefit from Fisher equity awards for Laboratory Products and Services in 2009 was $(14.7) million.
The Tax benefit from Fisher equity awards for the Total in 2009 was $(21.9) million.
The Currency translation for Analytical Technologies in 2009 was $(11.9) million.
The Currency translation for Laboratory Products and Services in 2009 was $6.8 million.
The Currency translation for the Total in 2009 was $(5.1) million.
The Other for Analytical Technologies in 2009 was $(7.4) million.
The Other for Laboratory Products and Services in 2009 was $8.3 million.
The Other for the Total in 2009 was $0.9 million.
The Balance at December 31, 2010 for Analytical Technologies was $3,682.1 million.
The Balance at December 31, 2010 for Laboratory Products and Services was $5,588.5 million.
The Balance at December 31, 2010 for the Total was $9,270.6 million.
```

Figure 34: Output of extracted table component in plain text representation from TABRAG$_{\text{TEXT}}$.

Query Engine Output: TABRAG$_{\text{JSON}}$

```
Question 1 (Correct):
The difference between the total amount of assets at the end of 2008 and the end of 2009 is calculated by subtracting the
      total
assets at the end of 2008 from the total assets at the end of 2009.
Total assets at the end of 2008: $8,677.7 million
Total assets at the end of 2009: $8,982.9 million
Difference: $8,982.9 million - $8,677.7 million = $305.2 million

Question 2 (Correct): The difference between the total amount of assets at the end of 2009 and the end of 2010 is
      calculated by
subtracting the total assets at the end of 2009 from the total assets at the end of 2010.

Total assets at the end of 2009: $8,982.9 million
Total assets at the end of 2010: $9,270.6 million
Difference: $9,270.6 million - $8,982.9 million = $287.7 million
```

Figure 35: Query engine output: The accurate representation from TABRAG$_{\text{JSON}}$ results in correct answers for both financial comparison questions

Query Engine Output: TABRAG$_{\text{MD}}$

```
Question 1 (Incorrect):
The provided information does not include the total amount of assets at the end of 2008 and 2009, making it impossible to
      calculate
the difference. Therefore, the answer is not available from the given data.

Question 2 (Incorrect):
The provided information does not include the total amount of assets at the end of 2009 and 2010, making it impossible to
      calculate
the difference between the two amounts. The data available only provides details about goodwill and other specific assets,
       not the
overall total assets for the years 2009 and 2010.
```

Figure 36: Query engine output: Inadequate context generation from structural representation of TABRAG$_{\text{MD}}$ results in incorrect answers for both financial comparison questions, where essential financial metrics are missed.

Query Engine Output: TABRAG$_{\text{Text}}$

```
Question 1 (Incorrect):
The difference between the total amount of assets at the end of 2008 and the end of 2009 cannot be directly calculated
      from the
provided information. The text provided focuses on specific components of assets, such as goodwill, joint ventures, and
      other
assets, but does not provide the total assets for the end of 2008 or 2009. To calculate the difference, the total assets
      for both
years would need to be known, which are not explicitly provided in the given text.

Question 2 (Incorrect):
The difference between the total amount of assets at the end of 2009 and the end of 2010 cannot be directly calculated
      from the
provided information. The text provided focuses on specific components of assets, such as goodwill, joint ventures, and
      other
assets, but does not provide the total assets for the end of 2009 or 2010. To calculate the difference, the total assets
      for both
years would need to be known, which are not explicitly provided in the text.
```

Figure 37: Query engine output: Inadequate context generation from structural representation of TABRAG$_{\text{TEXT}}$ results in incorrect answers for both financial comparison questions, where essential financial metrics are missed.

Prompt to GPT-5.4 for VLM Prompt Design

```
Please design a "structure-aware prompt" for a VLM with "explicit output schemas, and few-shot exemplars" (no images) to
parse documents for a RAG pipeline. The documents can contain tables. Please output the prompt in Markdown.
```

Figure 38: Prompt used to ask GPT-5.4 to design the complex VLM parsing prompt.

GPT-5.4 Generated Prompt

```
# SYSTEM ROLE
You are a specialized Document Parsing Agent. Your goal is to convert document images into a highly structured, machine-
      readable format optimized for a Retrieval-Augmented Generation (RAG) indexing pipeline. You must preserve the
      document's semantic hierarchy and represent complex elements like tables with high fidelity.

# OUTPUT SCHEMA
You must return only a valid JSON object following this schema:
‘‘‘json
{
  "metadata": {
    "page_number": "number",
    "primary_header": "string",
    "document_type": "string"
  },
  "chunks": [
    {
      "id": "integer",
      "type": "heading | text | table | list",
      "level": "h1 | h2 | h3 | body | null",
      "content": "string",
      "contextual_summary": "A 1-sentence summary of this chunk for semantic retrieval enhancement"
    }
  ]
}
‘‘‘

# EXTRACTION RULES
1. **Structural Awareness**: Respect the visual hierarchy. Use the ‘level‘ field to denote the depth of headings.
2. **Table Parsing**: Convert all tables into valid Markdown table strings within the ‘content‘ field. Ensure headers are
      correctly aligned.
3. **Reading Order**: Detect multi-column layouts and parse in the correct logical reading order (typically top-to-bottom,
       left-to-right).
4. **Lists**: Maintain bulleted or numbered list formatting within the ‘content‘ string.
5. **Cleanliness**: Remove artifacts like page numbers (unless in metadata), running footers, or watermarks from the ‘
      content‘ field.

# FEW-SHOT EXEMPLARS

### Exemplar 1: Standard Text & Headers
**Input Document Description**: A page titled "Safety Protocols" with a section on "Fire Hazards."
**Expected Output**:
‘‘‘json
{
  ... [omitted]
}
‘‘‘

### Exemplar 2: Document with Table
**Input Document Description**: A financial report snippet showing a revenue table.
**Expected Output**:
‘‘‘json
{
  ... [omitted]
}
‘‘‘

# TASK
Analyze the provided document image and output the extracted content according to the defined JSON schema. Ensure all
tables are converted to Markdown within the JSON structure.
```

Figure 39: Complex VLM parsing prompt generated by GPT-5.4 for the prompt ablation.

Models	Accuracy (%) ↑						Average
	TAT-DQA	MP-DocVQA	WikiTQ	TableVQA	ComTQA	FinQA	
Qwen3-VL-8B-Instruct	$66.26_{\pm.001}$	$85.81_{\pm.002}$	$61.71_{\pm.005}$	$66.29_{\pm.002}$	$67.80_{\pm.001}$	$66.07_{\pm.002}$	68.99
Qwen3-VL-8B-Instruct (Complex)	$89.50_{\pm.002}$	$13.59_{\pm.001}$	$15.46_{\pm.003}$	$23.00_{\pm.003}$	$84.20_{\pm.001}$	$64.69_{\pm.002}$	48.41
TabRAG$_{8B}$	**$92.56_{\pm.002}$**	**$88.87_{\pm.003}$**	**$68.49_{\pm.003}$**	**$70.60_{\pm.004}$**	**$82.60_{\pm.002}$**	**$71.55_{\pm.001}$**	**79.11**

Table 10: Average generation accuracy (%) of baseline prompts. We compare a LlamaIndex prompt (Liu, 2022) against a "complex" prompt designed by GPT-5.4.

F Artifact Use, Data Privacy, and Intended Use

We use existing public document question-answering datasets, pretrained vision-language and language models, embedding models, and vector-store libraries for research and evaluation purposes. We use these artifacts consistently with their intended research use and cite the corresponding sources. Any derived artifacts produced from these datasets, including parsed structured representations, self-generated in-context examples, or vector-store indexes, should be used only in accordance with the access conditions and licenses of the original datasets and models, and should not be deployed beyond research contexts unless such use is explicitly permitted. When applying TabRAG to sensitive or proprietary documents, users should ensure appropriate anonymization, access control, and data-governance procedures before processing or indexing the data.

We do not collect new human-subject data or intentionally introduce personally identifying information or offensive content. Our experiments use existing benchmark datasets as released by their original creators. We do not attempt to deanonymize individuals, link records across datasets, or infer sensitive attributes. Since TabRAG is designed to parse and index document contents, any application to sensitive or proprietary documents should include appropriate screening for personally identifying information, anonymization where necessary, access control, and secure data-governance procedures before processing or indexing the data.