Provenance-Constrained Multi-Stage LLM Extraction from Heterogeneous Product Catalogs: System Design and Benchmark Protocol

Pablo Cardozo
Independent researcher · Buenos Aires, Argentina

Abstract

Schema-valid JSON says nothing about whether a product came from the right row. A catalog parser can omit an item, merge two records, or attach a plausible price to the wrong source while satisfying every field type. The inspected pipeline separates document metadata, source-row reconstruction, and product normalization. Its final stage works in ten-row batches and must return the expected batch identity, row set, and page coverage before any product is persisted. Deterministic gates reject invalid provenance and defer ambiguous or non-normalizable output for review. At the recorded public revision, the complete local suite passed 569 tests across 20 files in 1.06 seconds. Those tests support the encoded validators and application behavior. They do not measure extraction accuracy. A separate performance file uses an in-memory array and provides no basis for claims about Convex, provider calls, or end-to-end throughput. The inspected artifacts contain no labelled corpus, single-pass baseline, raw prediction set, or accuracy result. What remains is a paired benchmark against a pinned single-pass condition on the same documents, reporting row recovery, field accuracy, source attribution, review burden, latency, and cost.

Keywords: document extraction; structured generation; provenance; line items; human review; evaluation protocol

STATUS	Architecture and 569 software tests reproduced; extraction benchmark still unexecuted · Partial local verification: 2026-08-04

LOCAL SUITE	569/569 tests · 20 files
DURATION	1.06 s under Node 24.14.0
PIPELINE	3 stages · 10-row batches
ACCURACY	Not measured on a labelled corpus

Contents

1. Valid output, uncertain origin

A supplier catalog is more than a set of strings. Its rows encode a product, brand, presentation, sale format, unit, price, and sometimes a discount or wholesale condition. Meaning is distributed across page headers, merged cells, visual groups, footnotes, and abbreviations. Spreadsheet files expose cells but may still contain decorative rows and multi-line records; PDFs may preserve appearance while losing table semantics. The extraction task must recover both business fields and a defensible link to the source row.

The inspected Cuqui implementation decomposes the task into document metadata, row reconstruction, and product normalization [1–3]. The decomposition is technically plausible: early stages preserve layout and source identity; later stages work on bounded context and a strict schema. Plausibility is not comparative evidence. The research question is whether that extra structure reduces omissions, hallucinations, and review cost relative to a single-pass prompt when model family, source documents, and scoring are held constant.

- Specify the extraction task at row, field, and provenance levels.
- Audit the implemented stage boundaries, validation rules, retry behavior, and review gates.
- Reproduce the repository's deterministic software-test result at a pinned revision.
- Design a controlled comparison against a single-pass baseline without claiming an unrun accuracy result.
- Define release artifacts sufficient for independent scoring and error analysis.

Claim boundary 569 passing tests establish properties encoded by those tests. They do not establish extraction accuracy on real catalogs.

2. Related work and task positioning

Document AI research has approached visually rich documents through OCR-dependent multimodal models and OCR-free image-to-sequence systems. LayoutLMv3 jointly pretrains text and image representations for document understanding tasks [8]. Donut removes the external OCR stage and directly maps document images to structured output [7]. Nougat focuses on converting scientific pages into markup while preserving semantics lost in PDF [10]. These systems are learned document models; the present system instead orchestrates a general-purpose multimodal model through explicit stages and business schemas.

DocILE is the closest benchmark reference because it separates key information localization/extraction from line-item recognition and provides annotated business documents, baselines, and layout variation [9]. Product catalogs differ from invoices in ontology and pricing conventions, but the methodological lesson transfers: line items must be identified before field-level extraction can be scored, and evaluation should include unseen layouts rather than random rows from the same template.

Approach	Input representation	Output	Comparison role
LayoutLMv3-like	OCR text + layout + image	Task-specific labels	Research context, not implemented baseline
Donut-like	Document image	Structured sequence	Research context, not implemented baseline
Single-pass LLM	Whole uploaded catalog	Product JSON	Primary controlled baseline
Three-stage LLM	File, then rows, then batches	Provenance-linked product JSON	System under study

Table 1. Position of the proposed benchmark.

3. Task definition and unit of analysis

Let a document D contain an ordered set of source rows R. A gold annotation maps each valid product row r to zero or one normalized product y, with fields such as canonical name, brand, category, packaging, price type, amount, unit, and review status. Decorative headers, totals, and terms are labelled non-product. A prediction must first

identify a source row, then assign fields. This avoids rewarding a system that produces plausible products without traceable origin.

Level	Question	Example failure
Row	Were all and only product rows recovered?	One multi-line item split into two products
Field	Are values normalized correctly?	12 × 500 g interpreted as 500 g total
Provenance	Does each product point to its real source row/page?	Correct product attached to another row ID
Decision	Was ambiguity routed to review?	Low-confidence price published automatically

Table 2. Three levels of correctness.

The benchmark unit is a gold source row, nested within a document. Row-level metrics should be macro-averaged by document as well as micro-averaged across rows so that one large spreadsheet does not dominate the result. Field accuracy is computed only after predicted and gold rows are matched by provenance. Unmatched predictions count as hallucinated rows; unmatched gold rows count as omissions.

4. Implemented three-stage pipeline

The pipeline uploads a validated file to the Gemini Files API, waits until the file is active, and then executes two document-wide stages before batch normalization [2]. Stage 1 and Stage 2 use gemini-3.1-pro; Stage 3 uses gemini-3.1-flash-lite-preview at the recorded revision. Every generation requests application/json and supplies a JSON Schema. Each model call has a 60-second timeout and up to three attempts with increasing one-second backoff [2]. Model identifiers are version-sensitive and must be pinned again at experiment time.

Stage	Input	Output	Deterministic gate
1 — metadata	Full file	Pages, layout, table regions, document cues	Schema parse
2 — rows	Full file + Stage 1	Ordered page rows with rowId and raw text	Schema, totalRowCount, unique row IDs
3 — products	Ten rows + relevant page metadata	Normalized products and batch context	Schema, batch identity, row membership, page coverage
Post-process	Product extraction	Persistable product or error	Price, confidence, packaging, normalized unit, review rules

Table 3. Stage responsibilities and gates.

Stage 2 is the structural hinge. Its flattened row count must equal totalRowCount and every rowId must be unique. Stage 3 partitions the rows into groups of ten. A deterministic batchId includes the batch index plus first and last row IDs; the model must return the same ID, index, total batch count, ordered row IDs, and page numbers. Any item whose sourceRowId is outside the current batch rejects the batch result [2,3].

Listing 1. Simplified control flow.

```
metadata = stage1(file)
rows = stage2(file, metadata)
assert unique(rows.rowId)
assert flatten(rows).length == rows.totalRowCount
for batch in chunk(rows, 10):
    result = stage3(metadata, batch)
    assert result.context == expected_context(batch)
    assert every(result.item.sourceRowId in batch.rowIds)
    normalize_or_route_to_review(result.items)
```

5. Provenance constraints and human-review gates

Schema-valid JSON is necessary but insufficient. The implementation adds relational checks that the schema alone cannot express: unique row IDs, exact batch identity, ordered row membership, page coverage, and sourceRowId containment [2,3]. These checks prevent several silent failure classes, including a model answering from an earlier batch, returning a structurally valid but misaligned row, or inventing a product with no source member in the current context.

Trigger	Rule	Rationale
Low confidence	confidence < 0.5	Do not auto-publish uncertain extraction
Missing packaging	No packaging object	Presentation and normalized price may be unsafe
Failed normalization	Packaging exists but unit price cannot be calculated	Avoid incomparable prices
Generic name	Empty, unknown, or fewer than three characters	Prevent unusable catalog entries
Invalid amount	Price ≤ 0	Reject rather than review
Invalid source row	sourceRowId outside batch	Reject entire response

Table 4. Deterministic review triggers after model extraction.

The 0.5 confidence threshold is a policy constant, not a calibrated probability. Model self-confidence may be poorly calibrated and can drift across versions. The benchmark must measure selective accuracy: among items auto-approved at a threshold, how many are correct, and what fraction of all items is deferred? This makes the review gate an evaluated decision rule rather than a decorative score.

FORMULA	selective_accuracy(τ) = correct_autoapproved(τ) / autoapproved(τ)
Evaluated together with coverage(τ) = autoapproved(τ) / all_predicted_items. A high selective accuracy with near-zero coverage is not operationally useful.	

6. Resumability, duplicate control, and operational state

The ingestion action validates file magic bytes and size, computes SHA-256, and rejects recent duplicates for the same provider [2]. It stores file identity, metadata JSON, row JSON, progress, batch counters, errors, and results in an ingestion-run record. Stage 3 can resume from the first batch whose row IDs are not already present. Temporary upload files are deleted in a finally path. These controls address long-running job reliability but also create experimental variables: retries, resumed batches, and duplicate rejection must appear in result metadata.

Event	Current behavior	Required result field
File processing delay	Poll every 2 s, up to 120 attempts	poll_count, activation_ms
Model timeout	60 s per attempt	stage, attempt, timeout
Retry	Up to 3 with linear backoff	error class, attempt count
Batch failure	Index stored in failedBatches	failed batch IDs and reason
Resume	Skip batches whose source rows exist	resume point and prior run ID
Duplicate file	Provider + SHA-256 check	duplicate decision

Table 5. Operational events that the benchmark should record.

7. Reproduced software evidence

The public repository was cloned and detached at a5d3aadbfa049f4f6d77a2f144f557fa9bf7cffb. Under Node 24.14.0 and npm 11.9.0, npm ci completed and npm test exited zero. Vitest reported 20 passing test files and 569 passing tests in 1.06 seconds, with 693 ms attributed to test execution. This count supersedes the README statement of 409 tests for the pinned revision [1,5].

Listing 2. Reproduction command.

```
git checkout --detach a5d3aadbfa049f4f6d77a2f144f557fa9bf7cffb
npm ci
npm test
```

Observation	Supported conclusion	Unsupported conclusion
569 tests pass	Encoded validators and application behaviors pass locally	569 independent research examples exist
20 files pass	Test suite is broader than the README count	All production integrations were contacted
1.06 s duration	Local deterministic suite is fast	Catalog ingestion completes in 1.06 s
Exact revision pinned	Result is tied to source	Future model behavior is unchanged

Table 6. What the reproduced suite establishes and does not establish.

The separate performance file also passed five tests. Its timed operation generates 10,000 mock products and pushes identifiers into an in-memory array while mirroring object-spread overhead [6]. It does not call Convex, perform network I/O, invoke Gemini, validate an uploaded document, or persist products. The repository decision document's multi-million-products-per-second interpretation is therefore not used here [4]. The defensible result is only that the simulation and its assertions completed locally.

Evidence boundary A fast in-memory loop is not a database benchmark, and a passing software suite is not an extraction-accuracy study.

8. Controlled benchmark design

The proposed pilot uses a versioned corpus of at least 60 permission-cleared catalogs: 20 PDF, 20 XLS, and 20 XLSX. Sampling should stratify by layout complexity, row count, multi-line density, packaging ambiguity, mixed units, discounts, and image/scanned content. Documents from the same supplier template must remain in the same split to avoid layout leakage. A frozen development subset is used for prompt iteration; the final test subset is opened once.

Condition	Description	Controlled variables
B0 — deterministic spreadsheet	Cell/table parser for XLS/XLSX where structure is available	Same gold rows and field scorer
B1 — single pass	One multimodal prompt returns complete product JSON	Same model family, source file, ontology
S1 — three stage	Implemented metadata → rows → products pipeline	Same source, final schema, model family where possible
S1-no-review	Three stage without forced review policy	Measures contribution of decision gate

Table 7. Experimental conditions.

Each document-condition pair should run at least three times because generative extraction can vary. The primary comparison is paired by document. Model identifiers, prompts, schemas, temperatures, retry policy, and pricing must be archived. A failure after all retries remains a failed document; silently excluding it would bias accuracy and latency upward.

1. Create annotation guidelines and label rows before inspecting model output.
2. Double-label at least 20% of documents; adjudicate row boundaries and normalized fields.
3. Freeze splits, prompts, schemas, models, and scoring code.
4. Execute every condition with identical source documents and record all retries.
5. Publish redacted gold labels, raw predictions, match decisions, aggregates, and failure examples.

9. Metrics and statistical analysis

FORMULA	row_precision = matched_predictions / all_predictions
Penalizes invented or duplicate product rows.	

FORMULA	row_recall = matched_gold_rows / all_gold_product_rows
Penalizes omitted source products. Report F1 but retain precision and recall separately.	

FORMULA	field_accuracy_f = correct_f / matched_rows_with_gold_f
Compute per field after provenance matching; normalize case, whitespace, currency, and units with published rules.	

FORMULA	provenance_error = wrong_sourceRowId / all_predictions
A value can be textually correct and still fail if attached to the wrong row.	

Family	**Measures**
Extraction	Row precision/recall/F1, field exact and normalized accuracy
Provenance	Invalid row IDs, wrong row matches, wrong page coverage, duplicates
Review	Review rate, selective accuracy, errors auto-approved, correct items deferred
Reliability	Document success rate, retries, timeouts, failed/resumed batches
Operations	Latency per stage/document/row, tokens, API cost, file processing time

Table 8. Required outcome families.

Report document-level macro means with bootstrap 95% confidence intervals. For paired binary row outcomes, use a paired test such as McNemar where matching is valid; for document-level continuous measures, use paired bootstrap differences. Multiple field comparisons should be treated as a family and interpreted with effect sizes, not selected p-values. Error examples should be sampled by a predeclared taxonomy rather than chosen only for visual impact.

10. Error taxonomy, validity, and ethics

Code	**Error**	**Example**
E-R1	Omitted row	A product line disappears
E-R2	Invented row	A header becomes a product
E-R3	Split/merge	One multiline product becomes two or two rows become one
E-F1	Field transcription	Brand or price copied incorrectly
E-F2	Normalization	Pack multiplier or unit conversion wrong
E-P1	Provenance	Correct value assigned to another row
E-D1	Decision	Incorrect item auto-approved or correct item needlessly deferred

Table 9. Proposed error taxonomy.

- Construct validity: JSON validity and test count are not proxies for extraction correctness.
- Internal validity: stage models differ, so gains cannot automatically be attributed only to decomposition.
- External validity: Argentine food catalogs may not represent other sectors, languages, or document conventions.
- Temporal validity: preview model behavior and pricing can change after the recorded revision.
- Annotation validity: row boundaries and packaging normalization require documented adjudication.

Catalogs may contain supplier identities, negotiated prices, contact details, or commercial terms. Public benchmarking requires permission or irreversible redaction. Hashes, raw files, model uploads, logs, and released examples need a retention and access policy. Human annotators should see only the minimum necessary data, and the released corpus should avoid reconstructable confidential combinations.

11. Conclusion

The inspected system contains a thoughtful extraction architecture: explicit stage boundaries, schema-constrained generation, row and batch identities, deterministic provenance checks, forced review rules, retries, and resumable state. The complete local software suite is reproducible at 569 passing tests. Those are legitimate engineering results.

The central research claim remains open. There is no labelled corpus or controlled single-pass comparison, so improved extraction reliability has not been demonstrated. The protocol in this paper defines the missing experiment and prevents software-test volume or in-memory throughput from being substituted for model quality. Running and releasing that benchmark is the next evidence-producing step.

A. Minimum annotation record

Listing A1. Conceptual gold-record schema.

```
{
  document_id, page_number, source_row_id, raw_text,
  is_product_row, canonical_name, brand, category, subcategory,
  packaging: { type, units_per_pack, net_quantity, net_unit },
  price: { amount, currency, type },
  ambiguity_codes, annotator_id, adjudication_status
}
```

B. Benchmark release checklist

- Permission-cleared and versioned document corpus with template-aware splits.
- Annotation guide, double-label statistics, and adjudication log.
- Pinned prompts, schemas, models, dependencies, prices, and retry policy.
- Raw predictions for every run, including failures and retries.
- Open scorer with row matching, normalization, and provenance metrics.
- Document-level aggregates, confidence intervals, cost, latency, and error examples.

C. References and reproducibility

1. Cardozo, P. cuqui, revision a5d3aadbfa049f4f6d77a2f144f557fa9bf7cffb, 2026. Source ↗
2. Cardozo, P. convex/ingest.ts, revision a5d3aadbfa049, 2026. Source ↗
3. Cardozo, P. convex/lib/schemas.ts, revision a5d3aadbfa049, 2026. Source ↗
4. Cardozo, P. Implementation Decisions — Cuqui v1.0, 30 March 2026. Source ↗
5. Cardozo, P. Cuqui README and Tech Inventory, revision a5d3aadbfa049, 2026. Source ↗
6. Cardozo, P. tests/performance/batch-throughput.test.ts, revision a5d3aadbfa049, 2026. Source ↗
7. Kim, G. et al. OCR-free Document Understanding Transformer. arXiv:2111.15664, 2021. Source ↗
8. Huang, Y. et al. LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking. arXiv:2204.08387, 2022. Source ↗
9. Šimsa, Š. et al. DocILE Benchmark for Document Information Localization and Extraction. arXiv:2302.05658, 2023. Source ↗
10. Blecher, L. et al. Nougat: Neural Optical Understanding for Academic Documents. arXiv:2308.13418, 2023. Source ↗

Suggested citation

Cardozo, Pablo. "Provenance-Constrained Multi-Stage LLM Extraction from Heterogeneous Product Catalogs: System Design and Benchmark Protocol." Research protocol, v0.2, 2026.

Independent working document. Not peer reviewed. Claims are limited to the artifacts and source revision named above.