LAVA: Logic-Aware Validation and Augmentation Framework for Large-Scale Financial Document Auditing

Ruoqi Shu[*†], **Xuhui Wang**[*†], **Isaac Wang**[‡], **Yanming Mai, Bo Wan**

BMO Financial Group
ruoqi.shu@unisystechconsulting.com, xuhuieric.wang@bmo.com,
{isaac.wang, yanming.mai, bo.wan}@bmo.com

Abstract

Financial document validation in production—such as payroll auditing, tax compliance, and loan underwriting—demands exceptional accuracy, consistency, and reproducibility under strict enterprise constraints. In practice, documents arrive with heterogeneous layouts and formats, semantically rich, context-dependent content, and embedded business rules that current pipelines struggle to process reliably. We introduce LAVA (Logic-Aware Validation and Augmentation)—a modular, backbone-agnostic pipeline built on multimodal large language models—that integrates a four-stage design: document–rule retrieval, layout-preserving information extraction, auxiliary metadata enrichment, and auditable symbolic/arithmetic verification. LAVA supports robust rule grounding, fine-grained error attribution, and consistent, traceable end-to-end execution—capabilities essential for high-stakes deployment. Evaluated on a large real-world benchmark with diverse financial documents and dozens of expert-curated validation rules, LAVA outperforms baselines in hallucination control and edge-case handling while maintaining efficient token usage, demonstrating practicality for high-volume, time-critical validation.

1 Introduction

Regulatory penalties, financial losses, and reputational damage can all result from a single error in financial document validation, making accuracy, consistency, and auditability non-negotiable. Financial institutions process millions of documents daily across workflows such as loan underwriting, payroll auditing, tax compliance, and fraud detection. The challenge is acute for semi-structured documents like statements, invoices, and tax slips, which span multiple pages, exhibit irregular layouts, encode domain-specific business logic, and often arrive as noisy scans or non-standard PDFs (Bhattacharyya et al., 2025; Ding et al., 2024a; Xu et al., 2020; Chen et al., 2024).

Recent years have witnessed rapid progress in visually rich document understanding through layout- and structure-aware pretraining of multimodal large language models (MLLMs). LayoutLM (Xu et al., 2020) pioneered spatial–textual joint encoding, followed by DocFormer (Appalaraju et al., 2021), DocLLM (Wang et al., 2024), mPLUG-DocOwl2 (Hu et al., 2025), and ROP (Zhang et al., 2024), advancing multimodal architectures for structural representation. This shift moves beyond pure text modelling to multimodal document intelligence. However, evaluations remain dominated by perceptual and question answering (QA) tasks, probing reasoning in a narrow, task-specific manner while rarely accessing validation—reasoning that tests both interpretation and reliability under structured and cross-field constraints.

Enterprise-grade document validation presents requirements beyond those of perception or reasoning alone. It calls for symbolic rule enforcement, cross-field consistency, and multi-step logical coherence—capabilities only partially reflected in existing benchmarks (Wang et al., 2023b; Li et al., 2025; Borchmann et al., 2021). Recent datasets have expanded evaluation to layout-aware perception and understanding (Zhu et al., 2024; Wu et al., 2023; Mathew et al., 2021; Stanisławek et al., 2021; Šimsa et al., 2023), but compliance-critical validation logic remains underexplored. In practice, enterprises often patch the gap by pairing general-purpose models with rigid rule-based modules (Shende et al., 2024). Yet even state-of-the-art models that excel at extraction or understanding exhibit ongoing limitations when directly applied to validation: hallucinations remain common, and reasoning traceability is limited. Adaptation across regulatory schemas is fragile, and computational costs escalate with token usage at enterprise scale.

[*]Equal contribution.
[†]Corresponding author.
[‡]Work done during an internship at BMO Financial Group.

Proceedings of The 10th Workshop on Financial Technology and Natural Language. EMNLP-2025, Suzhou, China

These shortcomings make validation not merely an extension but a distinct and emerging frontier of document intelligence—one that demands frameworks where accuracy, efficiency, audit readiness, and robustness are central requirements.

Our Work. To address these challenges, we present **LAVA** (**L**ogic-**A**ware **V**alidation and **A**ugmentation), a modular and efficient framework for verifiable reasoning over semi-structured, layout-complex financial documents, designed for real-world applicability, fine-grained error attribution, and rapid adaptation and reproducibility across structurally similar collections. Agnostic to backbone models, LAVA extends beyond static benchmarks by integrating (i) layout-informed knowledge extraction preserving structural cues, (ii) domain-aware augmentation with contextual metadata, and (iii) arithmetic and symbolic verification ensuring factual alignment with business rules, orchestrated in a four-stage system of retrieval, extraction, augmentation, and hybrid reasoning.

We evaluate LAVA on a real-world large-scale industrial benchmark with validation rules curated by senior industry experts reflecting real regulatory conditions. Results show competitive gains in factual accuracy and symbolic correctness, together with lower computational overhead than baseline MLLM pipelines, demonstrating robustness and cost-effectiveness in realistic financial validation scenarios.

Our main contributions are:

- **Task and System.** We formalize *financial document validation* as a multi-document reasoning task—largely absent in existing benchmarks—and instantiate it in **LAVA**, a novel modular framework designed for accurate and auditable validation of semi-structured financial documents.
- **Reasoning Strategy & Auditability.** We design a controllable hybrid reasoning framework unifying factual/contextual templates with symbolic and arithmetic tasks via explicit formula generation, with an external checker fallback ensuring correctness, thereby enhancing accuracy, interpretability, and operational robustness.
- **Evaluation.** We propose a comprehensive evaluation framework covering symbolic correctness, factual alignment, and hallucination control for fine-grained reasoning assessment in realistic validation workflows.

2 Related Work

Visually Rich Document Understanding. Recent work has shifted from extraction pipelines toward LLM-centric modeling that integrates layout and visual cues. DocLayLLM (Liao et al., 2025) adds visual patches and 2D positional tokens, VisDoM (Suri et al., 2025) combines multimodal retrieval with consistency constraints, 3MVRD (Ding et al., 2024b) aligns fine- and coarse-grained signals via multi-task distillation, and LayoutLLM (Fujitake, 2024) applies instruction tuning for unified document tasks. These advances improve representation and generalization; our work is complementary, focusing on *auditable validation*—explicit symbolic checks, cross-field consistency, and reproducible reasoning traces required in compliance-critical workflows. Our framework is model-agnostic, plugging into any MLLM backbone to introduce validation as a controllable layer.

Layout-Guided Document Encoding. Backbones such as LayoutLMv3 (Huang et al., 2022), FormNet (Lee et al., 2022, 2023), and DocFormer (Appalaraju et al., 2021) combine textual, spatial, and visual cues through large-scale pretraining. These excel at form-style entity extraction but remain embedding-level and not optimized for symbolic validation. In contrast, our framework leverages layout-derived structures to enable modular business rule checks and reproducible logic tracing, addressing auditability without massive training effort.

LLM Verification and Rule-based Validation. Progress in LLM factuality—via self-checking (Dhuliawala et al., 2024), retrieval-augmented prompting (Qin et al., 2025), symbolic grounding (Hennigen et al., 2024), and rectification (Kang et al., 2024)—has improved reliability in clean text and formal reasoning (Liu et al., 2025). Yet these methods falter on noisy, heterogeneous documents with long-range dependencies and embedded business rules. Traditional rule engines (e.g., Drools) provide transparency but fail under layout variation, while hybrid LLM–rule or knowledge graph systems (Vertsel and Rumiantsau, 2024; Sadowski and Chudziak, 2025) trade flexibility for interpretability. Our framework instead couples symbolic verification with layout-

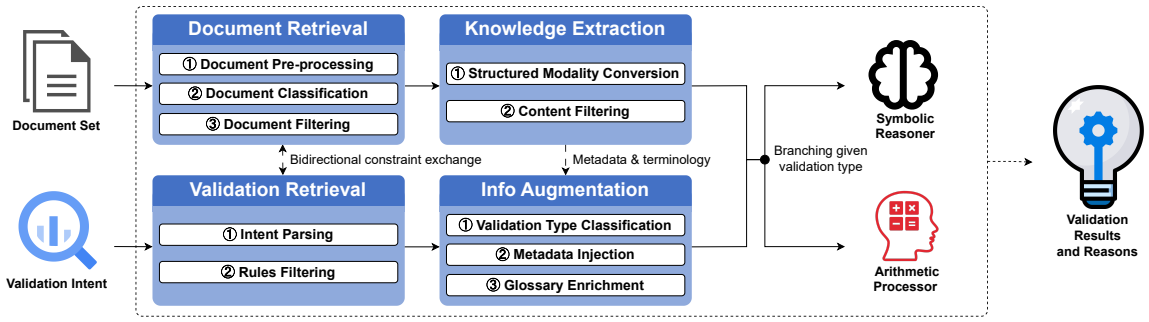


Figure 1: Overview of LAVA. The architecture comprises two parallel pipelines—document processing and rule grounding—interacting via bidirectional constraints (solid arrows: main data flow; dashed arrows: auxiliary exchange). The document-processing track performs retrieval, extraction, and augmentation to produce layout-preserving structured content with enriched metadata. The rule-grounding track retrieves, classifies, and dispatches applicable validation rules to either the Symbolic Reasoner or Arithmetic Processor, depending on reasoning type.

and metadata-informed prompting, merging rule transparency with neural adaptability, making it fit for compliance-critical validation.

3 Method

Problem Formalization. We define **financial document validation** as a human-in-the-loop copilot task, where the system assists users (e.g., underwriters, compliance officers, fraud analysts) in verifying whether a set of financial documents (e.g., from a mortgage application) satisfies certain business rules under real-world conditions of layout complexity, domain-specific logic, and noisy input.

Inputs. The system takes as input: (1) a document set $\mathcal{D} = \{d_1, \ldots, d_N\}$ in scanned PDF or image format, where each d_i is a semi-structured financial document (e.g., bank statement, tax form), and (2) a validation intent q, a user-specified verification goal in natural language (e.g., Does gross income exceed loan threshold?).

Objective. The system maps $(q, \mathcal{D})$ to a set of validation outputs $\mathcal{V} = \{v_k\}$, where each v_k includes: a subset of supporting documents $\mathcal{D}_k \subseteq \mathcal{D}$, a set of retrieved business rules $\mathcal{R}_k \subseteq \mathcal{R}$ from a **predefined** rule library, a binary verification label $y_k \in \{\texttt{Pass}, \texttt{Fail}\}$, and an explanation trace e_k for auditability. This formulation supports multi-document, logic-grounded reasoning while ensuring modular, interpretable validation aligned with enterprise workflows.

Architecture. Figure 1 illustrate LAVA architecture, which employs a modular design paradigm to address the complexity of enterprise-grade production systems. Although this design involves the integration of multiple components, it yields critical advantages for deployment. Decoupling the workflow isolates potential points of failure within individual modules, enabling targeted and independent validation and thereby systematically reducing the long-term verification cost and overall operational overhead. Furthermore, the ability to debug, update, or replace modules without systemic disruption enhances maintainability—a stark contrast to the challenges of managing opaque, end-to-end models. This design philosophy is therefore foundational to building a robust, auditable, and scalable system fit for the rigors of real-world financial validation.

3.1 Document and Validation Retrieval

We jointly describe the first two modules, as they operate in a tightly coupled fashion to determine relevant document-rule pairs for downstream extraction. Given a user-specified *validation intent* q and a document set $\mathcal{D} = \{d_1, \ldots, d_N\}$, the modules select a subset $\mathcal{D}_v \subseteq \mathcal{D}$ that is temporally valid and relevant to the task, along with a set of executable business rules $\mathcal{R}_v = \{r_1, \ldots, r_M\}$. Both subsets are tailored to the verification goal through a bidirectional constraint mechanism, ensuring only applicable document-rule pairs are forwarded for knowledge extraction and augmentation.

Document Retrieval. Documents are first preprocessed to normalize layout and correct visual artifacts (e.g., OCR errors, rotation, skew) (Boudraa et al., 2020), preserving alignment and structural fidelity for downstream modules. Each document d_i is then classified into a predefined document type using a lightweight image-based classifier such as TinyViT (Wu et al., 2022), as financial documents of the same type generally share consistent page-

level features. Recognized types are forwarded to Validation Retrieval to constrain rule applicability.

Document-level metadata (e.g., date, coverage period) is extracted using a template-guided NER pipeline with regex patterns and rule-based heuristics. Temporal constraints parsed from the validation intent in the next module (e.g., "past 3 months") are applied to filter out documents outside the relevant time window. In addition, applicable document types extracted from retrieved rules in Validation Retrieval are passed back to further prune documents irrelevant to all candidate rules.

Validation Retrieval. Given q, this module retrieves a subset of rules $\mathcal{R}_v$ from the predefined library $\mathcal{R}$, based on semantic relevance and document compatibility. The rule library is enriched with metadata specifying applicable document types. Lightweight LLMs can parse q to extract temporal constraints, while a sentence encoder (e.g., Sentence-BERT (Reimers and Gurevych, 2019)) encodes q to retrieve top-K semantically relevant rules. Retrieved rules are then filtered using document-type constraints from Document Retrieval, and their own document-type metadata is fed back to further refine $\mathcal{D}_v$.

In conclusion, temporal and semantic cues from q filter the document set, while recognized document types and rule metadata eliminate inapplicable rules. This closed-loop filtering minimizes irrelevant candidates on both sides, reduces reasoning load, and improves accuracy without sacrificing interpretability, ensuring downstream processing operates on the most relevant and valid document-rule pairs.

3.2 Knowledge Extraction

This module transforms each filtered document $d_i \in \mathcal{D}_v$ into a compact, layout-aware hybrid representation for downstream reasoning. Instead of flat key–value pairs, we produce a structured markup that encodes page layout, visual grouping, and field dependencies, serving as a bridge between scanned formats and language-model-friendly input.

Structured Modality Conversion. To capture the rich visual and semi-structural semantics of financial documents, we adopt an HTML-like markup constructed from parsed layout and OCR signals. Prior work shows that retaining tabular alignments, hierarchical sections, and field groupings improves reasoning fidelity (Sui et al., 2024), but raw markup is insufficient for noisy scans. We therefore augment it with: (1) structural parsing via document analysis tools (e.g., LayoutParser (Shen et al., 2021), LayoutLMv3 (Huang et al., 2022)), AWS Textract; (2) OCR-based recovery (e.g., Tesseract OCR (Smith, 2007)) for free-form or scattered content, including reconstruction of long paragraphs into coherent spans; (3) proximity-based grouping to merge fragmented tokens into coherent semantic units; and (4) visual region preservation for inherently non-textual content (e.g., charts, stamps, signatures), where candidate regions are identified from layout cues (e.g., low text coverage or OCR confidence) and retained as image patches for the MLLM input.

Content Filtering. The extracted representation often contains much noise from headers, footers, boilerplate blocks, or placeholders. We prune such elements using DOM structure, field labels (e.g., `Name`, `Address`), and positional cues, reducing token usage while improving attention focus for model prompts.

By combining structure-preserving markup, semantic recovery, selective visual preservation, and noise reduction, this module delivers a high-fidelity hybrid form that maintains interpretability while enabling reliable downstream reasoning.

3.3 Information Augmentation

This module enriches downstream reasoning by injecting auxiliary signals from both documents and retrieved rules. It serves two purposes: routing rules to the appropriate verification pathway and augmenting prompts with rich metadata to improve model understanding.

Each rule $r_k \in \mathcal{R}_v$ is classified as symbolic (context-dependent logic) or arithmetic (numeric computation) via a lightweight LLM query, avoiding brittle heuristics. In parallel, metadata is additionally extracted from layout-preserving outputs: language (via token-based detection), document types (from retrieval module), and domain-specific terms (e.g., withholding or CPP [1]) detected through lexical and structural heuristics targeting low-frequency tokens, abbreviations, and left-hand-side predicates in field labels or rule expressions. This metadata is distilled into concise semantic clarifications for interpretability and disambiguation. The resulting signals are incorporated into prompt headers or reference blocks, sharpening alignment

[1] CPP refers to the Canada Pension Plan, a mandatory pension contribution in Canadian payroll systems.

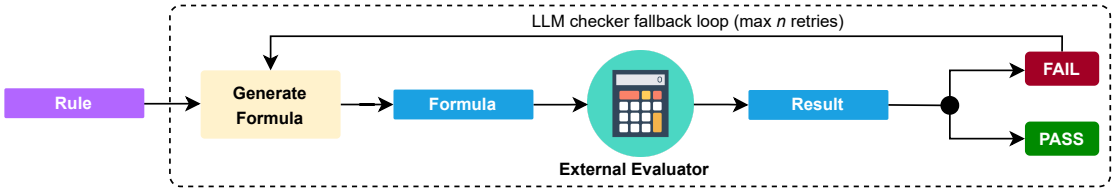


Figure 2: Arithmetic Processor. Given a validation rule, the system generates a formula and delegates evaluation to an external calculator. A checker LLM validates alignment between the rule and result. Failures trigger a fallback loop with negative feedback.

with rule semantics and improving precision in complex validation scenarios, while keeping the verification loop efficient.

3.4 Validation

To support diverse verification needs and enhance the reliability, we adopt a bifurcated framework with two sub-modules: **Arithmetic Processor** for numerical tasks (e.g., verifying tax deductions, calculating gross revenue), and **Symbolic Reasoner** for all other general rules requiring semantic and contextual reasoning.

Arithmetic Processor. As illustrated in Figure 2, when rules involving arithmetic or numerical computation are routed to this sub-module, instead of relying on LLMs for direct computation—prone to hallucinations and numeric instability, we adopt a tool-use paradigm where the model is used solely for generating a task-specific formula, which is then executed by a deterministic external engine such as a Python interpreter (Gao et al., 2023; Chen et al., 2022). To ensure alignment between the formula and the validation semantics, we introduce a fallback auditing loop: the rule and generated formula are reviewed by a secondary LLM "checker". If a mismatch is detected, the formula is regenerated, conditioned on the previous (incorrect) version as a negative example. This loop improves robustness by systematically detecting and correcting errors, mitigating hallucinated computations and flawed reasoning pathways.

Symbolic Reasoner. This sub-module handles semantic or structural reasoning. In contrast to Arithmetic Processor, it directly delegates rules to a general-purpose model, as such reasoning falls within the model's inherent strengths. This design choice reduces integration overhead and inference latency, while remaining sufficient for a wide range of non-arithmetic verification scenarios.

Prompt Excerpt for Symbolic Reasoner

Excerpt: only key parts are shown below, simplified for illustration.

You are an assistant verifying `{document_types}` documents (in `{languages}`), with extracted `{knowledge_names}` ...

Instructions:

1. Use `{structured_representation}` as references for identifying the layout...
2. Use `{semantic_fields}` as references for explicit context...

...

n. Output as follows:
   - Only output the fields requested in the validation rule...
   - Format: empty = ""; date = YYYYMMDD...

   Example: `{example_output}`

Task Input:
`{validation_rule}`
- `{term_1}: {definition_1}, ..., {term_n}: {definition_n}`

Output: Return JSON only. No explanation or preamble...

Figure 3: Prompt excerpt for Symbolic Reasoner, dynamically built from the retrieved validation rule, extracted knowledge, and augmentation components. Full versions for both Arithmetic Processor and Symbolic Reasoner are provided in Appendix A.1.1 and A.1.2.

Prompts. We adopt our prompt construction strategy in meta-prompting fashion (Zhang et al., 2023) that presents task-relevant information in a step-wise and zero-shot format rather than relying on illustrative examples. Unlike chain-of-thought (Wei et al., 2022), few-shot prompting (Brown et al., 2020), or self-consistency (Wang et al., 2023a), our approach avoids brittle curated examples, which struggle to generalize across heterogeneous financial documents (Zhou et al., 2023), and removes high inference cost of example-heavy prompting.

To ensure verification prompts are precisely tailored to both the rules and associated documents, we implement a dynamic, template-based prompt construction system. It integrates supplemental information from upstream modules with the rule to populate flexible prompt templates. By leveraging templating libraries like Jinja, we employ programmatic, code-like logic to govern the inclusion and granularity of metadata based on both document layout and task semantics. As illustrated

in Figure 3, this mechanism produces prompts that are maximally informative while remaining token-efficient and robust to both arithmetic and symbolic reasoning tasks. By structuring all inputs according to rule logic, metadata, and knowledge, our approach generalizes well across heterogeneous financial documents. The resulting system remains accurate, cost-efficient and context-aware, supporting scalable and reliable verification tasks.

4 Experiments

We evaluate LAVA on real-world Canadian mortgage application documents sampled from a proprietary database to assess the system within a consistent validation scenario. The set includes multiple document types and around 1,000 scanned PDFs or images—such as tax forms, bank/investment statements, and legal agreements—selected for their semi-structured formats, diverse layouts, and rich logical dependencies (see A.4). This focused yet heterogeneous domain enables rigorous testing of LAVA's ability to produce accurate outputs and avoid false positives, particularly where heuristic, zero-shot, and few-shot LLM-based methods struggle with layout variability or reasoning complexity.

To ensure rigorous and interpretable evaluation, we adopt rule-level testing rather than intent-level, with validation rules drawn from a curated library provided by business stakeholders. Each rule corresponds to an atomic validation unit and is applied only to the document types for which it is defined. For fairness and comparability, document type metadata is included with the dataset, since accurate validation cannot be assessed without first matching each document to its correct rule set; this ensures that all pipelines are evaluated under the same conditions, without document and validation retrieval. Ground-truth outcomes are manually annotated by domain experts, and automatic evaluation is complemented by manual audits from business collaborators, ensuring both accuracy and institutional credibility in line with production-grade expectations for document validation.

4.1 Implementations

To ensure fairness, all baselines, LAVA, and LAVA's ablation variants use Claude 3.7 Sonnet (Anthropic, 2024) as the validation component. We configure the model with a maximum response length of 5120 tokens and enable reasoning (thinking) with a budget of 1024 tokens, allowing the model to better understand complex validation rules and generalize across documents with diverse layouts and edge scenarios. For LAVA's Arithmetic Processor, the maximum retry number n in the fallback loop is set to 2, to ensure a practical trade-off between error correction and computational efficiency. And all other LLM parameters are kept identical across evaluations.

For document analysis in experiments, we prioritized a methodology that ensures our findings are portable, reproducible, and immediately accessible. To this end, our experiment exclusively utilizes off-the-shelf tools that do not require custom training or dataset-specific fine-tuning. A foundational layer of raw text and spatial information was established for the experiment using the open-source Tesseract OCR (Smith, 2007). For the processing of structural semantics, such as tables and forms, the publicly available AWS Textract service was employed. This consistent and transparent tooling strategy not only ensures that the comparative evaluation in Sections 4.4 and 4.5 fairly assesses the core architectural and reasoning capabilities of the different pipelines, but also directly supports other researchers to easily replicate and build upon our results.

4.2 Validation Rules

We group several dozen validation rules into five categories, reflecting distinct reasoning and computation demands. These categories assess the pipeline's ability to extract information, integrate rules, and perform contextual reasoning. See A.2 for complete versions of some representative rules.

4.3 Evaluation Metrics

Our evaluation metrics comprehensively assess reasoning quality across all pipeline stages. Given the distinct characteristics of our compliance-sensitive validation task compared to common document understanding settings, we design task-specific metrics with particular emphasis on suppressing false positives—a critical industrial concern causing costly investigations, delays, and loss of trust. We evaluate baselines from three perspectives to cover these aspects. Their formal definitions are given in Appendix A.3, along with visual examples showing how each metric manifests in an illustrative sample document.

Hallucination Control. We report the percentage of responses with two failure types:

Metric	Rule Group	VLM + Field-Level OCR	LLM + Field-Level OCR	LLM + Enhanced OCR	LAVA
Factual Hallucination Rate	C_1	0.03	0.03	0.01	**0.01**
	C_2	0.08	0.10	0.04	**0.03**
	C_3	0.31	0.33	0.30	**0.18**
	C_4	0.05	0.08	0.05	**0.03**
	C_5	0.28	0.30	0.15	**0.05**
Numerical Infidelity Rate	C_1	N/A	N/A	N/A	**N/A**
	C_2	0.08	0.04	0.03	**0.01**
	C_3	0.33	0.31	0.20	**0.10**
	C_4	0.11	0.08	0.05	**0.00**
	C_5	0.25	0.10	0.10	**0.00**
Edge Case Error Rate	C_1	N/A	N/A	N/A	**N/A**
	C_2	0.27	0.25	0.23	**0.02**
	C_3	0.86	0.90	0.82	**0.17**
	C_4	0.46	0.34	0.43	**0.11**
	C_5	0.89	0.92	0.86	**0.08**

Table 1: Performance of baselines across three metrics and five validation rule categories (C_1–C_5). C_1: content extraction; C_2: conditional logic reasoning; C_3: multi-step logic reasoning; C_4: unconstrained arithmetic consistency checking; C_5: constrained arithmetic consistency checking.

- **Factual Hallucination Rate:** Content not grounded in the source document, such as fabricated values or formulas, and unsupported claims.
- **Numerical Infidelity Rate:** Incorrect quantitative reasoning or numerical derivations, such as incorrect formula execution results and conclusions inconsistent with the given numerical evidence.

Edge Case Handling. This metric measures failure rate on complex or exception-driven scenarios, about 10%–25% of document–rule pairs. Such cases require *adaptive logic reasoning* beyond fixed rules and broader *logic coverage* for diverse edge conditions. As these challenges often overlap, we report a single rate to capture both, where higher values indicate weaker generalization and reasoning flexibility.

Token Cost. We measure input and output tokens per rule check to assess computational efficiency and deployment cost, highlighting trade-offs between reasoning quality and efficiency across pipelines. For each document-rule pair, token counts are recorded and aggregated over the full evaluation set.

4.4 Comparative Baselines

Here, *Field-Level OCR* denotes OCR output containing only individual field texts and their spatial information (e.g., from bounding boxes), without higher-level structure such as tables, sections, or relational links between fields. This representation preserves raw layout signals but omits the structured markup used in LAVA's pipeline.

We consider three representative settings:

- **VLM + Field-Level OCR:** Document images with Field-Level OCR in a VQA setup, where OCR text aids visual grounding.
- **LLM + Field-Level OCR:** Same OCR content without images, measuring performance from textual–spatial data alone in a QA setup.
- **LLM + Enhanced OCR (no structural markup):** OCR refined with domain-specific cleanup and recovery steps identical to LAVA's extraction, but without structured markup, isolating LAVA's impact of structured representation and modular reasoning.

Since OCR and markup information might introduce noise (e.g., when text is misrecognized), we also examine an OCR-free configuration to assess the impact of textual knowledge. This setting is evaluated in ablation study (Section 4.5), where all other modules are preserved but the model operates only on document images. It is excluded from the baseline set because the baselines are intended to preserve textual content in some form for fair comparison, whereas this variant probes the limits of visual-only reasoning and represents a more extreme ablation of the pipeline.

As shown in Table 1, LAVA achieves the lowest failure rates across all metrics and categories, with

especially large gains in multi-step logic reasoning (C_3) and constrained arithmetic consistency checking (C_5), reducing hallucination and numerical errors by over 10% compared to the best baseline. And it records approximately 0% Numerical Infidelity Rate in three of four categories. LAVA also substantially lowers edge case errors, indicating strong generalization under layout ambiguity and rare conditions. Common VQA and QA settings represented by the baselines underperform in this compliance-oriented validation task due to their limited handling of layout variability and domain-specific rules. These results highlight the benefit of structured prompts and modular reasoning in suppressing unsupported content generation and enabling reliable multi-step, context-aware reasoning over structural data.

We also sampled a subset of DocVQA (Mathew et al., 2021) documents with five manually defined rules each, where LAVA also achieved strong performance, confirming that our framework readily transfers to public data. More importantly, real-world validation tasks—where both documents and rules are richer in semantics, context, and reasoning complexity—pose substantially greater challenges, where our approach proves particularly effective under industry-level conditions.

4.5 Ablation Study

To assess the contribution of each core module in LAVA, we performed an ablation study using a unified metric: the percentage of responses that fail to exactly match the ground truth. This stricter measure was chosen because LAVA's earlier evaluations achieved low error rates in category-specific metrics, so a single comprehensive indicator better reveals performance drops when modules are altered or removed.

We evaluated six configurations: the full LAVA system; three **Knowledge Extraction (KE)** variants; and versions without **Info Augmentation (IA)** or **Arithmetic Processor (AP)**. The KE variants progressively reduce structural detail and change input format:

1. **Markdown KE** replaces LAVA's structured markup with Markdown-like fashion.

2. **Plain-text KE** flattens layout into text aligned to mimic visual spacing but without structural tags.

3. **No KE** omits textual knowledge entirely, prompting the model with original document images. This configuration serves as the OCR-free, vision-only benchmark we referred to in the baseline comparison (Section 4.4).

Pipeline	C_1	C_2	C_3	C_4	C_5
LAVA w/o KE	0.03	0.65	0.67	0.09	0.66
LAVA w/ Plain-text KE	0.02	0.63	0.58	0.08	0.48
LAVA w/ Markdown KE	0.01	0.25	0.54	0.05	0.55
LAVA w/o IA	0.01	0.15	0.45	0.05	0.12
LAVA w/o AP	0.01	0.03	0.29	0.10	0.56
LAVA	**0.01**	**0.03**	**0.28**	**0.05**	**0.07**

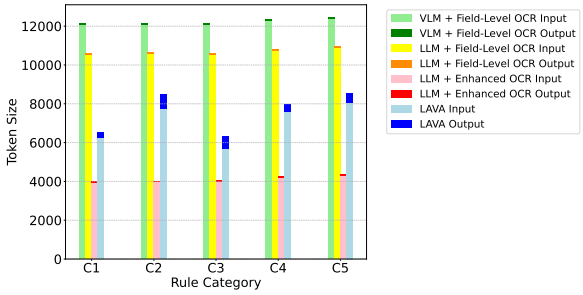
Table 2: Percentage of responses failing to exactly match ground-truth answers across five rule categories.

This stepwise removal strips explicit layout and relational cues, forcing reliance on surface text or raw images. For AP ablation, AP is replaced by Symbolic Reasoner, removing explicit calculation/result validation. The two Retrieval modules are not ablated since all experiments use a fixed rule set, making them independent of validation accuracy.

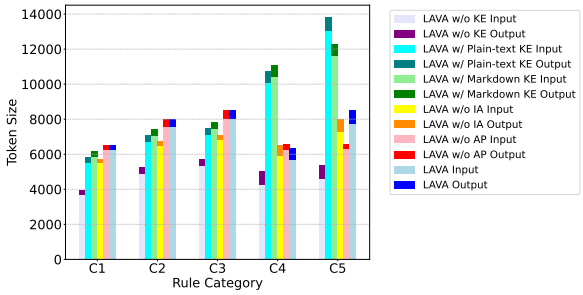
The error patterns in table 2 shows that KE, IA, and AP play complementary roles, with KE being the most critical. Removing KE forces the model to rely almost entirely on visual reasoning, raising C_2, C_3, and C_5 failure rate to 0.65–0.67—over twice LAVA's error in multi-step logic (C_3) and nearly tenfold in constrained arithmetic consistency checking (C_5) even with the validation module's assistance. Two factors explain this: (1) Structural conversion: without KE's structural format that preserves complex and nested layout, grouping, and field dependencies, multi-step logic and table reasoning lose explicit relational cues. Downgrading to Markdown or plain text progressively strips away these signals, with Markdown underperforming plain text in C_5 due to weaker preservation of columnar alignment in tables; (2) Content filtering: without KE's filtering stage, noisy headers, footers, and irrelevant fields dilute attention and amplify hallucination risk. This gradient from fully structured to none shows that noise suppression, explicit spatial and relational cues are essential for stable reasoning, especially in multi-field and arithmetic-heavy rules for financial document validation tasks.

IA mainly impacts logical reasoning: removing it raises C_3 error from 0.28 to 0.45, suggesting that metadata and domain cues help models resolve field semantics and linking conditions across steps.

AP's effect is computation-focused: removing it leaves C_1–C_3 unchanged but drives C_4/C_5 error



(a) LAVA and baselines.



(b) LAVA and ablated variants.

Figure 4: Average input and output token counts across five rule categories.

to 0.10/0.56, reflecting the limits of unconstrained LLM reasoning in arithmetic tasks and the benefit of explicit calculation with fallback mechanism.

Overall, the degradations are consistent with the design intent: KE supplies the structured substrate, IA enriches it with semantic context, and AP enforces numerical fidelity, justifying LAVA's functionally complementary and modular composition.

4.6 Business Impact

While Section 4.4 and 4.5 highlight LAVA's technical advantages in accuracy and robustness, real-world adoption also depends on its operational efficiency and cost-effectiveness. By consolidating token usage, we can further assess how LAVA scales under realistic operational constraints, using token consumption as a proxy for latency and API expenditure in high-volume production workflows.

LAVA amortizes multi-step costs by extracting structured knowledge once and reusing it across rules. As shown in Figure 4a, this design cuts input tokens by 25%–45% versus VLM and LLM baselines with Field-Level OCR, lowering inference costs and enabling faster turnaround for time-critical financial validation processes, while also achieving fewer errors (Table 1).

Ablation results (Figure 4b) show that in non-computation tasks (C_1–C_3), excluding variant w/o KE, LAVA adds minimal tokens with stable output length due to the predefined schema-based response format. For computation tasks (C_4–C_5), the less faithful structural representation from plain-text and Markdown KE inflate inputs by causing more model output errors and retries in the validation loop. Across all ablations, full LAVA adds under 3k tokens per rule ($\sim$ \$0.009 with Claude 3.7 or $\sim$ \$0.006 with GPT-4.1), an overhead outweighed by substantial gains in accuracy and reasoning stability. This demonstrates that LAVA delivers a dual advantage: (1) immediate computational savings from reduced token usage, and (2) long-term operational efficiency from its modular, auditable design. This combination suggests that LAVA's architectural benefits outweigh the modular integration complexity, making it a cost-effective and scalable choice for high-volume, time-critical real-world financial workflows.

5 Conclusion

We presented LAVA, a modular, interpretable and backbone- and domain-agnostic system for enterprise-grade financial document validation. Combining layout-preserving extraction, domain-aware augmentation, and symbolic and arithmetic verification, LAVA achieves high factual alignment and numerical reliability on a large real-world benchmark, with lower computational overhead than monolithic prompting. Evaluation with factual hallucination rate, numerical infidelity rate, and edge case handling shows that structured, multi-stage reasoning enables fine-grained error attribution, stronger robustness in compliance-sensitive workflows, and transfer to structurally similar data. Looking ahead, we aim to handle noisier, less-structured documents and to learn rule generalization across formats and domains.

6 Ethical Considerations

This research was conducted using a proprietary dataset of financial documents, with all data fully anonymized and handled under strict institutional privacy protocols. We acknowledge the potential risk of algorithmic bias in financial decision-making. LAVA's design as an accurate, scalable and auditable system is a deliberate step to mitigate this risk by enhancing transparency and ensuring human accountability. Nevertheless, any production deployment would necessitate rigorous, ongoing audits for demographic bias to ensure fair and equitable outcomes.

References

Anthropic. 2024. Claude 3 model family. https://www.anthropic.com/news/claude-3-family. Accessed: 2025-08-07.

Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha. 2021. Docformer: End-to-end transformer for document understanding. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 973–983. IEEE.

Aniket Bhattacharyya, Anurag Tripathi, Ujjal Das, Archan Karmakar, Amit Pathak, and Maneesh Gupta. 2025. Information extraction from visually rich documents using LLM-based organization of documents into independent textual segments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17241–17256, Vienna, Austria. Association for Computational Linguistics.

Łukasz Borchmann, Michał Pietruszka, Tomasz Stanislawek, Dawid Jurkiewicz, Michał Turski, Karolina Szyndler, and Filip Graliński. 2021. Due: End-to-end document understanding benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Omar Boudraa, Walid Khaled Hidouci, and Dominique Michelucci. 2020. Using skeleton and hough transform variant to correct skew in historical documents. *Mathematics and computers in simulation*, 167:389–403.

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. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks.

Yufan Chen, Jiaming Zhang, Kunyu Peng, Junwei Zheng, Ruiping Liu, Philip Torr, and Rainer Stiefelhagen. 2024. Rodla: Benchmarking the robustness of document layout analysis models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 15556–15566. IEEE.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578, Bangkok, Thailand. Association for Computational Linguistics.

Yihao Ding, Jean Lee, and Soyeon Caren Han. 2024a. Deep learning based visually rich document content understanding: A survey. *ArXiv preprint*, abs/2408.01287.

Yihao Ding, Lorenzo Vaiani, Caren Han, Jean Lee, Paolo Garza, Josiah Poon, and Luca Cagliero. 2024b. 3MVRD: Multimodal multi-task multi-teacher visually-rich form document understanding. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15233–15244, Bangkok, Thailand. Association for Computational Linguistics.

Masato Fujitake. 2024. LayoutLLM: Large language model instruction tuning for visually rich document understanding. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10219–10224, Torino, Italia. ELRA and ICCL.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: program-aided language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

Lucas Torroba Hennigen, Shannon Shen, Aniruddha Nrusimha, Bernhard Gapp, David Sontag, and Yoon Kim. 2024. Towards verifiable text generation with symbolic references. *Preprint*, arXiv:2311.09188.

Anwen Hu, Haiyang Xu, Liang Zhang, Jiabo Ye, Ming Yan, Ji Zhang, Qin Jin, Fei Huang, and Jingren Zhou. 2025. mPLUG-DocOwl2: High-resolution compressing for OCR-free multi-page document understanding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5817–5834, Vienna, Austria. Association for Computational Linguistics.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 4083–4091, New York, NY, USA. Association for Computing Machinery.

Haoqiang Kang, Juntong Ni, and Huaxiu Yao. 2024. Ever: Mitigating hallucination in large language models through real-time verification and rectification. *Preprint*, arXiv:2311.09114.

Chen-Yu Lee, Chun-Liang Li, Timothy Dozat, Vincent Perot, Guolong Su, Nan Hua, Joshua Ainslie, Renshen Wang, Yasuhisa Fujii, and Tomas Pfister. 2022. FormNet: Structural encoding beyond sequential modeling in form document information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume*

1: Long Papers), pages 3735–3754, Dublin, Ireland. Association for Computational Linguistics.

Chen-Yu Lee, Chun-Liang Li, Hao Zhang, Timothy Dozat, Vincent Perot, Guolong Su, Xiang Zhang, Kihyuk Sohn, Nikolay Glushnev, Renshen Wang, Joshua Ainslie, Shangbang Long, Siyang Qin, Yasuhisa Fujii, Nan Hua, and Tomas Pfister. 2023. FormNetV2: Multimodal graph contrastive learning for form document information extraction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9011–9026, Toronto, Canada. Association for Computational Linguistics.

Siqi Li, Yufan Shen, Xiangnan Chen, Jiayi Chen, Hengwei Ju, Haodong Duan, Song Mao, Hongbin Zhou, Bo Zhang, Bin Fu, Pinlong Cai, Licheng Wen, Botian Shi, Yong Liu, Xinyu Cai, and Yu Qiao. 2025. Gdi-bench: A benchmark for general document intelligence with vision and reasoning decoupling.

Wenhui Liao, Jiapeng Wang, Hongliang Li, Chengyu Wang, Jun Huang, and Lianwen Jin. 2025. Doclayllm: An efficient multi-modal extension of large language models for text-rich document understanding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 4038–4049.

Chengwu Liu, Ye Yuan, Yichun Yin, Yan Xu, Xin Xu, Zaoyu Chen, Yasheng Wang, Lifeng Shang, Qun Liu, and Ming Zhang. 2025. Safe: Enhancing mathematical reasoning in large language models via retrospective step-aware formal verification. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12171–12186, Vienna, Austria. Association for Computational Linguistics.

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.

Yuehan Qin, Shawn Li, Yi Nian, Xinyan Velocity Yu, Yue Zhao, and Xuezhe Ma. 2025. Don't let it hallucinate: Premise verification via retrieval-augmented logical reasoning.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Albert Sadowski and Jarosław A. Chudziak. 2025. Explainable rule application via structured prompting: A neural-symbolic approach. *Preprint*, arXiv:2506.16335.

Zejiang Shen, Ruochen Zhang, Melissa Dell, Benjamin Charles Germain Lee, Jacob Carlson, and Weining Li. 2021. Layoutparser: A unified toolkit for deep learning based document image analysis. In *ICDAR*.

Abhishek Shende, Mahidhar Mullapudi, and Narayana Challa. 2024. Enhancing document verification systems: A review of techniques, challenges, and practical implementations. *International Journal of Computer Engineering & Technology*, 15:16–25.

Štěpán Šimsa, Milan Šulc, Michal Uřičář, Yash Patel, Ahmed Hamdi, Matěj Kocián, Matyáš Skalický, Jiří Matas, Antoine Doucet, Mickaël Coustaty, and 1 others. 2023. Docile benchmark for document information localization and extraction. In *International Conference on Document Analysis and Recognition*, pages 147–166. Springer.

Ray Smith. 2007. An overview of the tesseract ocr engine. In *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pages 629–633, Washington, DC, USA. IEEE Computer Society.

Tomasz Stanisławek, Filip Graliński, Anna Wróblewska, Dawid Lipiński, Agnieszka Kaliska, Paulina Rosalska, Bartosz Topolski, and Przemysław Biecek. 2021. *Kleister: Key Information Extraction Datasets Involving Long Documents with Complex Layouts*, page 564–579. Springer International Publishing.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 645–654.

Manan Suri, Puneet Mathur, Franck Dernoncourt, Kanika Goswami, Ryan A. Rossi, and Dinesh Manocha. 2025. VisDoM: Multi-document QA with visually rich elements using multimodal retrieval-augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6088–6109, Albuquerque, New Mexico. Association for Computational Linguistics.

Aliaksei Vertsel and Mikhail Rumiantsau. 2024. Hybrid llm/rule-based approaches to business insights generation from structured data. *Preprint*, arXiv:2404.15604.

Dongsheng Wang, Natraj Raman, Mathieu Sibue, Zhiqiang Ma, Petr Babkin, Simerjot Kaur, Yulong Pei, Armineh Nourbakhsh, and Xiaomo Liu. 2024. DocLLM: A layout-aware generative language model for multimodal document understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8529–8548, Bangkok, Thailand. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023a. Self-consistency

improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Zilong Wang, Yichao Zhou, Wei Wei, Chen-Yu Lee, and Sandeep Tata. 2023b. VRDU: A benchmark for visually-rich document understanding. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 5184–5193. ACM.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Anran Wu, Luwei Xiao, Xingjiao Wu, Shuwen Yang, Junjie Xu, Zisong Zhuang, Nian Xie, Cheng Jin, and Liang He. 2023. Dcqa: Document-level chart question answering towards complex reasoning and common-sense understanding.

Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. 2022. Tinyvit: Fast pretraining distillation for small vision transformers. In *European conference on computer vision*, pages 68–85. Springer.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 1192–1200. ACM.

Chong Zhang, Yi Tu, Yixi Zhao, Chenshu Yuan, Huan Chen, Yue Zhang, Mingxu Chai, Ya Guo, Huijia Zhu, Qi Zhang, and Tao Gui. 2024. Modeling layout reading order as ordering relations for visually-rich document understanding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9658–9678, Miami, Florida, USA. Association for Computational Linguistics.

Yifan Zhang, Yang Yuan, and Andrew Chi-Chih Yao. 2023. Meta prompting for ai systems.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Fengbin Zhu, Ziyang Liu, Xiang Yao Ng, Haohui Wu, Wenjie Wang, Fuli Feng, Chao Wang, Huanbo Luan, and Tat Seng Chua. 2024. Mmdocbench: Benchmarking large vision-language models for fine-grained visual document understanding.

A Appendix

A.1 Prompt used in LAVA

A.1.1 Prompt for Symbolic Reasoner

Dynamically Generated Prompt

You are an assistant who verifies financial documents. You are given {document_types} (in {languages}) and extracted {knowledge_names}. Your job is to verify the documents based on the validation rule. Please do not make up any values.

<instructions>
1. Use structured representation as references for identifying the layout and structure of the documents.
   <html_tables>{knowledge[tables]}</html_tables>
   <forms>{knowledge[forms]}</forms>
   ...
2. Use the semantic fields as references for providing explicit contextual information.
   <semantic_fields>{knowledge[fields]}</semantic_fields>

...

n. Output as follows:
   - Only output the fields requested in the validation rule, do not include any other fields stated in example_output.
   - Replace all spaces in keys with underscores "_".
   - If any field is not present, set value to "".
   - Output date in YYYYMMDD format.

   <example_output>{example_output}</example_output>
</instructions>

<validation_rule>
{validation_rule}
- {domain_specific_term_1} : {explanation_1}

...
- {domain_specific_term_n} : {explanation_n}
</validation_rule>

<output_format>
Return solely the JSON object without any additional explanations, comments, preamble, or formatting like backticks.
</output_format>

A.1.2 Prompt for Arithmetic Processor

Prompt for Formula Generation

System Prompt:
You are an assistant who generates a Python-style calculation formula to explain how a request field should be computed. You are given {document_types} (in {languages}), as well as knowledge extracted from the documents and a validation rule.

<instructions>
1. Identify the requested field from the rule. Store its name in the JSON object with the key "field_name", delete any special symbols and replace all spaces in keys with underscores "_".
2. Extract the stated value for the requested field.
   - Store it in the JSON object with the key "stated" and make sure it is a valid float or integer.
   - If the requested field is not present or empty, set "stated" to "NaN".
   - Do not make up any values.
3. Identify the relevant numerical fields in the knowledge that are needed to compute the requested field. Consider how those fields logically combine to produce the value of the requested field.
4. Once you find an appropriate calculation expression, store it in Python-executable format in the JSON object with the key "formula":
   - Do not compute the result.
   - Do not include any functions like "round(...)" or similar.
   - Only use raw numerical operations.
5. Return solely the JSON object without any additional explanations, comments, preamble, or formatting like backticks.
</instructions>

<example_output>
{
    "field_name": "Current_Total_Income",
    "stated": 600,
    "formula": "100 + 200 + 300"
}
</example_output>

User Prompt:
<knowledge>{knowledge}</knowledge>

<validation_rule>
{validation_rule}
- {domain_specific_term_1} : {explanation_1}

...
- {domain_specific_term_n} : {explanation_n}
</validation_rule>

Prompt for Formula Correction

System Prompt:
You are the greatest financial auditor, logician and deducer. You are given {document_types} (in {languages}), as well as knowledge extracted from the documents, a validation rule, and a calculation expression.

<instructions>
1. The calculation expression in response to the validation rule is wrong based on the knowledge extracted.
2. Check if there is any other correct calculation expression that can be derived from the data given.
3. Once you find an appropriate calculation expression, store it in Python-executable format:
   - Do not compute the result using equal sign.
   - Do not include any functions like "round(...)" or similar.
   - Only use raw numerical operations.
4. Return solely the calculation expression without any additional explanations, comments, preamble, or formatting like backticks.
</instructions>

<example_output_1>100 + 200 + 300</example_output_1>
<example_output_2>80 * 100.5</example_output_2>

User Prompt:
<knowledge>{knowledge}</knowledge>

<wrong_calculation>{wrong_calculation}<wrong_calculation>

<validation_rule>
{validation_rule}
- {domain_specific_term_1} : {explanation_1}

...
- {domain_specific_term_n} : {explanation_n}
</validation_rule>

A.2 Representative Validation Rules and Output Examples Used in Experiment

1. Content Extraction
   Rule:
   (a) Are Current and YTD regular pay/salary amounts present in documents?
   (b) Is Current and YTD CPP/QPP (may appear as Government Pension) present in the paystub?
   (c) Is Social Insurance Number (SIN) present in the T4 document?

   Output Example

   {
       "Employer_Name": {"present": true, "value": "organization"},
       "Current_Regular_Pay": {"present": false, "value": ""},
   }

2. Conditional Logic Reasoning
   (a) **Rule:**

<CRA_EI_data>{EI_data}</CRA_EI_data>
<EI_verification_rules>

- When Current EI is non-zero, skip the check and set "valid" to true.
- When Current and YTD EI are both 0 or blank, skip the check and set "valid" to false.
- When Current EI is 0 or blank, YTD EI is not 0 or blank. If YTD EI $\leq$ EI cap, set "valid" to true. Otherwise, set "valid" to false.

</EI_verification_rules>
Do the EI values comply with rules based on the Canada Revenue Agency (CRA) guidelines?

Output Example


```
{
    "Pay_Date": "20241128",
    "EI": {"Current": "100", "YTD": "200", "Cap":
"1049.12", "valid": true}
}
```


(b) **Rule:**
<numerical_verification_rules>

- Cells with empty value, or only numbers, symbols, punctuation are valid.
- Cells containing words (alphabetic characters) are invalid.
- If all fields are valid, return an empty object {}.

<numerical_verification_rules>
Based on the rules provided, does every non-header cell in tables contain a valid numerical value?

Output Example


```
{
    "This_Period_Regular": "amount",
    "YTD_CPP": "value"
}
```


3. Multi-step Logic Reasoning
Rule:
<pay_freq_cap>
[
{"freq": "monthly", "cap": 12},
{"freq": "semimonthly", "cap": 24},
{"freq": "biweekly", "cap": 26},
{"freq": "weekly", "cap": 52}
]
</pay_freq_cap>
<freq_verification_rules>

- If start or end date is empty, set "stated" to "" and set "equal" to None.
- If No. of pay period is empty, set "stated" to "" and set "equal" to None.
- If No. of pay period is not empty, find its corresponding "frequency cap" from pay frequency rules:
  - Compute the expected "current period number": Use "Pay Frequency", "Start Date" and "frequency cap" to calculate how many periods have elapsed since the start of the year.
  - Extract current period number from No. of pay period stated (e.g., 22 on "22 of 26").
  - Check if the "stated" current period number" equals the "expected current period number". If both equal, set "equal" to true. Otherwise, set "equal" to false.

</freq_verification_rules>
Based on the rules provided, does pay frequency and No. of pay period in the extracted date comply with the pay frequency rules?

Output Example


```
{
    "Start_Date": "20240101",
    "End_Date": "20240131",
    "Pay_Frequency": {"stated": "semimonthly", "expected":
"monthly", "equal": false},
    "NO_Pay_Period": {"stated": "11", "expected": "1",
"equal": false}
}
```


4. Unconstrained Arithmetic Consistency Checking
Rule:

(a) Is the YTD gross pay calculated correctly by summing up all earning items?
(b) Is Line 23600 (Net Income) correctly calculated as Line 15000 (Total Income) minus Deductions from Total Income?
(c) Is the ending balance correctly calculated as the starting balance plus total deposits minus total withdrawals for the period?

Output Example


```
{
    "Current_Gross_Pay": {"calculation": "100 + 200 + 300 =
600", "stated": "300", "equal": true},
    "YTD_Gross_Pay": {"calculation": "1500 + 200 + 300 =
2000", "stated": "", "equal": null}
}
```


5. Constrained Arithmetic Consistency Checking
Rule:

(a) Is the Current regular pay calculated correctly by rate * hours when rounding the result to 2 decimal places? Set formula to

an empty string"" when rate or hours is not present or 0.

(b) Is the current net pay correctly computed as the current gross pay minus all current deductions, with the result rounded to 2 decimal places? If the table includes a "Total Deduction" field, use it directly instead of summing individual deduction items.

Output Example


```
{
    "Current_Gross_Pay": {"calculation": "100 + 200 + 300 = 600", "stated": "300", "equal": true},
    "YTD_Gross_Pay": {"calculation": "1500 + 200 + 300 = 2000", "stated": "", "equal": null}
}
```


A.3 Evaluation Metrics

A.3.1 Factual Hallucination Rate (FHR).

Given a fixed rule r with evaluation document set $\mathcal{D}_r = \{d_1, \ldots, d_{n_r}\}$, let $\hat{\mathcal{E}}_{i,r}$ denote the predicted evidence subset in the model's output explanation trace for (d_i, r), and $\mathcal{E}(d_i, r)$ the gold evidence set from d_i. Let $\hat{\varphi}_{i,r}$ denote the predicted formula for (d_i, r), and g_r the corresponding gold formula.

We mark a sample (d_i, r) as factually hallucinated if either hold:

(a) **Evidence Hallucination:** $\hat{\mathcal{E}}_{i,r} \not\subseteq \mathcal{E}(d_i, r)$.

(b) **Formula Hallucination:** $\hat{\varphi}_{i,r} \not\equiv g_r$ (tested via polynomial identity testing with Schwartz–Zippel).

Formally, the sample-level indicator is

$$\mathrm{FH}_{i,r} = \mathbf{1}\big[\,(\hat{\mathcal{E}}_{i,r} \not\subseteq \mathcal{E}(d_i, r)) \;\vee\; (\hat{\varphi}_{i,r} \not\equiv g_r)\big]. \tag{1}$$

The rule-level rate is

$$\mathrm{FHR}_r = \frac{1}{|\mathcal{D}_r|} \sum_{d_i \in \mathcal{D}_r} \mathrm{FH}_{i,r}, \tag{2}$$

and the group-level average for $C \subseteq \mathcal{R}$ is

$$\mathrm{FHR}_C^{\mathrm{w}} = \frac{\sum_{r \in C} |\mathcal{D}_r| \cdot \mathrm{FHR}_r}{\sum_{r \in C} |\mathcal{D}_r|}. \tag{3}$$

Containment checks normalize case, punctuation, numeric formatting, and unit conventions. Formula equivalence ($\equiv$) is evaluated by polynomial identity testing under randomized substitution.

Example. Consider the rule "Is the YTD total deduction computed as the sum of all YTD items in the deduction table?" The pipeline is asked to extract relevant evidence, provide a calculation, and conclude the validation result for this task. Based on Figure 5, the ground truth is:

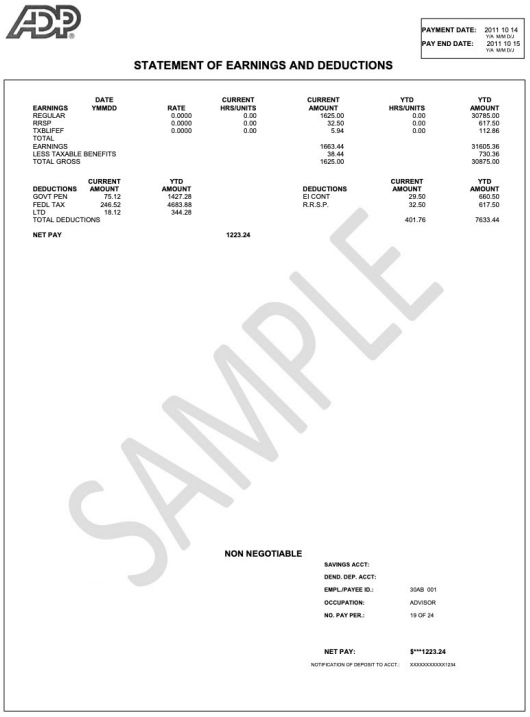


Figure 5: Sample document shown for clarity and error illustrations; metrics are document-agnostic and applied unchanged across all document types (sensitive content redacted).

Ground Truth Output


```
{
    "YTD_Total_Deduction": {
        "calculation": "1427.28 + 4683.88 + 344.28 + 660.50 + 617.50 = 7733.44",
        "stated": "7633.44",
        "equal": false
    }
}
```


When *factual hallucination* occurs, the pipeline fabricates one or some of the variables in the calculation formula (e.g., changing 660.50 to 560.50) so that the sum exactly matches the stated total:

Output with Factual Hallucination


```
{
    "YTD_Total_Deduction": {
        "calculation": "1427.28 + 4683.88 + 344.28 + 560.50 + 617.50 = 7633.44",
        "stated": "7633.44",
        "equal": true
    },
}
```


Consider the rule: "Are pay period start and end dates present in the document?" The pipeline is asked to check the presence of specific fields and extract the corresponding value. The ground truth for Figure 5 is:

Ground Truth Output


```
{
    "Start_Date": {"present": false, "value": ""}
    "End_Date": {"present": true, "value": "20111015"}
}
```


When *factual hallucination* occurs, the pipeline fabricates a start date by using the value of another field (in this case, the payment date) and incorrectly marking it as present as a consequence:

Output with Factual Hallucination


```
{
    "Start_Date": {"present": true, "value": "20111014"}
    "End_Date": {"present": true, "value": "20111015"}
}
```


These two examples illustrate distinct modes of factual hallucination. In the *deduction case*, the pipeline invents a new numerical value so that the arithmetic matches the stated total. In the *date case*, the pipeline fabricates a missing field by repurposing another field. In both scenarios, the error arises not from incorrect arithmetic or logical reasoning but from introducing evidence that is absent from the source document.

A.3.2 Numerical Infidelity Rate (NIR).

Given a fixed rule r with evaluation set $\mathcal{D}_r = \{d_1, \ldots, d_{n_r}\}$, let $\hat{\varphi}_{i,r}$ denote the predicted formula or reasoning process appearing in the model's output explanation trace for (d_i, r), and $v^\star_{i,r}$ the gold numerical result.

We mark a sample (d_i, r) as numerically infidel if the evaluated result of the predicted formula or reasoning process deviates from the gold value:

$$\mathrm{NI}_{i,r} = \mathbf{1}\left[\mathrm{eval}(\hat{\varphi}_{i,r}) \neq v^\star_{i,r}\right], \tag{4}$$

with deviations judged within absolute/relative tolerances.

The rule-level rate is

$$\mathrm{NIR}_r = \frac{1}{|\mathcal{D}_r|} \sum_{d_i \in \mathcal{D}_r} \mathrm{NI}_{i,r}, \tag{5}$$

and the group-level average for $C \subseteq \mathcal{R}$ is

$$\mathrm{NIR}^{\mathrm{w}}_C = \frac{\sum_{r \in C} |\mathcal{D}_r| \cdot \mathrm{NIR}_r}{\sum_{r \in C} |\mathcal{D}_r|}. \tag{6}$$

Note that hallucinations in formula generation are already captured under FHR; NIR isolates purely numerical inconsistencies after formula or reasoning process generation.

Example. Using the same rule for the YTD deduction value as in A.3.1:

Output with Numerical Infidelity


```
{
    "YTD_Total_Deduction": {
        "calculation":  "1427.28 + 4683.88 + 344.28 + 660.50 +
617.50 = 7633.44",
        "stated": "7633.44",
        "equal": true
    },
}
```


Here, all evidence terms are faithfully extracted, and the pipeline correctly generates the formula. But the result is miscomputed as 7633.44 instead of the correct 7733.44, possibly influenced by the stated value in the document. Unlike factual hallucination, this error does not arise from fabricating or misattributing values, but from applying the correct evidence while failing to maintain consistency in quantitative reasoning and arithmetic derivations.

A.3.3 Edge Case Handling (ECH).

For each rule r with evaluation set $\mathcal{D}_r$, we predefine a subset $\mathcal{D}_r^{\mathrm{edge}} \subseteq \mathcal{D}_r$ containing complex or exception-driven cases (typically 10%–25% of document–rule pairs). Such cases include missing or abnormal values, atypical field combinations, and boundary conditions requiring adaptive reasoning beyond commom patterns.

For a document $d_i \in \mathcal{D}_r^{\mathrm{edge}}$, let $\hat{y}_{i,r} \in \{\texttt{Pass}, \texttt{Fail}\}$ be the model's predicted label and $y^\star_{i,r}$ the gold label. We define the indicator

$$\mathrm{EC}_{i,r} = \begin{cases} 1, & \hat{y}_{i,r} \neq y^\star_{i,r}, \\ 0, & \hat{y}_{i,r} = y^\star_{i,r}. \end{cases} \tag{7}$$

The rule-level error rate is

$$\mathrm{ECH}_r = \frac{1}{|\mathcal{D}_r^{\mathrm{edge}}|} \sum_{d_i \in \mathcal{D}_r^{\mathrm{edge}}} \mathrm{EC}_{i,r}, \tag{8}$$

and the group-level average is

$$\mathrm{ECH}^{\mathrm{w}}_C = \frac{\sum_{r \in C} |\mathcal{D}_r^{\mathrm{edge}}| \cdot \mathrm{ECH}_r}{\sum_{r \in C} |\mathcal{D}_r^{\mathrm{edge}}|}. \tag{9}$$

Higher values of ECH indicate weaker generalization and lower robustness on edge conditions.

Example. Consider the rule: "Is the current gross pay calculated correctly by summing up the current amount of all earning items?" In the sample (Figure 5), the earnings table contains a subtotal of 1663.44 followed by a negative adjustment of 38.44. Correct handling requires the pipeline to exclude the subtotal from the formula and include

the adjustment as a subtraction. The ground truth is:

Ground Truth Output


```
{
    "Current_Gross_Pay": {
        "calculation": "1625.00 + 32.50 + 5.94 - 38.44 = 1625.00",
        "stated": "1625.00",
        "equal": true
    }
}
```


However, the pipeline misinterprets the table:

Output with Edge Case Error


```
{
    "Current_Gross_Pay": {
        "calculation": "1625.00 + 32.50 + 5.94 + 1663.44 + 38.44 =
3365.32",
        "stated": "1625.00",
        "equal": false
    },
}
```


In this output, the pipeline incorrectly reuses the subtotal 1663.44 as if it were another earning item, and also flips the negative adjustment 38.44 into a positive contribution. The error is not due to arithmetic mistakes but to misinterpreting atypical structures in the table. Unlike factual hallucination (fabricating values) or numerical infidelity (miscomputing a correct formula), this case reflects weaker robustness to edge conditions. Instead of applying rules mechanically, a reliable industrial pipeline should capture the underlying business logic and correctly adapt it to the diverse formats and exception cases present in financial documents.

A.4 Evaluation Dataset

Our evaluation uses a proprietary collection of production-level mortgage application documents from an active industrial workflow. The corpus spans a broad range of core and supporting document categories—covering proof of income, property appraisal, account statements, tax forms, and legal agreements—with file lengths varying from single-page forms to multi-dozen-page reports. Unlike public datasets that are often synthetic or visually uniform, our corpus retains the full heterogeneity, layout irregularities, OCR noise, and compliance constraints encountered in real underwriting. While individual files and categories vary across evaluations, the overall data composition and associated challenges are fully described, enabling reproducibility on comparable financial document collections.

Table 3: Representative mortgage document types and key characteristics.

Document Type	**Key Structural Features**	**Key Semantic Features**	**Valid Rule Categories**
Appraisal	Multi-page PDF; photos + valuation tables; firm templates; checkboxes	Market value; comparable properties; adjustments; effective date	C_1, C_2
Bank Account Statement	Multi-column ledgers; footnotes inside tables; embedded logos and stamps	Balance progression; transaction category; cross-month reconciliation	$C_1 – C_5$
Employment Letter	Letterhead; signatures; uneven paragraphs; low-contrast scans	Title; start date; compensation; pay frequency vs. paystubs	C_1, C_2, C_3
Investment Account Statement	Dense holdings tables; various formats; small-font disclosures and disclaimers	Issue/maturity date; interest rate; holding list; balance	C_1, C_3, C_4, C_5
Mortgage Statement	Multi-section layout; various formats; tables with merged cells	Outstanding balance; interest rate and type; payment due date; amount due	C_1, C_3, C_4, C_5
Notice of Assessment	CRA form; dense numeric blocks; tiny labels; presence of official headers	Total/taxable income; refund amount; social insurance number; match T4	C_1, C_2, C_3, C_4
Paystub	Multiple tables and forms; various layouts; faded scans	Gross; net; deductions; start/end/pay date	C_1 - C_5
Personal Income Tax Return	Multipage; mixed sections; checkboxes; pre-filled and handwritten; cross-page linkage	Address; citizenship; total/employment/net income	C_1, C_2, C_4
Property Tax Bill	Multiple tables; various layouts; inconsistent labelling for tax components; watermarks and logos	Annual tax; installments/penalties; roll/assessment IDs; schedule math	C_1 - C_5
Purchase Sales Agreement	Long contract; initials/signatures; clause order varies; appendices with separate numbering	Property address; price; closing date; buyer/seller name	C_1, C_2, C_3
Realtor Listing	Embedded images; dense tables and forms; inconsistent field ordering	List price; property address; listing date and status; square footage	C_1, C_2
Realtor Listing UW	Listing + UW notes; handwritten overlays	Adjusted price; remarks; reconcile with appraisal/APS	C_1, C_2, C_3
Rental Lease Agreement	Multi-page legal contract; various templates; small-font clauses; handwritten	Monthly rent; term dates; payment plan; residence overlap	C_1, C_2, C_3
Separation Agreement	Multi-page legal document; dense narrative clauses; annex schedules	Support obligations; asset division; enforceable clauses; child custody	C_1, C_2, C_3
T4: Statement of Remuneration Paid	Fixed CRA box layout; small-font numeric fields; bilingual field labels; scan noise	Employment income; Province; EI/CPP exemptions; align paystub/NOA	C_1, C_2, C_4
T4A: Statement of Pension, Retirement, Annuity, and Other Income	Fixed CRA box layout; small-font numeric fields; bilingual field labels; scan noise	Pension/annuity/self-employment income; income tax deducted; social insurance number	C_1, C_2, C_4
T4A(P): Statement of Canada Pension Plan Benefits	Fixed CRA box layout; small-font numeric fields; bilingual field labels; scan noise	CPP benefits amount; Income tax deducted;	C_1, C_2, C_4
T4RIF: Statement of Income From a Registered Retirement Income Fund	Fixed CRA box layout; small-font numeric fields; bilingual field labels; scan noise	Taxable amounts; Payer/issuer name; taxable amount; social insurance number	C_1, C_2, C_4
T5: Statement of Investment Income	Fixed CRA box layout; small-font numeric fields; bilingual field labels; scan noise	Actual amount of dividends; Interest from Canadian sources; Reported income vs. bank statements	C_1, C_2, C_5
T776: Statement of Real Estate Rentals	Fixed CRA box layout; small-font numeric fields; bilingual field labels; scan noise	Gross rents; Total expenses; Net income (loss) before adjustments; Capital cost allowance (CCA) claim	C_1 - C_5