Proceedings of Machine Learning Research **318** (2026)
The 39th Canadian Conference on Artificial Intelligence

Canadian AI 2026
25–29 May 2026, Vancouver, BC

A Multimodal Data Extraction Pipeline with Table Layout Correction

Kecen Yao[†,‡], Anton Shesternev[†], Ahmad Pesaranghader[†], Erin Li[†*]
[†] CIBC
[‡] University of Toronto

Abstract

Financial documents such as paystubs, invoices, and financial statements contain heterogeneous layouts and visually complex tables, making reliable information extraction challenging for both optical character recognition (OCR) based pipelines and end-to-end vision–language models (VLMs). In this paper, we present a pipeline that unifies layout analysis, one-shot multimodal table correction, and downstream extraction and reasoning without any model fine-tuning. The pipeline converts document images into a hybrid Markdown-HTML representation and applies a multi-modal correction module to rectify layout-level errors in tables, yielding demonstrable improvements in Tree-Edit-Distance-based Similarity (TEDS) scores. Additionally, using this corrected representation, the system performs robust schema-based extraction and document-level question answering. Experimental results across paystub field extraction and Finance question-answering (QA) tasks show that our approach consistently outperforms both OCR-only pipelines and direct VLM baselines. These results demonstrate that incorporating explicit table layout and multimodal table correction provides a scalable and generalizable path toward robust financial document understanding.

Keywords: Layout-Aware Information Extraction, Vision–Language Models, Multimodal Document Processing, Table Layout Correction, Table Layout Analysis, Financial Document Understanding

1. Introduction

Financial institutions routinely process large volumes of documents such as paystubs, invoices, tax statements, and financial reports. These documents vary widely in layout, typography, and often contain complex elements including handwritten fields, multi-column tables, and embedded graphics [1]. Extracting information from such heterogeneous sources is essential for workflows like credit assessment, compliance auditing, and financial reporting, where accuracy is critical. In practice, many extraction errors originate from complex table layout within financial documents.

Because images and text belong to different modalities, document-image information extraction is typically handled through a two-stage process. A visual document is first converted into an intermediate representation, plain text, layout-aware text, or embeddings, and extraction is then performed over this representation [2–4]. Common examples include OCR text followed by an information extractor, layout models that output layout-aware text consumed by a downstream extractor, or vision–language models (VLMs) that encode the image and directly decode answers. (Further details are discussed in Section 2.)

The effectiveness of these approaches ultimately depends on the fidelity of the intermediate representation: when the representation faithfully preserves layout (e.g., borders, merged cells, header positions, etc.) and layout-level context (table entries), downstream extraction is significantly more accurate. However, existing layout models frequently introduce layout errors such as misaligned rows, fragmented cells, or incorrect table boundaries.

This exposes a critical gap in current methods: prior work does not reliably correct layout errors, particularly table-related, while these errors often propagate into the extraction stage.

* Erin.Li@cibc.com

Motivated by this observation, we introduce a simple yet effective idea: use a VLM to correct table layout errors produced by layout models, repairing misaligned or fragmented tables without VLM retraining. The corrected layout provides a more faithful representation of the document and substantially improves downstream extraction accuracy.

Building on this idea, we propose a fine-tuning-free pipeline for information extraction from financial document images. The pipeline consists of three components: (1) a layout detection module that converts page images into a hybrid Markdown–HTML representation; (2) a one-shot multimodal table-correction module that visually grounds and repairs erroneous HTML tables using a VLM; (3) and an information-extraction module that performs schema-driven field extraction and document-level question answering over the corrected layout representation (see Section 3, Figure 1).

We evaluate this pipeline across two representative financial-document tasks: key–value extraction from Paystub dataset [5], and document-level question answering on the FinanceQA dataset [6]. To further validate generalizability, we also evaluate the table-correction module on PubTabNet [7], demonstrating table correction across diverse layouts. Additional results are presented in Section 4. Together, these findings provide comprehensive empirical evidence that multimodal table correction delivers substantial accuracy gains and strong cross-domain generalization for financial document understanding.

2. Related Work

Most document-image extraction methods follow a two-stage pipeline in which the visual page is first converted into an intermediate representation, text, layout markup, or embeddings, before downstream reasoning occurs. This section expands on these major modeling paradigms and examines their respective limitations.

2.1. OCR-Based Pipelines

Traditional OCR-based pipelines convert document images into plain text before downstream processing. Classic OCR engines such as Tesseract [8] and commercial pipelines (e.g., ABBYY FineReader [9], Amazon Textract [10]) focus primarily on textual transcription, discarding much of the spatial and visual information that encodes document element's layout. This limitation is particularly severe for tabular elements, where semantic relations are visually organized rather than linearly encoded. As a result, downstream extractors, typically rule-based methods, regex templates, or natural language processing (NLP) models, face difficulty capturing row–column relationships, merged cells, and hierarchical table layouts. OCR noise and formatting inconsistencies further propagate errors into entity extraction and key-value retrieval tasks. This motivates approaches that preserve both textual and layout cues for tables.

2.2. Vision–Language Models for Document Understanding and Information Extraction

Attention-based transformer architectures have driven major advances in multimodal document understanding [11]. Models such as LayoutLM [3], LayoutLMv2 [12], and LayoutLMv3 [13] jointly encode text, spatial coordinates, and visual cues, substantially improving tasks such as form understanding, key information extraction, and receipt parsing. Later architectures, including DocFormer [14], further unified these text–layout–vision information streams.

Despite these gains, transformer-based layout models still rely heavily on OCR-derived text tokens. This dependency exposes them to OCR noise, transcription errors, and domain

shift issues that become particularly severe in visually dense financial documents containing irregular tables, nested headers, and handwritten adjustments.

General-purpose vision–language models (VLMs), including LLaVA [15] and vision-enabled GPT-4o [16], process document images end-to-end without relying on a separate OCR engine and can thus mitigate some limitations of traditional OCR-based pipelines. In contrast, document- and UI-focused models such as Donut [17] and Pix2Struct [18] are explicitly trained to map text-heavy images to pre-defined format (e.g., JSON or HTML), providing task-specific alternatives to conventional OCR-plus-extractor workflows. However, existing VLMs struggle with fine-grained document understanding, particularly when tasks require accurate integration of textual content and layout. A prior study [19] shows that VLMs often underperform on reading-intensive or layout-dependent tasks due to insufficient resolution and incomplete capture of layout context, limiting their reliability for precise extraction.

2.3. **VLMs for Layout Representations**

A complementary line of work explicitly generates layout document representations, such as Markdown, HTML, or XML. Systems including Smoldocling [20], Nanonets-ocr [21], and commercial engines (e.g., Azure AI Document Intelligence [22], Google Document AI [23], Mistral OCR [24]) convert page images into symbolic markup that is interpretable and supports downstream extraction or reasoning.

However, even these document-to-markup conversion models are vulnerable to layout inconsistency, missing cell boundaries, faulty nested tables, or misaligned merged cells, especially for low-resolution scans or the highly irregular layouts common in financial documents.

3. **System Overview**

Motivated by these limitations, we introduce a one-shot multimodal table-correction module that visually grounds table layout and repairs layout errors. Overall, we propose a pipeline for information extraction from scanned financial documents with heterogeneous layouts, especially for complex tables *without any retraining* of underlying models (see Figure 1). The system comprises three modules: (1) A layout detection module that converts each page into a layout-aware textual artifact, Markdown for text and embedded HTML for tabular elements, preserving both reading order and table layout. (2) A visually grounded table-correction module that repairs layout errors in the HTML while keeping the surrounding Markdown intact. We use the term visually grounded to mean that the correction module jointly processes the original document image alongside the noisy HTML, so that the image serves as the authoritative visual reference (ground truth) for the table's true structure, enabling the model to identify and repair discrepancies between what it sees in the image and what the markup encodes. (3) An information-extraction module that operates on this unified artifact in two modes: (i) Field extraction, producing a normalized JSON record for the declared document type (e.g., paystub fields); and (ii) Document-level question answering (QA), providing concise, source-grounded answers to free-form queries (e.g., "*Which earnings category is largest?*").

In our system, each module runs independently (i.e., as a subsystem), allowing flexible integration. The layout and correction modules are document-agnostic, while the final extraction module is document-specific for different use-cases, implemented as a plug-in for easy extension to new document types or downstream tasks.

3.1. **Layout Analysis and Representation**

Layout detection begins with a page image and produces a single Markdown file that preserves the original document layout. We employ a high-accuracy layout detection service

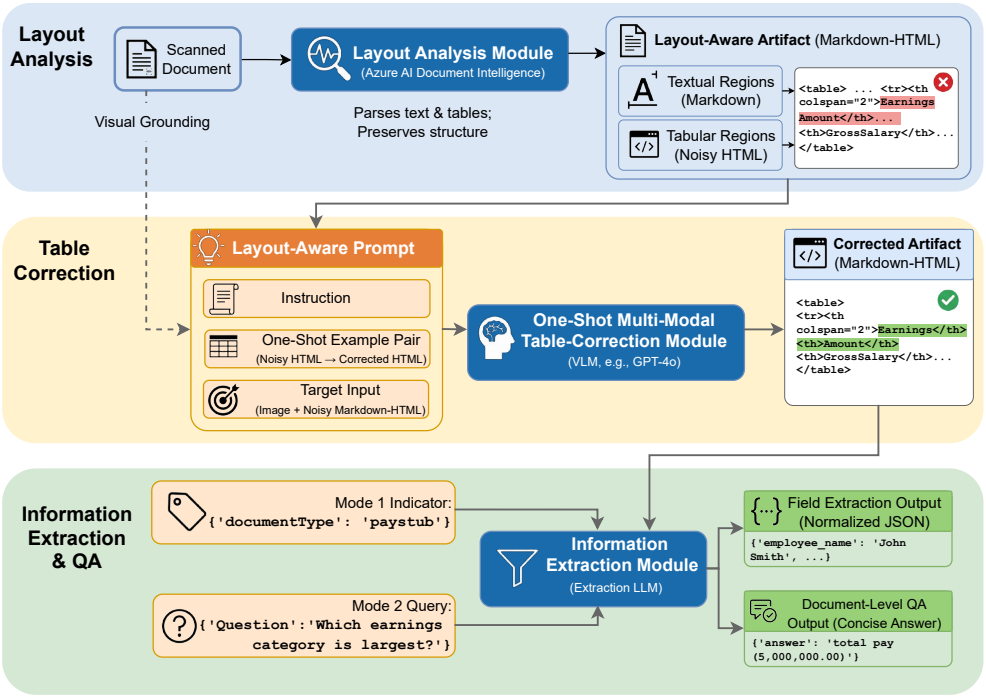

Figure 1. Multi-stage document extraction framework architecture with table correction.

(Azure AI Document Intelligence "layout" model [22]) to detect and parse both text and tables. To select the most reliable model for layout detection, we conducted comparative experiments between open-source and closed-source solutions (see Section 4.2).

In the Markdown file, non-tabular regions are serialized as Markdown format for compactness and readability, while tabular regions are represented as literal HTML *<table>* elements with explicit *<thead>*, *<tbody>*, *<tr>*, *<th>*, and *<td>* tags, preserving *rowspan* and *colspan* attributes. We refer to this format as Markdown-HTML.

This hybrid representation allows complicated table layout and retains layout information that is often lost in plain-text OCR outputs. By embedding HTML tables within Markdown, we obtain a self-contained and deterministic format for downstream processing.

3.2. One-Shot Multi-Modal Table Correction

Despite recent advances, most layout models often produce imperfect table layout when applied to real-world cases. For example, HTML tables generated from the layout model may show column drift, header misalignment, or spurious merges. To address these issues, we use a one-shot correction approach in which a single example of a noisy table and its corrected version, together with the document image, is provided to the VLM as in-context guidance, enabling it to repair alignment errors and restore layout complexity and consistency without any fine-tuning. The input to the correction module is a Markdown–HTML document in which tables are represented as embedded HTML, and the output is a corrected Markdown–HTML document with the same non-tabular Markdown content preserved and only the HTML table segments repaired.

To be precise, this module constructs a concise, layout-aware prompt including: (i) an instruction describing the system role and table correction task; (ii) a one-shot example

(noisy Markdown–HTML and corrected Markdown-HTML pair) for in-context learning; and (iii) the target image with its raw Markdown-HTML representation from the layout detection model. We conducted experiments and prompt-engineering studies to identify the most effective prompt design (see Appendix A for the exact prompt, refer to Section 4.2.3 for experimental results).

A VLM (GPT-4o [16]) consumes this prompt and runs with fixed decoding parameters (`temperature=0.0`, `max_tokens=4096`) to ensure deterministic and reproducible outputs. We selected GPT-4o as a representative high-capability vision–language model to evaluate the effectiveness of the proposed table-correction strategy, rather than to optimize performance for a specific backend.

3.3. Information Extraction and Document-Level QA

The final stage consumes the corrected Markdown from the table correction module and supports two complementary modes:

(1) Field extraction: The model receives the layout artifact and a document-type indicator, returning a strict JSON record with predefined key–value fields.
(2) Document-level QA: The same artifact is used with a QA-related prompt emphasizing concise, evidence-grounded answers. Queries may target global properties (e.g., totals) or relational patterns within tables (e.g., "Which deduction is highest?").

Unifying data element extraction and document QA tasks over a single layout-preserving representation ensures that improvements from table correction directly enhance downstream accuracy for both tasks.

Overall, this design aims to enables a generalizable, interpretable, and scalable document-processing pipeline adaptable to diverse financial domains.

4. Experiments

We next present a comprehensive empirical evaluation of our proposed pipeline. This section outlines the datasets, baselines, and metrics used to assess performance across extraction, QA, and table-correction tasks.

4.1. Experiment Setup

4.1.1. Datasets

We conducted extensive experiments to evaluate the performance and generalization of our proposed multimodal document understanding pipeline with three datasets.

Paystub Dataset. The paystub extraction task utilized a total of 53 scanned paystub images containing 159 annotated key-value pairs. Among these, 10 were anonymized samples from an internal dataset, and 43 were randomly selected from the *Kaggle Personal Financial Dataset for India* [5]. Each document was annotated for three key fields: *employer_name*, *employee_name*, and *gross_income*.

FinanceQA Dataset. For the document-level question answering (QA) task, we evaluated our pipeline on a subset of 154 financial document images with 992 question–answer pairs derived from the *Sujet-Finance-QA-Vision-100k* dataset [6]. Regarding subsampling, given the original dataset's large scale (over 100k samples) and its high diversity in document layouts and content types, we employed a CLIP-based clustering strategy to obtain a representative subset. Specifically, we extracted visual embeddings using CLIP [25] and applied K-means clustering (K=50) to group documents by visual similarity. From each cluster, we strategically sampled representative instances to ensure balanced coverage across layout types. Additionally, because the original dataset contains non-negligible annotation

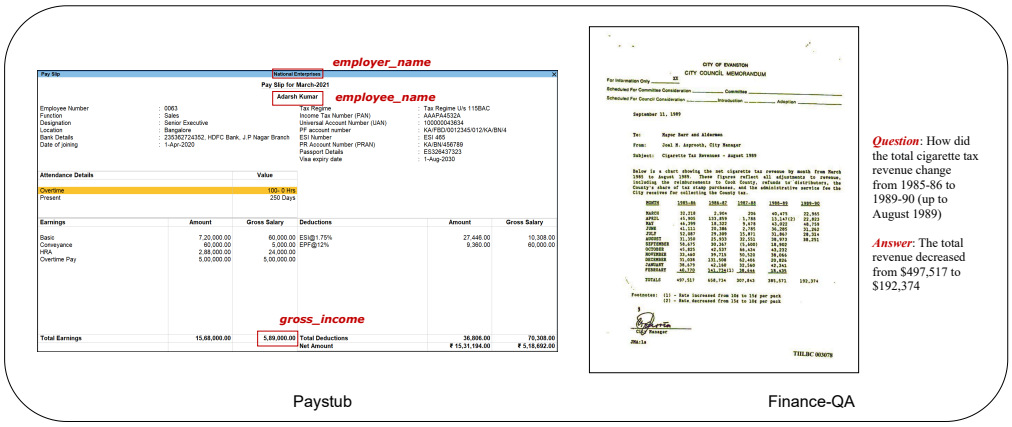

Figure 2. Sample Images from Paystub Dataset and FinanceQA Dataset.

noise, we manually reviewed the selected QA pairs and corrected only clearly invalid or inconsistent annotations (e.g., answers contradicting visible document content or referencing missing fields). This process was applied uniformly across all evaluated pipelines and did not introduce task- or method-specific modifications. No questions or answers were added or removed based on model performance. The final samples cover a wide range of query types, from factual lookups (e.g., *"What is the due date?"*) to relational reasoning (e.g., *"Which deduction category is largest?"*), providing a comprehensive assessment of the model's ability to reason over financial documents.

PubTabNet Dataset. The table correction task was evaluated using 300 complex tables from the *PubTabNet* test split [7]. These image–HTML pairs were selected to include challenging layouts such as nested headers and merged cells.

For all the datasets, we split them into a development set (10%) and a held-out test set (90%). The development set was used only for prompt iteration and refinement. After selecting the final prompt, we evaluated it once on the held-out test set, which was never used during prompt engineering.

4.1.2. Baselines and Comparison Pipelines

To benchmark our proposed multi-stage pipeline, we compared it against both single-stage and multi-stage alternatives, isolating the contribution of each component.

(1) Direct VLM Baseline (Single-Stage). A pure visual baseline using GPT-4o-Vision directly performs document visual question answering (VQA) and field extraction from the image without intermediate layout modeling. Throughout this paper, when the model is used in its vision-enabled configuration (i.e., receiving both an image and text as input), we refer to it as GPT-4o-Vision. This tests whether our multi-stage design yields measurable advantages over end-to-end VLM reasoning.

(2) Two-Stage Baseline (Plain Text + Extraction). A minimal two-stage configuration where the first stage performs OCR-only text extraction using Azure AI Document Intelligence "read" model (Azure Read) [26], followed by GPT-4o for extraction or document QA. This serves as the reference for evaluating layout model quality and its downstream impact.

(3) Enhanced Two-Stage Pipelines (Different Layout Models). We further tested variants using different layout detection backends (both open-source and closed-source), including Azure AI Document Intelligence pre-built "layout" model (Azure Layout) [22],

Nanonets OCR [21], and Mistral OCR [24] to generate the Markdown representations. Among which, Azure Layout and Nanonets OCR produce Markdown-HTML and Mistral produce plain Markdown. All variants share the same extraction and document QA model (GPT-4o), treating it as a fixed black-box component to ensure fair comparison across layout encoders.

(4) Multi-Stage Pipelines (With vs. Without Table Correction). For each available two-stage configuration, we evaluated the effect of integrating the proposed layout-correction module. This ablation measures whether the correction step generalizes across layout models and tasks.

To ensure fair comparison, we use identical extraction and document-level QA prompts across all baselines that invoke GPT-4o. The same prompt templates, output schemas, and answer formatting constraints are applied regardless of whether the input is a raw image (direct VLM baseline), plain OCR text, or a layout-aware Markdown–HTML artifact. The only difference across pipelines is the intermediate representation provided to the extraction model, not the prompting strategy itself.

Additionally, we conducted a prompt-level comparison in the table-correction task to quantify how different prompting strategies influence table correction performance, using the uncorrected layout model output as the baseline.

4.1.3. Evaluation Metrics

Performance was measured using (1) field-level extraction accuracy for data element extraction task, (2) answer accuracy for document-level QA task, and (3) Tree-Edit-Distance-based Similarity (TEDS) [7] for table correction task. TEDS measures tables' HTML similarity by counting the minimal tree-edit operations required to match generated artifact and ground truth. Following the PubTabNet paper, we report two variants: structure-only TEDS (setting `structure_only=True`), which measures the correctness of the table layout independent of cell content, and structure+text TEDS (setting `structure_only=False`), which jointly evaluates both structure and textual accuracy. These complementary metrics together capture robustness, reasoning precision, and layout correctness across the entire pipeline.

4.2. Experimental Results

4.2.1. Data Element Extraction

Table 1 summarizes the extraction accuracy across different pipelines for paystub dataset. The direct VLM approach using GPT-4o-Vision only achieves 97.5% accuracy. Naïve OCR pipelines (Azure Read) achieve around 95% accuracy, limited by unawareness of the layout. Switching to Azure Layout model, which produces layout-aware Markdown-HTML representation, boosts accuracy to 99.3%, indicating that layout information greatly assists large language models in contextual reasoning. Finally, the full pipeline with one-shot multimodal correction module achieves 100% accuracy on this benchmark.

These results show that both layout representation and multimodal correction contribute cumulatively to precision in document-level extraction.

4.2.2. Document Question Answering

We then test the model's ability to answer free-form questions grounded in document content and layout. Table 2 shows results on the manually cleaned Sujet-Finance-QA subset. Vision-only and plain OCR pipelines hover around 66%, constrained by mis-segmented table regions and ambiguous numeric associations. Azure Layout model boosts accuracy to 71.3%, and our complete pipeline with table correction reaches 74.0%, confirming that fine-grained

Category	Pipeline	w/out table correction	w/ table correction
Direct VLM	GPT-4o-Vision	97.5	–
OCR-only	Azure Read + GPT-4o	94.9	–
Layout-aware	Mistral-OCR + GPT-4o	95.6	–
	Nanonets-OCR-s + GPT-4o	95.6	**98.1**
	Azure Layout + GPT-4o	99.3	**100.0**

Table 1. **Data Element Accuracy (%)** comparison on the Paystub dataset. "–" indicates configurations where table correction is not applicable because no explicit table markup (Markdown-HTML) is produced in that pipeline.

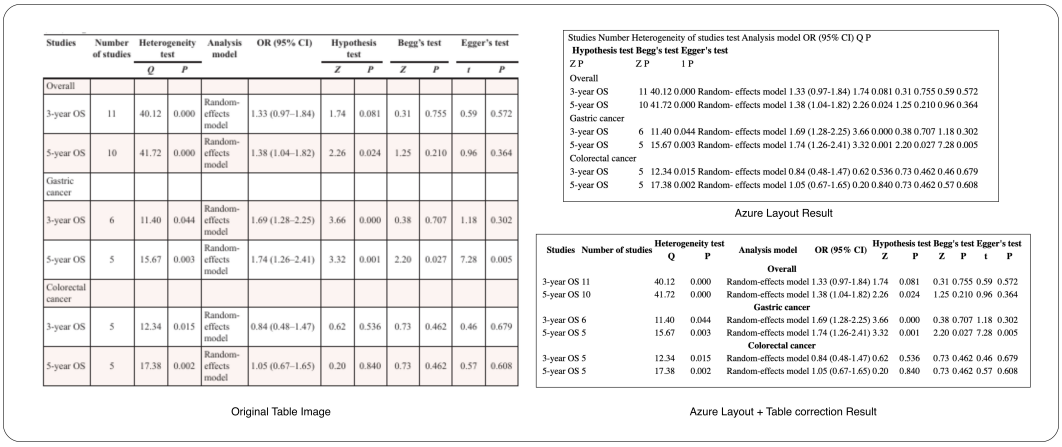

Figure 3. An example of PubTabNet sample: original table image and Azure Layout output before and after multimodal correction.

table correction benefits not only schema-based extraction but also general reasoning and comprehension required in question answering.

Category	Pipeline	w/out table correction	w/ table correction
Direct VLM	GPT-4o-Vision	66.6	–
OCR-only	Azure Read + GPT-4o	66.7	–
Layout-aware	Mistral-OCR + GPT-4o	62.3	–
	Nanonets-OCR-s + GPT-4o	62.3	**65.0**
	Azure Layout + GPT-4o	71.3	**74.0**

Table 2. **Question Answer Accuracy (%)** comparison on the Finance-QA dataset. "–" indicates configurations where table correction is not applicable because no explicit table markup (Markdown-HTML) is produced in that pipeline.

4.2.3. **Multimodal Table Correction Ablation Study**

To isolate and quantify the contribution of the table correction module, we evaluate several correction strategies on PubTabNet dataset. Figure 3 presents a representative example in which the noisy HTML generated by Azure Layout contains layout inconsistencies that could be recovered using multimodal table correction module.

We consider four settings: no correction, text-only correction, multimodal correction, and our final one-shot multimodal approach. The text-only correction setting feeds only the

noisy Markdown–HTML to the VLM model, without providing the table image. In contrast, the multimodal settings allow the VLM model to jointly process the table image and the noisy HTML, enabling it to repair row–column alignment, reconstruct header hierarchies, and recover missing cell spans.

Throughout development, we explored a wide range of prompting strategies and evaluated them on the development set of the dataset. We tested prompts that explicitly defined the task and described common layout errors, as well as prompts that incorporated few-shot examples. We also experimented with a two-step formulation where the model first identifies layout errors and then applies targeted fixes. Different reasoning styles were examined, including directly editing the noisy Markdown–HTML using the image as guidance, and alternatively regenerating a clean Markdown–HTML representation from the image while using the noisy HTML only as a weak reference. After comparing these approaches, we selected and reported the best-performing strategy on the held-out test set. The full prompt is provided in Appendix A.

Correction Method	**TEDS (structure)**	**TEDS (structure + text)**
None (no correction)	0.82	0.69
Text-only correction	0.82	0.69
Multimodal correction	0.86	0.75
One-shot + multimodal correction (ours)	**0.88**	**0.76**

Table 3. **TEDS Scores on the PubTabNet Dataset**.

As shown in Table 3, our best one-shot multimodal approach improves TEDS (structure) from 0.82 to 0.88 and TEDS (structure + text) from 0.69 to 0.76. Importantly, this improvement requires no retraining; all gains arise entirely from in-context prompting with a single example.

5. Discussion

The results across the extraction, QA, and table-correction ablation study collectively demonstrate that our layout-aware pipeline offers clear advantages over both traditional OCR pipelines and direct end-to-end VLM reasoning. In this section, we reflect on the broader implications of these findings, outline current limitations, and discuss considerations for real-world deployment.

5.1. Summary of Key Findings

Our experiments highlight three central findings regarding the role of layout representations and multimodal correction in financial document understanding.

First, one-shot multimodal table correction provides consistent and cross-task benefits on our benchmarks. By jointly using the image as layout ground truth and the noisy HTML as textual guidance, the correction module reliably fixes alignment drift, restores header hierarchies, and produces cleaner HTML tables without any model fine-tuning. These improvements propagate across all downstream tasks: paystub extraction improves from 99.3% to 100%, Finance-QA from 71.3% to 74.0%. Notably, the gains hold even when the input markup comes from open-source layout models (e.g., Nanonets OCR), illustrating that the correction module is model-agnostic.

Second, Markdown–HTML representations substantially outperform plain-text OCR outputs on the evaluated benchmarks. Across both the paystub and FinanceQA benchmarks, layout-aware inputs consistently enable stronger grounding compared with plain OCR pipelines. The preservation of reading order, row–column relationships, and explicit table boundaries

allows GPT-4o to perform extraction and reasoning over a clean layout artifact rather than raw pixels or linearized text. This demonstrates that accurate intermediate layout, rather than more powerful vision encoders alone, is a key determinant of downstream accuracy.

Third, the multi-stage pipeline outperforms monolithic end-to-end VLM reasoning on the tested datasets. Although GPT-4o-Vision provides strong single-stage baselines, it remains less reliable on tasks requiring fine-grained layout understanding or numeric associations. By decoupling layout detection, table correction, and reasoning, our pipeline enables each module to specialize and reduces the cognitive burden on the LLM. This leads to higher accuracy, better interpretability, and easier debugging than direct VLM interpretation of raw images.

Together, these findings confirm that explicit layout and multimodal correction form a powerful foundation for financial document understanding. Because the pipeline operates over generic document and table layout primitives rather than domain-specific semantics, it readily extends to other table-rich documents such as legal contracts and medical records. Its fine-tuning-free design further makes it practical for enterprise deployment, as components can be upgraded independently while retaining system stability.

5.2. **Limitations & Future Work**

While the pipeline performs strongly across tasks, several limitations remain. First, our evaluation uses relatively small subsets for paystub extraction and Finance-QA, which may limit statistical robustness; future work will scale to larger public splits and report uncertainty estimates (e.g., confidence intervals) to better characterize variance. Second, the current instantiation relies on proprietary components (Azure Layout [22] and GPT-4o [16]), and our baselines do not yet include recent fine-tuned document models; we plan to add a fully open-source instantiation (open-source layout/OCR + open-source VLM) and compare against strong fine-tuned baselines to better contextualize gains. Third, the table-correction module incurs additional inference cost and latency because it invokes a large VLM for every table correction. This overhead can constrain throughput in high-volume settings. Finally, the correction module is sensitive to prompt design. Although the one-shot example generalizes well, unusual layout or heavy OCR noise can still degrade performance. Future work will investigate automated prompt-optimization strategies and iterative self-refinement loops that allow the model to detect and repair layout inconsistencies more effectively, reducing reliance on manual prompt tuning.

6. **Conclusion**

This work introduces a training-free pipeline for financial document understanding that integrates layout representation, one-shot multimodal table correction, and unified downstream extraction and reasoning. By converting document images into a Markdown–HTML format and repairing table layout using visually grounded prompting, the proposed system addresses a key limitation of existing OCR and VLM-based approaches: the inability to reliably preserve fine-grained layout information. Empirical evaluations across diverse datasets demonstrate consistent gains, 100% paystub extraction accuracy, 74.0% Finance-QA accuracy, and substantial improvements in PubTabNet TEDS, confirming that layout fidelity directly enhances both extraction and semantic reasoning. Overall, our results highlight the value of interpretable, compositionally pipelines for scalable, accurate, and enterprise-ready financial document processing.

References

[1] X. Zhong, J. Tang, and A. J. Yepes. *PubLayNet: largest dataset ever for document layout analysis*. 2019. arXiv: 1908.07836 [cs.CL]. URL: https://arxiv.org/abs/1908.07836.

[2] A. R. Katti, C. Reisswig, C. Guder, S. Brarda, S. Bickel, J. Höhne, and J. B. Faddoul. *Chargrid: Towards Understanding 2D Documents*. 2018. arXiv: 1809.08799 [cs.CL]. URL: https://arxiv.org/abs/1809.08799.

[3] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou. "LayoutLM: Pre-training of Text and Layout for Document Image Understanding". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '20. ACM, Aug. 2020, 1192–1200. DOI: 10.1145/3394486.3403172. URL: http://dx.doi.org/10.1145/3394486.3403172.

[4] G. Jaume, H. K. Ekenel, and J.-P. Thiran. *FUNSD: A Dataset for Form Understanding in Noisy Scanned Documents*. 2019. arXiv: 1905.13538 [cs.IR]. URL: https://arxiv.org/abs/1905.13538.

[5] M. Singal. *Personal Financial Dataset for India (Kaggle)*. https://www.kaggle.com/datasets/mehaksingal/personal-financial-dataset-for-india. Accessed 2025-11-10. 2021.

[6] Sujet AI. *Sujet-Finance-QA-Vision-100k (Hugging Face Dataset)*. https://huggingface.co/datasets/sujet-ai/Sujet-Finance-QA-Vision-100k. Accessed 2025-11-10. 2024.

[7] X. Zhong, E. ShafieiBavani, and A. Jimeno Yepes. "Image-based Table Recognition: Data, Model, and Evaluation". In: *European Conference on Computer Vision (ECCV)*. 2020. URL: https://arxiv.org/abs/1911.10683.

[8] R. Smith. "An Overview of the Tesseract OCR Engine". In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 2. 2007, pp. 629–633. DOI: 10.1109/ICDAR.2007.4376991.

[9] ABBYY. *ABBYY FineReader: AI-Powered OCR and PDF Software*. https://www.abbyy.com/finereader/. Accessed: 2025-01-01. 2024.

[10] Amazon Web Services. *Amazon Textract: Machine Learning-Based Document Text and Data Extraction*. https://aws.amazon.com/textract/. Accessed: 2025-01-01. 2024.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[12] Y. Xu, Y. Xu, T. Lv, L. Cui, F. Wei, G. Wang, Y. Lu, D. Florencio, C. Zhang, W. Che, M. Zhang, and L. Zhou. *LayoutLMv2: Multi-modal Pre-training for Visually-Rich Document Understanding*. 2022. arXiv: 2012.14740 [cs.CL]. URL: https://arxiv.org/abs/2012.14740.

[13] Y. Huang, T. Lv, L. Cui, Y. Lu, and F. Wei. "LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking". In: *Proceedings of ACM Multimedia*. 2022. DOI: 10.1145/3503161.3548112. URL: https://arxiv.org/abs/2204.08387.

[14] S. Appalaraju, B. Jasani, B. U. Kota, Y. Xie, and R. Manmatha. *DocFormer: End-to-End Transformer for Document Understanding*. 2021. arXiv: 2106.11539 [cs.CV]. URL: https://arxiv.org/abs/2106.11539.

[15] H. Liu, C. Li, Q. Wu, and Y. J. Lee. *Visual Instruction Tuning*. 2023. arXiv: 2304.08485 [cs.CV]. URL: https://arxiv.org/abs/2304.08485.

[16] OpenAI. *GPT-4o System Card*. https://openai.com/index/gpt-4o-system-card/. Accessed 2025-11-10. 2024.

[17] G. Kim, T. Hong, M. Yim, J. Nam, J. Park, J. Yim, W. Hwang, S. Yun, D. Han, and S. Park. "OCR-Free Document Understanding Transformer". In: *European Conference on Computer Vision (ECCV)*. 2022.

[18] K. Lee, M. Joshi, I. Turc, H. Hu, F. Liu, J. Eisenschlos, U. Khandelwal, P. Shaw, M.-W. Chang, and K. Toutanova. *Pix2Struct: Screenshot Parsing as Pretraining for Visual Language Understanding*. 2023. arXiv: 2210.03347 [cs.CL]. URL: https://arxiv.org/abs/2210.03347.

[19] M. S. Nacson, A. Aberdam, R. Ganz, E. B. Avraham, A. Golts, Y. Kittenplon, S. Mazor, and R. Litman. *DocVLM: Make Your VLM an Efficient Reader*. 2024. arXiv: 2412.08746 [cs.CV]. URL: https://arxiv.org/abs/2412.08746.

[20] A. Nassar, A. Marafioti, M. Omenetti, M. Lysak, N. Livathinos, C. Auer, L. Morin, R. T. de Lima, Y. Kim, A. S. Gurbuz, M. Dolfi, M. Farré, and P. W. J. Staar. *SmolDocling: An ultra-compact vision-language model for end-to-end multi-modal document conversion*. 2025. arXiv: 2503.11576 [cs.CV]. URL: https://arxiv.org/abs/2503.11576.

[21] S. Mandal, A. Talewar, P. Ahuja, and P. Juvatkar. *Nanonets-OCR-S: A model for transforming documents into structured markdown with intelligent content recognition and semantic tagging*. 2025.

[22] Microsoft Corporation. *Document Intelligence: Layout model*. https://learn.microsoft.com/en-us/azure/ai-services/document-intelligence/prebuilt/layout. Accessed 2025-11-10. 2025.

[23] Google Cloud. *Google Cloud Document AI*. https://cloud.google.com/document-ai. Accessed: 2025-11-20.

[24] Mistral AI. *Mistral OCR*. https://mistral.ai/news/mistral-ocr. Accessed 2025-11-10. 2025.

[25] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. "Learning Transferable Visual Models From Natural Language Supervision". In: *Proceedings of the 38th International Conference on Machine Learning (ICML)*. Vol. 139. PMLR. 2021, pp. 8748–8763. URL: https://arxiv.org/abs/2103.00020.

[26] Microsoft Corporation. *Document Intelligence: Read OCR model*. https://learn.microsoft.com/en-us/azure/ai-services/document-intelligence/prebuilt/read. Accessed 2025-11-10. 2025.

Appendix A. Example of A Table Correction Prompt

```
_ONESHOT_SYSTEM_PROMPT = """
You correct and regenerate clean, valid HTML tables inside a MARKDOWN document.

INPUTS:
(A) A DOCUMENT IMAGE (may contain one or multiple tables). This image is the ground truth for
      all table structures (layout, row/col counts, merged cells).
(B) A MARKDOWN FILE containing noisy HTML tables embedded within text. The noisy HTML tables
     may have OCR or structural errors.

YOUR TASK:
Reconstruct and replace each noisy HTML table in the Markdown file with a corrected HTML
     version that matches the IMAGE. The rest of the Markdown content (paragraphs, lists,
     captions, etc.) should remain unchanged.

PRIORITY & STRATEGY
- IMAGE is the ground truth for TABLE STRUCTURE.
- NOISY HTML is mainly for textual content.
- If IMAGE and NOISY HTML disagree on STRUCTURE -> follow IMAGE.
- If IMAGE and NOISY HTML disagree on TEXT -> prefer clearer/legible text from IMAGE; fall
     back to HTML text if uncertain.

OUTPUT - STRICT JSON (no markdown fences):
{
  "corrected_markdown": "### Document Title\n\nSome text...\n\n<table>...</table>\n\nMore
       text..."
}

RULES
1. Keep the Markdown layout intact - only replace each <table>...</table> segment.
2. Each corrected table must be a single self-contained HTML block: <table>...</table>.
3. Use valid HTML semantics: <thead>, <tbody>, <tr>, <th>, <td>. Use <th> for header cells.
4. Reconstruct merged headers/rows (rowspan/colspan) exactly as shown in the IMAGE.
5. Preserve all visible information (from IMAGE or HTML if not contradictory).
6. Fix clear OCR/text typos only - do not paraphrase or reformat text.
7. Preserve inline formatting (e.g., <sup>, <sub>, <b>, <i>).
8. Ensure table alignment: body rows align with header leaf columns.
9. Maintain original table order and placement in Markdown.
10. Do not add extra commentary, Markdown fences, or explanations - output pure JSON.

EXAMPLE:
If a Markdown file contains two tables, return one unified corrected_markdown string with
     both tables fixed in place.
"""
```