Engineering a LLM-Based Data Ingestion Process for Semi-Structured Documents: An Empirical Comparison with Human-Centered Workflows

Sandrelly L. Coutinho[1], Paulo J. L. Adeodato[1], Bruno C. de Souza[1], Carlos E. de S. Fontes[2]

[1] Universidade Federal de Pernambuco (UFPE)

[2] Centro Universitário Maurício de Nassau (UNINASSAU)

slc@cin.ufpe.br, pjla@cin.ufpe.br, bcampello@uol.com.br, cadufontes@gmail.com

Abstract

Data-driven software systems increasingly rely on ingesting information from semi-structured documents, whose manual processing is costly and error-prone. Although Large Language Models (LLMs) can interpret such documents, their systematic integration into reliable ingestion processes remains underexplored. This paper proposes an LLM-driven data ingestion process from a Software Engineering perspective, distinguishing prompt-based optimization from model training. Prompts are treated as versioned engineering artifacts and refined iteratively through automated comparison with ground truth under stateless execution. A controlled experiment evaluates multiple LLM backends using extraction quality, cost, and generalization.

Keywords*: Large Language Models, Data Ingestion, Semi-Structured Documents, Prompt Engineering, Empirical Software Engineering.*

1. Introduction

Reliable and timely data ingestion is a fundamental requirement for data-driven software systems. In industrial environments, however, critical information is frequently distributed as semi-structured documents—such as PDFs and spreadsheets—whose layouts, terminology, and conventions vary across organizations and evolve over time. This heterogeneity introduces recurring engineering effort, operational bottlenecks, and an increased risk of inconsistencies, particularly in domains characterized by frequent updates and large document volumes.

This challenge is especially evident in real estate market intelligence platforms, where up-to-date unit-level information (e.g., identifiers, prices, and commercial status) must be continuously extracted from pricing tables published by developers. In practice, these documents are processed through human-centered workflows involving manual interpretation, data registration, and quality assurance. Although this approach can achieve acceptable accuracy, it scales poorly and becomes increasingly costly as the number of monitored projects grows.

Recent advances in Large Language Models (LLMs) have demonstrated strong capabilities in text understanding and information extraction, motivating their adoption

for transforming semi-structured documents into structured data (Zheng et al., 2023; Ahmad et al., 2024; Tiwari et al., 2023). However, applying LLMs to industrial ingestion pipelines raises key Software Engineering challenges. Extraction tasks differ substantially in complexity, ranging from explicit numerical parsing to inferential and structurally inferred attributes. Moreover, many reported applications rely on ad-hoc prompting and single-shot evaluation, offering limited insight into whether LLM-based ingestion processes can systematically improve over time or generalize beyond observed documents (Zheng et al., 2023; Ahmad et al., 2024). Comparisons across LLMs are also often conducted as isolated benchmarks without controlling the surrounding process architecture, and practical trade-offs involving quality, stability, and operational cost relative to human-centered workflows remain insufficiently quantified.

This paper addresses these challenges by framing LLM-based data ingestion as an engineered software process rather than as an isolated application of a language model. We propose an iterative ingestion pipeline in which prompts are treated as versioned engineering artifacts, LLM executions are stateless to ensure reproducibility, outputs are automatically compared against manually curated ground truth, and structured feedback enables autonomous prompt refinement. Extraction errors and operational indicators—such as token consumption and estimated cost—are systematically recorded to support empirical analysis of learning behavior, convergence, and efficiency.

An empirical evaluation was conducted using 80 heterogeneous real estate pricing tables obtained from an operational data intelligence platform. Sixty documents were used during iterative prompt refinement, while the remaining 20 were reserved for explicit generalization evaluation. Two widely adopted LLM backends were embedded within the same ingestion architecture and executed under identical stateless conditions. After refinement, the final prompt was frozen and applied to unseen documents to assess generalization. The evaluation analyzes extraction quality across attribute types, learning and convergence behavior, operational cost, and contrasts the automated pipeline with a human-centered baseline.

This paper contributes an engineered LLM-driven ingestion process, a controlled empirical evaluation across attribute types, a convergence-oriented analysis of iterative prompt refinement, a cross-model comparison, and a practical comparison with a human-centered baseline.

2. Related Work

Large Language Models (LLMs) have been increasingly explored in Software Engineering research, with applications spanning code generation, program comprehension, testing, requirements engineering, and rule interpretation. Surveys and systematic reviews highlight both the potential of LLMs and persistent challenges related to reliability, reproducibility, cost, and evaluation rigor when integrating these models into software systems (Zheng et al., 2023; Ahmad et al., 2024). However, most existing work focuses on task-level assistance or isolated applications rather than on engineered end-to-end processes that embed LLMs as operational components.

Empirical studies comparing LLMs under controlled conditions show that model behavior is highly sensitive to task formulation and prompt design (Silva et al., 2025). Practitioner-focused studies further indicate that, although LLMs are increasingly used

in real-world software development, concerns remain regarding correctness, predictability, explainability, and operational cost (Rasnayaka et al., 2024; Wyrich et al., 2025).

Parallel research has explored the use of LLMs for structured information extraction from heterogeneous and semi-structured documents. Prior work shows that LLMs can generate structured representations from complex document collections without relying on rigid rule-based pipelines (Arora et al., 2023), but also reveals strong sensitivity to prompt formulation and task specification (Tiwari et al., 2023). Most of these studies address single-step extraction tasks and do not analyze prompt evolution, learning dynamics, or robustness across successive refinements.

More recent contributions propose integrating LLMs into broader data ingestion and analytics pipelines (Kumar et al., 2024; Sharma et al., 2024). Although these studies demonstrate architectural feasibility, they generally lack rigorous empirical validation and do not compare LLM-based pipelines against fully manual data engineering workflows using consistent quality, cost, and scalability metrics. Survey work further identifies the absence of reusable engineering processes supporting iterative improvement, performance monitoring, and controlled human validation as a key open problem in this area (Zhou et al., 2025).

In contrast to prior work, this paper frames LLM-driven data ingestion as a Software Engineering problem and evaluates an engineered process with autonomous prompt refinement, controlled comparison, and generalization assessment.

3. Problem Definition and Research Questions

The continuous availability of accurate and up-to-date data is a fundamental requirement for modern data-driven software systems. In domains such as real estate intelligence, this requirement is particularly challenging because critical information is published in semi-structured documents whose layout, terminology, and implicit assumptions vary across organizations and over time. As a result, automated data ingestion becomes a Software Engineering problem involving structure recognition, inference, validation, and process control.

In current industrial practice, this type of ingestion relies heavily on human-centered workflows. Data engineers manually interpret pricing tables, infer missing or implicit information, and register unit-level attributes into structured systems, followed by quality assurance steps performed by additional personnel. While this workflow can achieve reasonable accuracy, it scales poorly, incurs high operational cost, and becomes increasingly difficult to sustain as document volumes grow.

Large Language Models (LLMs) offer a potential alternative because they can interpret semi-structured text and generate structured outputs. However, most existing applications treat LLMs as black-box tools applied in isolation, often using ad-hoc prompts and static evaluation. From a Software Engineering perspective, several questions remain open: how LLM-based ingestion behaves across different types of extraction tasks, whether systematic prompt refinement can induce learning and stable convergence, how different LLM backends behave under the same engineered process, and whether such pipelines are viable when compared with human-centered workflows in terms of accuracy, scalability, and cost.

This work addresses these challenges by framing LLM-based data ingestion as an engineered software process rather than as a standalone use of a language model. The

proposed approach embeds LLMs as first-class components within a controlled iterative pipeline that combines stateless execution, ground truth comparison, autonomous prompt refinement, and systematic metric collection. Particular emphasis is placed on assessing whether improvements obtained through iterative refinement generalize to previously unseen documents.

Based on this problem formulation, the study is guided by the following research questions:

RQ1 (Extraction Quality): How does an LLM-driven ingestion process perform across different types of attributes (explicit numerical, inferential, and structurally inferred) when extracting structured data from semi-structured documents?

RQ2 (Learning and Convergence): Does iterative prompt refinement enable learning and stable convergence in LLM-based data ingestion processes, and how does this behavior vary across LLM backends?

RQ3 (Cross-Model Behavior): How do different LLM backends behave when embedded within the same engineered ingestion process under identical experimental conditions?

RQ4 (Practical Viability): How does an LLM-driven ingestion process compare to a human-centered baseline in terms of accuracy, scalability, and operational cost?

4. Proposed Approach

This section presents the proposed LLM-driven data ingestion approach from a software engineering perspective. Rather than treating Large Language Models as opaque black-box tools, the approach embeds them within a structured, iterative, and measurable ingestion process. Learning is induced exclusively through prompt evolution, without any modification to the underlying model parameters.

The approach is designed to support reliable extraction of structured, unit-level data from heterogeneous semi-structured documents, while enabling systematic evaluation, traceability, and comparison across different LLM backends.

4.1 Process Overview

The proposed ingestion process follows an iterative execution loop composed of document processing, structured output generation, automated validation against ground truth, and prompt refinement. Each iteration processes a single document in isolation and produces both extraction results and diagnostic artifacts used to guide subsequent refinements.

At a high level, the process operates as follows: The process begins when a semi-structured pricing table is provided as input and the current prompt version is applied to extract structured data. The generated output is then automatically compared against a manually curated ground truth dataset, producing structured error descriptions that are used to refine the prompt for the next iteration.

This loop is repeated across successive documents, enabling gradual improvement of extraction behavior while preserving strict control over execution conditions.

4.2 Reference Architecture

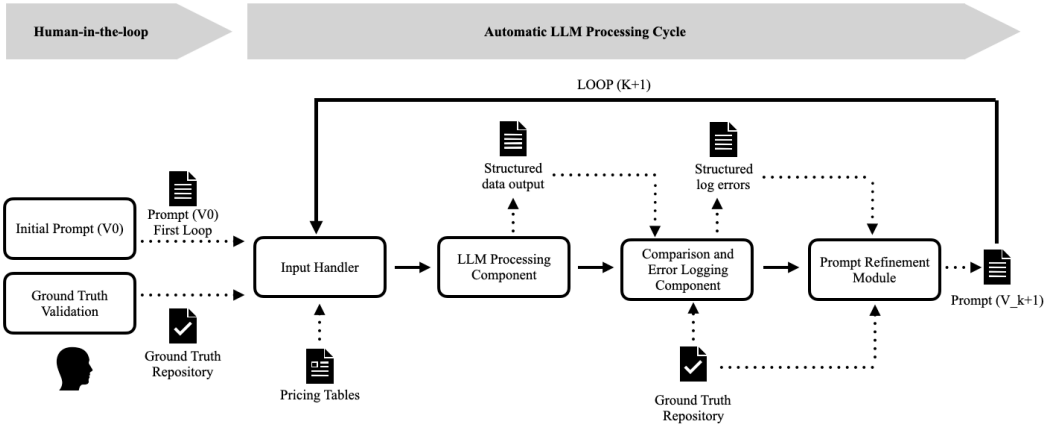

Figure 1. Reference architecture of the proposed LLM-driven data ingestion process with iterative prompt refinement.

Figure 1 illustrates the reference architecture of the proposed LLM-driven data ingestion process. The architecture is composed of six main components, each responsible for a specific function within the engineered workflow.

The architecture integrates both automated processing and optional human oversight, while ensuring that learning remains encapsulated within prompt evolution.

The architecture comprises an Input Handler, an LLM Processing Component, a Comparison and Error Logging Component, a Prompt Refinement Module, a Ground Truth Repository, and an optional Human-in-the-Loop Interface.

This architecture ensures traceability across prompt versions, repeatability of experiments, and strict separation between model execution and learning artifacts.

4.3 Prompt Structure and Engineering Strategy

In the proposed approach, prompts are treated as versioned engineering artifacts rather than ad-hoc instructions. The initial prompt (Prompt V0) is authored using domain expertise and defines the extraction task, expected output schema, output constraints, domain-specific rules, and the input document.

The initial prompt instructs the LLM to extract unit-level information from pricing tables and generate structured output containing unit identifiers, final price, and commercial status. It also encodes domain-specific rules, such as selecting the maximum monetary value as the final price when multiple financial values appear in the same row and inferring sold units from absent records when supported by document context.

From Prompt V1 onward, prompt refinement is performed autonomously by the LLM itself. At each iteration, the system provides the previous prompt version, the structured extraction output, the corresponding ground truth, and a structured error report generated by the comparison module. Based on these artifacts, the LLM revises the prompt by clarifying ambiguous rules, strengthening output constraints, and correcting previously observed failure modes. Once the iterative loop begins, no human intervention occurs during prompt refinement. For example, if an iteration fails to identify sold units mentioned outside the main tabular region, the error report records this mismatch and the next prompt version strengthens the corresponding inference rule.

In practice, prompt evolution is therefore driven by error-guided specification updates rather than by changes to model parameters. This design enables learning to be incorporated at the prompt level while preserving stateless execution, traceability across prompt versions, and reproducibility of the overall ingestion process.

4.4 Iterative Prompt Refinement Loop

Each iteration submits the current prompt and one input document to the LLM, generates structured output, compares it against ground truth, records attribute-level errors, and produces a refined prompt version for the next iteration.

All prompt versions are preserved, enabling analysis of learning behavior, convergence patterns, and cross-model differences. The same refinement protocol is applied independently to each LLM backend, ensuring experimental comparability.

4.5 Generalization-Oriented Execution Design

To assess whether improvements achieved during iterative refinement generalize beyond the learning phase, the proposed approach explicitly separates refinement and evaluation stages. After the refinement loop concludes, the final prompt version is **frozen** and applied to previously unseen documents.

No further prompt updates are allowed during this phase. This design enables the evaluation of robustness and transferability under realistic conditions, where document structures differ from those encountered during refinement.

This generalization-oriented execution design is central to assessing practical viability and directly supports the research questions related to learning stability and cross-document robustness.

5. Experimental Design

This section describes the experimental design adopted to evaluate the proposed LLM-driven data ingestion process. The design aims to ensure fairness, reproducibility, and comparability across different Large Language Model (LLM) backends, while enabling systematic analysis of extraction quality, learning behavior, generalization, and operational efficiency.

5.1 Dataset and Ground Truth

The experimental evaluation was conducted using a collection of semi-structured real estate pricing tables obtained from an operational market intelligence platform. These documents are published by real estate developers and construction companies and constitute the primary source for updating unit-level information in the platform.

A total of 80 pricing tables were used in the study. From this set, 60 documents were employed during the iterative prompt refinement phase, while the remaining 20 documents were reserved exclusively for the generalization evaluation. The two subsets are strictly disjoint, ensuring that generalization performance is assessed on previously unseen document structures.

The pricing tables exhibit substantial heterogeneity in layout, column naming conventions, unit aggregation strategies, monetary formatting, and representation of commercial status. Some tables explicitly list units and prices, while others require inference from narrative text, missing records, or document-specific conventions. This

variability was intentionally preserved to evaluate the robustness of the proposed ingestion process under realistic industrial conditions.

For each pricing table, a corresponding ground truth dataset was manually constructed prior to the experiments. Ground truth construction required manual inspection of each document and registration of the expected unit-level outputs according to domain-specific business rules. These outputs include unit identifiers, final price, and commercial status.

Ground truth annotations were created by domain specialists with experience in real estate data engineering and independently validated before being used in the experiments. This validation step reduced the risk of inconsistencies in the reference data and ensured that the same ground truth was used throughout all iterations and across all evaluated LLM backends.

The ground truth datasets served two purposes: first, they provided a stable reference for computing attribute-level extraction error rates; second, during the refinement phase, they were used as structured feedback to support autonomous prompt evolution. Ground truth data remained fixed throughout the study and was never modified during the iterative process.

By combining heterogeneous real-world documents with manually validated reference data, the experimental design supports a controlled evaluation of process behavior under realistic industrial constraints, while not aiming at statistical generalization across domains.

5.2 Baseline: Human-Centered Data Ingestion

To contextualize the proposed approach, a human-centered data ingestion workflow currently used in the platform was adopted as the baseline.

In this workflow, pricing tables are automatically collected via APIs as PDF files from heterogeneous systems maintained by real estate developers. A data engineering team then manually inspects each document, interprets its structure, and registers unit-level information through structured forms associated with each real estate project. This step includes identifying units, prices, and commercial status, as well as inferring implicit information when required by the document layout or domain conventions.

After manual registration, a separate quality assurance team performs an independent verification step. This team reviews the registered data, checks for inconsistencies or omissions, and corrects errors before publication. The presence of this verification layer is important because it reflects the actual operational workflow used in production rather than a simplified manual extraction scenario.

Operational measurements indicate that manual registration requires an average of 12.3 minutes per pricing table, while the quality verification step requires an additional 4.6 minutes, resulting in a total average processing time of approximately 16.9 minutes per project. Despite this two-stage workflow, the observed average error rate is 8.9%, considering incorrect values, missing units, and misclassified commercial status.

The scalability limitations of this baseline become evident at larger operational scales. Since both registration and verification depend on human effort, the total processing time grows linearly with the number of projects, making frequent large-scale updates increasingly costly and difficult to sustain.

This human-centered workflow serves as the baseline against which the proposed LLM-driven ingestion process is evaluated in terms of extraction quality, processing effort, and operational viability.

5.3 LLM Configuration and Experimental Protocol

This section describes the configuration of the Large Language Models (LLMs) and the experimental protocol adopted to evaluate the proposed data ingestion process. The goal of this protocol is to ensure a fair, controlled, and reproducible comparison across different LLM backends while keeping the ingestion process itself unchanged.

5.3.1 LLM Backends and Execution Setup

The proposed ingestion process was evaluated using two widely adopted large language model backends accessed through their official APIs and orchestrated via intelligent agents: GPT, using the model *GPT-5.2-2025-12-11*, and Gemini, using the model *Gemini-3-pro-preview*.

These two backends were selected because they represent widely used state-of-the-art commercial LLM ecosystems with strong document understanding capabilities, stable API access, and practical relevance for industrial deployment. Evaluating both within the same engineered ingestion process enables a controlled comparison of backend behavior without altering the surrounding architecture, refinement protocol, or evaluation criteria.

Both models were integrated into the same ingestion architecture and executed under identical conditions. Each document was processed through a dedicated and independent stateless execution, with no retention of dialogue history or contextual information across documents or iterations.

All model invocations were allowed to consume the maximum number of tokens required to complete each extraction task, without exceeding the model-specific limits. The remaining model parameters were kept fixed across executions to ensure that observed differences could be attributed to backend behavior rather than configuration variance.

5.3.2 Iterative Prompt Refinement Protocol

The experimental protocol follows the iterative ingestion process described in Section 4 and illustrated in Figure 1. During the refinement phase, 60 pricing tables were processed sequentially, with each iteration corresponding to a single document.

Each iteration consists of the following steps: (i) submission of the current prompt version (Prompt Vk) and the input document to the LLM; (ii) generation of structured extraction output; (iii) automatic comparison of the generated output against the corresponding ground truth dataset; (iv) generation and recording of structured error descriptions; and (v) autonomous refinement of the prompt specification by the LLM, producing a new prompt version (Prompt Vk+1).

Prompt refinement is fully automatic during the experimental loop. Once the initial prompt has been authored and the ground truth datasets validated, no human intervention occurs during subsequent refinement steps. At each iteration, the system provides the LLM with the previous prompt version, the structured extraction output, the corresponding ground truth, and the structured error report generated by the comparison module. Based on these artifacts, the LLM revises the prompt by clarifying

ambiguous rules, strengthening output constraints, and correcting previously observed failure modes.

For example, if an iteration fails to identify sold units mentioned outside the main tabular region, the resulting error report records this mismatch against ground truth, and the next prompt version strengthens the corresponding inference rule. Similar refinements are performed for final-price selection when multiple monetary values appear in the same row.

Each refined prompt fully replaces the previous version, and all prompt versions are preserved to support traceability and learning analysis. The same initial prompt, document order, and refinement protocol were applied independently to both LLM backends.

5.3.3 Generalization Evaluation Protocol

To evaluate robustness and transferability, the final refined prompt obtained after the 60th iteration was frozen and applied to the remaining **20 unseen pricing tables**.

No further prompt modifications were allowed during this phase. The same execution and evaluation procedures used during refinement were applied, enabling explicit assessment of whether improvements obtained through iterative prompt refinement generalize beyond the learning cycle.

5.4 Evaluation Metrics and Comparative Criteria

Extraction quality is evaluated using attribute-level error rates computed by comparing LLM-generated outputs against ground truth data. The evaluation considers attributes with different semantic and structural characteristics, including fully inferential attributes (sales status), structurally inferred attributes (missing units), and explicit numerical values (final price).

In addition to extraction quality, operational efficiency is assessed through token consumption, estimated operational cost, and processing time per document when applicable. Comparative analysis is conducted along two dimensions: a cross-LLM comparison under identical experimental conditions and a comparison between the LLM-based ingestion process and the human-centered baseline, focusing on accuracy, scalability, and reduction of human effort.

All quantitative indicators, figures, and comparative tables derived from these metrics are reported and analyzed in Section 6 (Results).

6. Results

This section presents the experimental results obtained from the evaluation protocol described in Section 5. Results are organized according to extraction quality, learning behavior, operational efficiency, and comparative analyses, following the same structure defined in the evaluation design.

6.1 Extraction Quality by Attribute

This subsection reports attribute-level extraction accuracy.

Figure 6.1 presents the evolution of the error rate for sales status classification across iterations. The GPT backend exhibits persistently high error rates, frequently exceeding 85%, with no evidence of sustained convergence. In contrast, the Gemini

backend achieves stable convergence after approximately iteration 22, maintaining near-zero error rates for the remainder of the process.

Figure 6.1. Learning and convergence behavior for sales status extraction under iterative prompt refinement (GPT-5.2 and Gemini-3).

Figure 6.2 shows error rates for missing unit inference. GPT fails to generalize structural inference rules, with error rates remaining above 80% throughout most iterations. Gemini demonstrates rapid convergence, achieving near-zero error after a small number of iterations and maintaining stable performance across heterogeneous document structures.

Figure 6.2. Error rate evolution for missing unit inference across iterative prompt refinement (GPT-5.2 vs. Gemini-3).

As shown in Figure 6.3, both LLMs perform well on final price extraction. Error rates remain close to zero for most iterations, with only isolated spikes associated with atypical pricing structures. These errors do not persist and do not affect overall convergence.

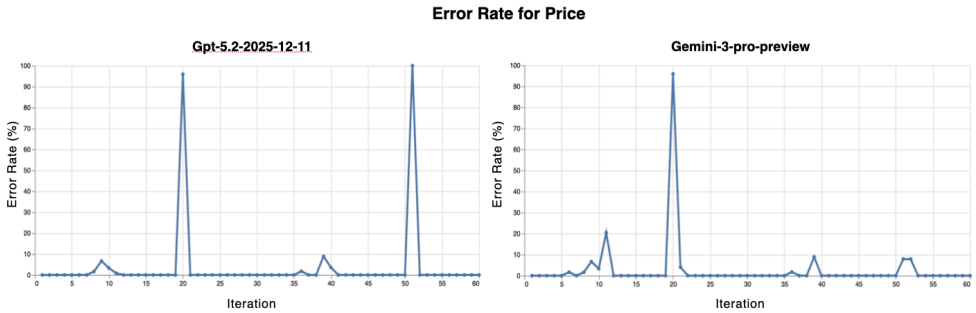

Figure 6.3. Error rate evolution for final price extraction across iterative prompt refinement (GPT-5.2 vs. Gemini-3).

6.2 Learning Behavior and Convergence Analysis

Across iterations, GPT shows prolonged high-error plateaus on inferential and structurally inferred attributes, with no sustained convergence. Gemini exhibits clearer learning phases and reaches stable convergence after transient regressions, indicating that prompt refinement can encode progressively stronger extraction rules.

Table 6.1. Convergence behavior and error rates by attribute and LLM backend.

Attribute	Model	Convergence	Avg. Error (Train)	Post-Conv. Error	Final Error
Sales Status	GPT-5.2	No	85–90%	N/A	68–70%
Sales Status	Gemini-3	Yes	20–25%	<2%	0%
Missing Units	GPT-5.2	No	80–85%	N/A	70%
Missing Units	Gemini-3	Yes	6–8%	<1%	0%
Final Price	GPT-5.2	Yes	3–5%	0%	0%
Final Price	Gemini-3	Yes	4–6%	<1%	0%

6.3 Operational Efficiency and Cost

Figure 6.5 shows token consumption per iteration. GPT consistently consumes fewer tokens on average, while Gemini requires higher token budgets due to richer prompt structures. However, neither model exhibits cumulative token growth across iterations.

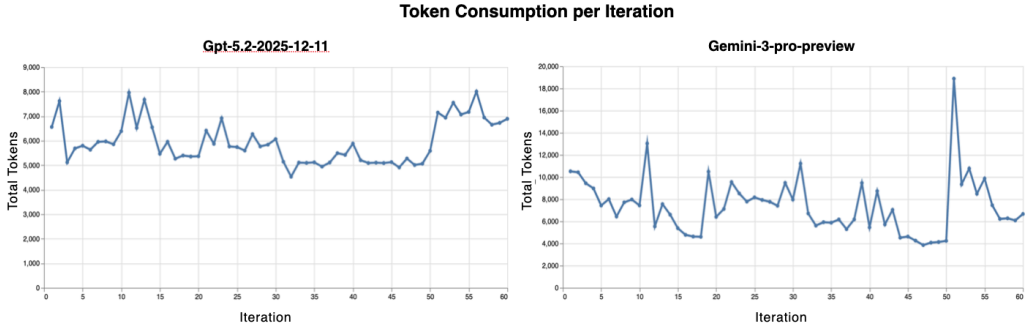

Figure 6.5. Token consumption per iteration across iterative prompt refinement (GPT-5.2 vs. Gemini-3).

Figure 6.6 presents the operational cost per iteration. Gemini incurs higher average costs but remains within predictable and bounded limits.

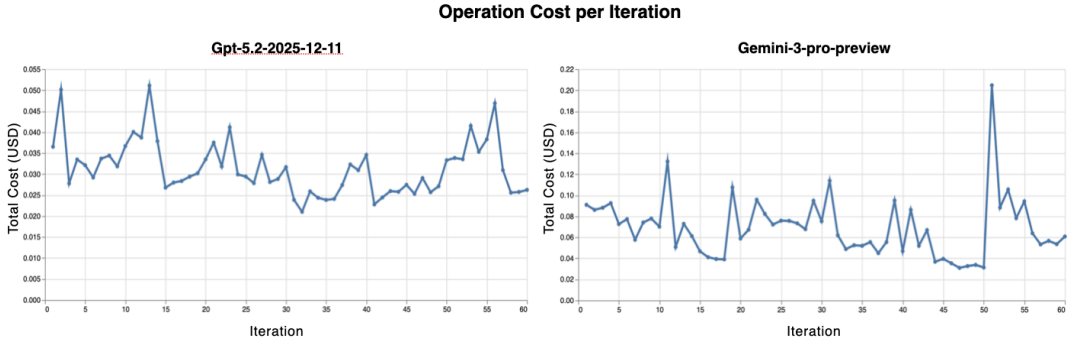

Figure 6.6. Operational cost per iteration across iterative prompt refinement (GPT-5.2 vs. Gemini-3).

Table 6.2. Token consumption and operational cost per iteration (USD).

Metric	GPT-5.2	Gemini-3

Avg. Tokens / Iteration	5,800–6,000	7,500–8,000
Min Tokens	4,700	4,000
Max Tokens	8,000	19,000
Avg. Cost / Iteration	0.030–0.032	0.065–0.070
Min Cost	0.021	0.030–0.035
Max Cost	0.052	0.20–0.21
Cost Growth Over Iterations	Stable	Stable

6.4 Comparative Evaluation

Table 6.3 compares the LLM-based ingestion pipeline with the human-centered baseline. While manual ingestion requires substantial human effort and exhibits higher error rates, the Gemini-based pipeline significantly reduces processing time and shifts human involvement from data entry to validation.

Table 6.3: Comparison between human-centered and LLM-based data ingestion pipelines.

Dimension Ingestion Data	Human-Centered Process	LLM-Based (Gemini)
Avg. Time per Pricing Table	12.3 min	1.6 min
Avg. Error Rate	8.9%	<2%
Scalability	Low	High
Cost Structure	Linear (human hours)	Sub-linear (API cost)
Operational Bottleneck	Manual processing	None

6.5 Generalization Results

To assess generalization, the final refined prompt obtained after 60 iterations was applied to 20 previously unseen pricing tables. Gemini maintained near-zero error across all attributes, indicating that prompt-level learning generalized beyond the refinement set. GPT showed no meaningful improvement on inferential and structural attributes, mirroring its refinement-phase behavior.

7. Discussion

This section discusses the results presented in Section 6 in light of the research questions defined in Section 3. The discussion focuses on interpreting observed patterns, explaining behavioral differences across attribute types and LLM backends, and deriving implications for the engineering of LLM-based data ingestion systems. Particular attention is given to the distinction between convergence observed during iterative refinement and performance obtained during generalization on unseen documents.

7.1 Extraction Quality Across Attribute Types (RQ1)

Extraction quality is strongly influenced by the semantic and structural nature of the target attribute. Explicit numerical attributes, such as final price, converge rapidly and achieve low error rates, reflecting their direct presence in pricing tables.

Inferential and structurally inferred attributes, including sales status and missing units, present greater challenges. Their correct extraction depends on implicit cues,

absent records, or domain-specific conventions, leading to higher initial error rates and slower convergence.

These results show that aggregate accuracy metrics obscure meaningful differences across tasks. From a Software Engineering perspective, ingestion pipelines should treat attribute extraction as a set of semantically distinct subproblems rather than as a single homogeneous task.

7.2 Learning Dynamics and Convergence Behavior (RQ2)

Iterative prompt refinement induces measurable learning behavior, but learning is neither uniform nor monotonic. Low error rates observed in isolated iterations do not constitute convergence; stable convergence is achieved only when performance remains consistently low across successive iterations.

Prompt refinement operates as a constrained learning mechanism: improvements arise from incremental clarification of task specifications rather than from model parameter updates. Consequently, learning capacity is bounded by the model's ability to internalize and generalize increasingly precise instructions.

The generalization evaluation confirms that convergence during refinement does not automatically imply robustness. Only sustained performance under frozen-prompt execution provides evidence that learning has been effectively incorporated into the prompt itself.

This distinction is also important to reduce the risk of prompt overfitting, in which refinements improve performance on previously seen structures without yielding robust behavior on unseen documents.

7.3 Cross-LLM Behavioral Differences Under Identical Processes (RQ3)

When embedded within the same engineered ingestion process, the evaluated LLM backends exhibit markedly different behavioral profiles. Gemini converges faster, maintains lower error rates, and shows greater stability across iterations. GPT exhibits higher variability, including oscillations and regressions after partial improvement.

These differences should not be interpreted as absolute measures of model quality. Instead, they demonstrate that LLMs respond differently when integrated into iterative, feedback-driven software processes. From an engineering standpoint, stability and predictability are critical properties, particularly for production pipelines.

Generalization results further reinforce these differences: Gemini preserves performance on unseen documents, while GPT shows degradation on inferential and structural attributes, indicating higher sensitivity to dataset-specific prompt adaptations.

7.4 Practical Implications for Data Ingestion Pipelines (RQ4)

Compared to the human-centered baseline, the proposed LLM-driven ingestion process offers substantial gains in scalability and operational efficiency. Human effort shifts from manual data entry toward validation and supervision, enabling the pipeline to scale while maintaining acceptable data quality.

The ability to preserve performance under generalization is essential for real-world deployment, where document formats continuously evolve. Rather than replacing human expertise, the proposed approach reallocates responsibilities, improving efficiency while retaining reliability through monitoring and oversight mechanisms.

8. Threats to Validity

This section discusses potential threats to the validity of the study and the measures adopted to mitigate them, following standard empirical Software Engineering guidelines.

Internal validity concerns whether the observed effects can be attributed to the proposed ingestion process rather than to uncontrolled factors. To reduce this risk, both LLM backends were evaluated under the same stateless execution protocol, with identical document ordering, prompt initialization, refinement logic, and evaluation criteria. Once the experimental loop began, no human intervention occurred in the prompt refinement process, reducing the influence of manual bias during iterative execution.

External validity concerns the extent to which the findings may generalize beyond the evaluated setting. The study focuses on real estate pricing tables, a domain-specific class of semi-structured documents. Although the dataset is limited in size, it exhibits substantial heterogeneity in layout, formatting, terminology, and implicit business rules. The paper therefore does not claim statistical generalization across domains; rather, it aims to evaluate process behavior under realistic industrial conditions. Generalization was further assessed by applying the final refined prompts to a disjoint set of previously unseen documents.

Construct validity concerns whether the selected metrics adequately capture the intended phenomena. The study evaluates extraction quality through attribute-level error rates in order to preserve task-specific differences that would be obscured by aggregate accuracy measures. These quality indicators are complemented by operational metrics—token consumption, estimated cost, and processing effort—which are directly relevant to the practical viability of data ingestion pipelines. Because the focus of the paper is controlled process evaluation, the analysis emphasizes descriptive process-level indicators and convergence behavior rather than hypothesis-driven statistical inference.

Conclusion validity concerns the robustness of the study's inferences. The results are supported by repeated observations across multiple iterations, attribute types, and two distinct LLM backends, as well as by a separate generalization phase. Nevertheless, because the dataset is relatively small and backend implementations may evolve over time, the findings should be interpreted as evidence of process-level behavior under the evaluated conditions rather than as universal performance claims.

9. Conclusion and Future Work

This paper framed semi-structured data ingestion as a Software Engineering problem and evaluated an LLM-driven pipeline with stateless execution, ground-truth comparison, and autonomous prompt refinement. Results show that prompt evolution can induce measurable learning without model retraining, but behavior differs substantially across LLM backends: Gemini converged and generalized across attributes, whereas GPT remained unstable on inferential and structural tasks despite similar process conditions. Compared to the human-centered baseline, the approach reduces human effort by shifting work from manual entry to validation. Future work includes testing additional domains and backends, and adding adaptive human-in-the-loop triggers for exceptions.

References

AHMAD, A. *et al*. (2024) "Large Language Models in Software Engineering: A Focus on Integration Challenges and Opportunities", *In:* CEUR Workshop Proceedings, v. 3762. https://ceur-ws.org/Vol-3762/534.pdf.

ARORA, S. *et al*. (2023) "Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes", arXiv preprint. https://arxiv.org/abs/2304.09433.

KUMAR, R. *et al*. (2024) "Integrating Large Language Models into Data Pipelines for Automated Insight Generation", ResearchGate preprint. https://www.researchgate.net/publication/398154369.

RASNAYAKA, S. *et al*. (2024) "An Empirical Study on the Usage and Perceptions of Large Language Models in Software Engineering", arXiv preprint. https://arxiv.org/abs/2401.16186.

SHARMA, P. *et al*. (2024) "From Raw Data to Actionable Insights: Leveraging Large Language Models for Automation", *International Journal of Recent Innovations in Computer and Communication Engineering*. https://ijritcc.org/index.php/ijritcc/article/view/11244.

SILVA, R. *et al*. (2025) "Comparing Large Language Models in Business Rule-Following", *In:* Proceedings of the Ibero-American Conference on Software Engineering (CIbSE). Porto Alegre: Sociedade Brasileira de Computação. https://sol.sbc.org.br/index.php/cibse/article/view/35327.

TIWARI, A. *et al*. (2023) "An Empirical Study on Information Extraction Using Large Language Models", arXiv preprint. https://arxiv.org/abs/2305.14450.

WYRICH, M. *et al*. (2025) "Understanding the Role of Large Language Models in Software Engineering: Evidence from an Industry Survey", arXiv preprint. https://arxiv.org/abs/2512.21347.

ZHENG, Z. *et al*. (2023) "Large Language Models for Software Engineering: Survey and Open Problems", ResearchGate preprint. https://www.researchgate.net/publication/378732714.

ZHENG, Z. *et al*. (2023) "Towards an Understanding of Large Language Models in Software Engineering Tasks", arXiv preprint. https://arxiv.org/abs/2308.11396.

ZHOU, Y. *et al*. (2025) "A Comprehensive Survey on Integrating Large Language Models with Knowledge Systems", arXiv preprint. https://arxiv.org/abs/2501.13947.