LongListBench: A Benchmark for Long-List Entity Extraction Under Layout and OCR Noise

Anton Fedoruk[1*] Serhii Shchoholiev[1†] Akhil Mehta[1‡]

[1]Kay.ai, Brooklyn, NY, USA

January 28, 2026

Abstract

Current document extraction benchmarks focus on short, key-value forms, leaving long-list entity extraction (recovering dozens to hundreds of repeated records from tables and mixed layouts) underexplored. Business documents such as loss runs, invoices, and itemized bills commonly contain such lists. We introduce LongListBench, a synthetic benchmark for long-list extraction that pairs structured ground truth (JSON) with rendered PDFs and OCR text, enabling reproducible end-to-end evaluation under layout and OCR noise. The benchmark injects seven document phenomena observed in production (page breaks, multi-row entities, duplicates, large documents, irrelevant tables, multi-column layouts, and merged cells) to stress segmentation and schema-conformant extraction at scale. Across 80 documents (6,828 incident rows), incident and reference numbers achieve 100% verbatim coverage in OCR, while zero-shot LLM baselines achieve 81.9% (Gemini 2.5) and 78.1% (GPT-5.2) field-level F1. Results highlight the table format and layout disruptions as primary failure modes, even when identifiers are reliably present.

1 Introduction

Long-list entity extraction (recovering dozens to hundreds of repeated records from semi-structured documents) is a core requirement for document automation in domains such as insurance, finance, and procurement. While recent advances in document understanding models (e.g., layout-aware pretraining [1] and OCR-free approaches [2]) and general-purpose LLMs have improved extraction quality, robust evaluation of long-list scenarios remains limited.

Many established benchmarks focus on key-value style extraction or relatively short, form-like documents (e.g., FUNSD [3]) or narrow document types such as receipts (SROIE [4]).

*anton@kay.ai

†serhii@kay.ai

‡akhil@kay.ai

More recent datasets such as DocILE [5] include business documents and line items, but long lists in the wild often exhibit additional failure modes: repeated entities, page breaks, multi-column reading order, irrelevant tables, and table constructs such as merged cells. VRDU [6] highlights that hierarchical and long-list fields remain challenging for LLM-based extraction.

We introduce LongListBench, a benchmark designed to stress-test long-list extraction from semi-structured business documents with long incident/line-item lists under systematically injected document phenomena and OCR noise. The benchmark is inspired by recurring patterns observed in real-world claims documents.

1.1 Background and Motivation

This work originates from production challenges encountered at Kay.ai and in prior industry experience. A client engagement required generating Statements of Values (SOVs) from loss-run PDFs containing insurance claims, often spanning hundreds of pages with varied formats and numerous layout artifacts. After evaluating off-the-shelf models and commercial extraction services, we identified long-list extraction as an underserved problem: existing tools performed adequately on short forms but degraded on documents with dozens to hundreds of repeated records. Similar challenges arose when processing itemized medical bills containing thousands of claim lines, exhibiting wide variation in table structures and OCR quality. These experiences motivated the development of a dedicated extraction pipeline (to be described in a subsequent paper) and, in turn, the need for a rigorous benchmark to measure progress on extraction methods that remain reliable as list length grows and as layout artifacts accumulate.

1.2 Research Questions

This work is organized around three practical questions:

- How do common long-list document phenomena (page breaks, duplicates, multi-row cells, multi-column layouts, irrelevant tables, merged cells) affect extraction quality?
- To what extent are end-to-end failures attributable to OCR versus downstream extraction?
- How do strong off-the-shelf LLMs perform under a simple, reproducible zero-shot protocol?

1.3 Contributions

We make the following contributions:

- A reproducible benchmark generation pipeline that produces paired ground truth JSON, rendered PDFs, and OCR text.
- A dataset of 80 documents (40 detailed, 40 table) containing 6,828 incident rows across four difficulty tiers, with an extreme tier reaching 500 incidents per document.

- A taxonomy of seven injected problem types and evaluation scripts for field-level scoring and OCR identifier coverage.
- Baseline results for GPT-5.2 and Gemini 2.5 under a shared prompt, highlighting remaining gaps in long-list extraction.

2 Related Work

Research on information extraction (IE) from visually rich documents has produced a broad ecosystem of datasets and models. However, much of the public evaluation landscape emphasizes either short documents (e.g., forms) or key-value extraction, leaving long-list entity extraction underexplored.

2.1 Document IE benchmarks

Early and widely used benchmarks such as FUNSD [3] focus on form understanding in noisy scans. Receipt datasets and challenges such as SROIE [4] emphasize OCR and key fields in narrow document types. These benchmarks are valuable, but typically contain relatively short documents and do not directly stress long lists of repeated entities.

DocILE [5] broadens the scope to business documents and includes line-item recognition, which is closer in spirit to long-list extraction. VRDU [6] further argues that hierarchical and repeated fields (e.g., invoice line items) remain difficult for LLM-based extraction. Our benchmark complements these efforts by focusing on list length, repeated entity boundaries, and a targeted taxonomy of long-list failure modes.

2.2 Document understanding models

Layout-aware pretraining approaches such as LayoutLM [1] jointly model textual content and 2D document structure, yielding strong performance on a range of document understanding tasks. In parallel, OCR-free approaches such as Donut [2] avoid explicit OCR by directly generating structured outputs from document images, mitigating OCR error propagation at the cost of specialized training.

In contrast, our work is model-agnostic: we provide paired PDF, OCR text, and ground truth, enabling evaluation of OCR-based pipelines, OCR-free models, and LLM-based extraction. Our primary goal is to support reproducible measurement of long-list extraction robustness under realistic layout artifacts.

3 Benchmark Construction

We construct LongListBench[1], a synthetic benchmark for long-list entity extraction in semi-structured business documents with long incident/line-item lists, inspired by recurring patterns observed in real-world claims documents. Each benchmark instance consists of (i)

[1]Code, data, and implementation details are available at https://github.com/kaydotai/longlistbench.

structured ground truth incidents (JSON), (ii) a rendered PDF, and (iii) OCR text of the PDF in Markdown.

3.1 Entity schema

Ground truth incidents follow a structured schema (see Appendix A) that includes incident identifiers, policy metadata, narrative text, and nested financial breakdowns (`bi`, `pd`, `lae`, `ded`). While downstream workflows often emphasize fields such as `incident_number`, `company_name`, `date_of_loss`, `status`, `driver_name`, `coverage_type`, and `total_incurred`, our evaluator requires and scores the full schema.

3.2 Document generation

Figure 1 illustrates the benchmark construction pipeline, which consists of two phases.

Schema design (green path). Human annotators reviewed real insurance loss run documents from trucking companies to identify common fields, layouts, and formatting patterns. From this analysis, we defined a structured JSON schema capturing incident identifiers, policy metadata, narrative descriptions, and nested financial breakdowns. This schema serves as ground truth for evaluation.

Synthetic generation (purple path). Rather than using real documents (which contain sensitive information), we generate synthetic documents that preserve realistic layout challenges:

1. **Problem configuration**: We identify recurring layout artifacts in real documents (e.g., page breaks splitting records, merged cells, and multi-column layouts) and encode them as configurable problem types.
2. **Document rendering**: We generate synthetic incident records under the target schema, render them to HTML with selected problem types injected, and convert the resulting HTML to PDF via headless Chromium.
3. **OCR**: We obtain OCR text for each PDF using Gemini 2.5 Flash on page images.
4. **Markdown output**: OCR text is stored as structured Markdown that preserves table structure and approximate spatial layout, consistent with typical OCR pipeline outputs.

Documents are rendered in one of two formats: (i) **Detailed**, which uses repeated incident blocks with narrative text and financial tables, or (ii) **Table**, which uses a compact tabular representation. All generation uses seeded randomness for reproducibility.

3.3 Injected problem types

We inject seven recurring document phenomena that complicate long-list extraction. These effects are applied at the HTML level prior to PDF rendering.

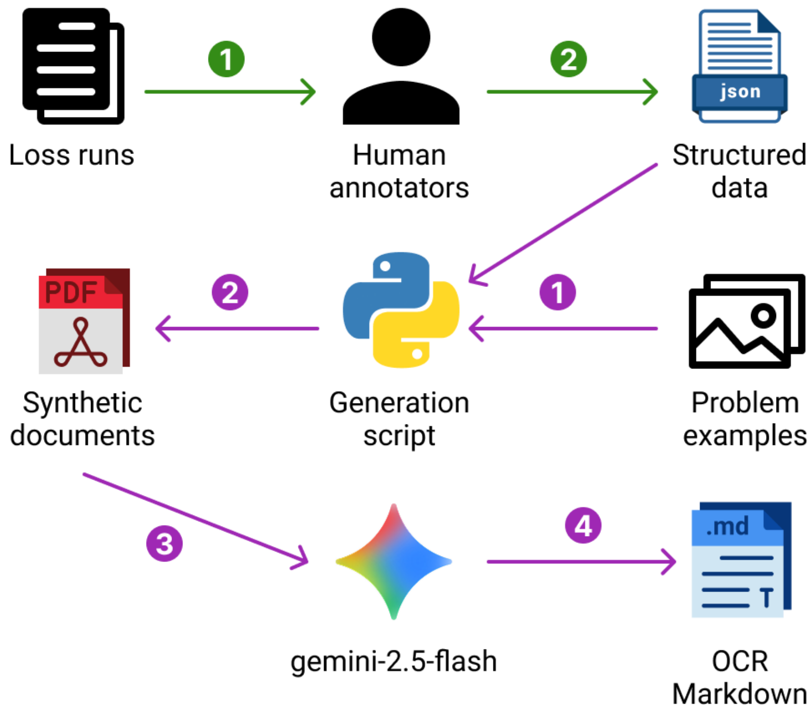
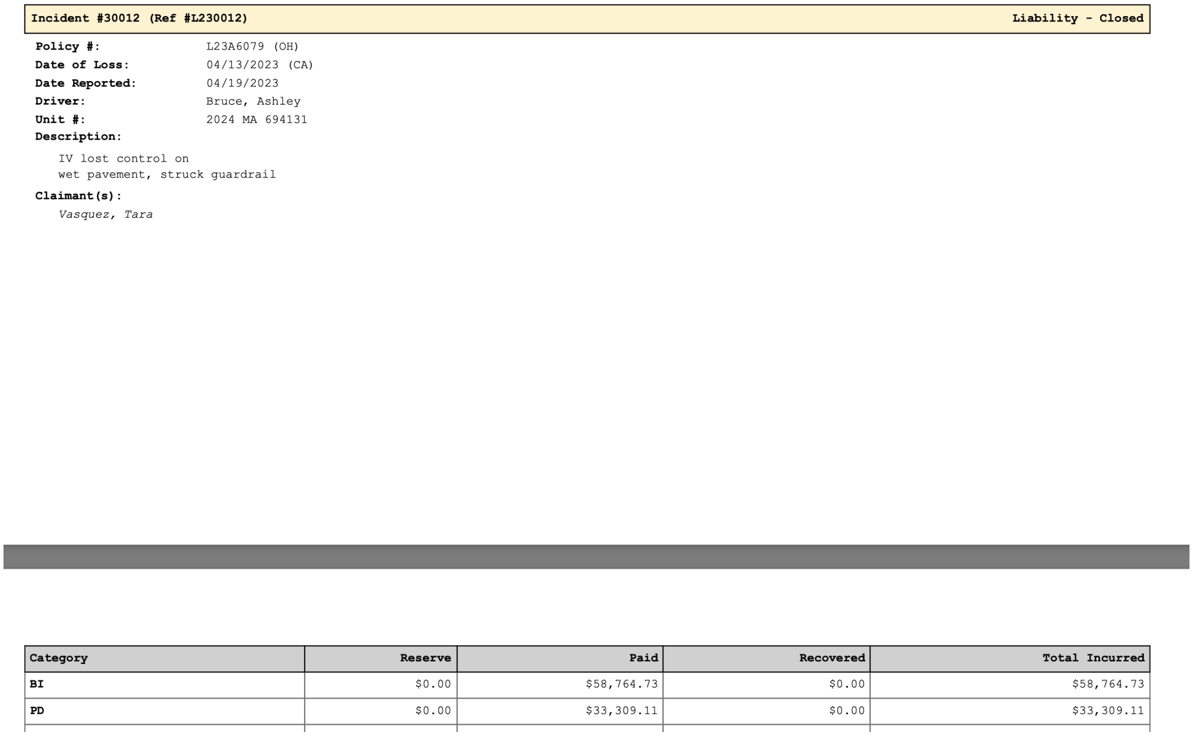


Figure 1: Benchmark construction pipeline. Green path: schema design from real documents. Purple path: synthetic document generation and OCR processing.

Page breaks. In real-world documents, page boundaries frequently split logical entities mid-record. A single incident may begin on one page with identifiers and description, while its financial breakdown appears on the next. This is particularly common in loss runs and itemized bills where dense formatting leaves no natural breakpoints. OCR systems typically process pages independently, producing separate text blocks that must be reassembled. Extraction models must recognize continuation patterns and avoid treating the second half of a split record as a new entity or dropping it entirely.

Multi-row entities. Table cells often contain text that wraps across multiple lines, especially for description fields, addresses, or claimant lists. When OCR linearizes such content, it may interleave text from adjacent columns or treat each line as a separate cell. For example, a description spanning three lines might appear as three distinct rows in the OCR output, with column alignment lost. Extraction models must recognize that these lines belong to a single logical cell and reconstruct the original cell boundaries from visual or positional cues.

Exact duplicates.
Production documents sometimes contain intentionally repeated records. In insurance contexts, the same incident may appear multiple times due to amendments, re-openings, or reporting across multiple policy periods. Unlike data entry errors, these duplicates are se-



Incident #30012 (Ref #L230012) **Liability - Closed**

Policy #: L23A6079 (OH)
Date of Loss: 04/13/2023 (CA)
Date Reported: 04/19/2023
Driver: Bruce, Ashley
Unit #: 2024 MA 694131
Description:
IV lost control on
wet pavement, struck guardrail
Claimant(s):
Vasquez, Tara

Category	Reserve	Paid	Recovered	Total Incurred
BI	$0.00	$58,764.73	$0.00	$58,764.73
PD	$0.00	$33,309.11	$0.00	$33,309.11

Figure 2: Example of an incident split across a page boundary.

mantically meaningful and must be preserved in the extracted output. Many extraction pipelines include deduplication as a post-processing step, which would incorrectly collapse valid duplicate entries. Models must faithfully reproduce the document content without applying implicit deduplication logic.

Large documents. Real loss runs and itemized bills routinely contain hundreds of line items. A single trucking company's annual loss run may list 200–500 incidents; a hospital bill for a complex procedure can exceed 1,000 charge lines. These documents push against context window limits of current LLMs. Even models with 128K+ token windows may exhibit degraded recall on items appearing in the middle of very long documents (the "lost in the middle" phenomenon). We include an extreme tier with 500 incidents per document to measure how extraction quality scales with document length.

Multiple tables. Business documents frequently embed auxiliary tables alongside the primary data. A loss run PDF might include a cover page with agent contact information, a summary table of totals by coverage type, or a glossary of status codes; none of these should be extracted as incident records. Models must distinguish the target entity table from these distractors based on schema matching, header recognition, or positional cues. Naive approaches that extract all tabular content will produce false positives from irrelevant tables.

Great West Casualty **Loss Run - Detailed Format**

9879 Joseph Square, West Deborahton, WY 98950
Phone: +1-891-832-7054x259

Account: Liberty Express LLC, Acct. # A0202155
Policy Period: 07/01/2023 - 07/01/2024 (Trucking)
Run Date/Time: 07/15/2024, 12:08 PM

Incident #	Reference #	Company	Coverage	Status	Policy #	Loss Date	State	Driver	Description	Reserve	Paid	Incurred
#30001	L230001	Liberty Express LLC	Liability	Open	L23A9836	03/17/2023	OH	Glover, Michelle	IV lost control on wet pavement, struck concrete barrier	$26,339.60	$0.00	$26,339.60
#30002	L230002	Atlas Logistics Group	Liability	Closed	L23A3581	08/09/2023	MI	Lewis, Richard	Tire blowout caused IV to veer into OV	$0.00	$38,116.39	$30,891.49
#30003	L240003	White Hauling Co	Cargo	Closed	L24A1824	01/20/2024	CA	–	Temperature-controlled cargo spoiled	$0.00	$22,461.42	$22,461.42
#30004	L230004	East Jody Transport	Liability	Open	L23A2873	04/09/2023	CA	Taylor, Cynthia	IV backing into loading dock, hit parked OV	$9,704.05	$0.00	$9,704.05

Figure 3: Example of a cell containing multiple lines of text.

Multi-column layout. Some documents use multi-column layouts to fit more content per page. This creates reading-order ambiguity: should text be read left-to-right across columns or top-to-bottom within each column? Standard OCR often assumes a single-column flow, interleaving content from parallel columns into an incoherent sequence. For example, incident #1's description might be concatenated with incident #10's financial data if both appear at the same vertical position in adjacent columns. Correct extraction requires column detection and proper reading-order reconstruction.

National Interstate **Loss Run - Detailed Format**

6339 Fowler Lodge, Jamesville, NM 23161
Phone: 275-804-2068x855

Account: Richardborough Logistics, Acct. # A0281280
Policy Period: 07/01/2023 - 07/01/2024 (Trucking)
Run Date/Time: 07/17/2024, 09:44 AM

Company Directory (Reference Only)

Employee	Department	Email	Phone
Adam Gonzalez	Sports administrator	stevenssarah@example.net	001-910-576-7816x9737
Lisa Cameron	Glass blower/designe	nguyencharles@example.net	(546)537-9547x195

Employee	Department	Email	Phone
Jennifer Franklin	Acupuncturist	arichardson@example.org	8384303616
Nicholas Clark	Best boy	connie58@example.org	912.254.2118x8117
Janice Oneal	Economist	mayerrebecca@example.org	4563525045
Stephen Barker	Counselling psycholo	thomasknapp@example.net	(845)901-7960x905
Loretta Ramirez	Manufacturing engine	danielwilliams@example.com	389-985-0673
Joseph Contreras	Recycling officer	jenniferjacobs@example.com	680-201-6566x5502
Justin Collier	Hydrographic surveyo	floresjackie@example.org	+1-398-832-7996x519
Susan Adams	Scientist, physiolog	jacobsmith@example.net	(898)939-9873x348

Figure 4: Example of a two-column document layout.

Merged cells. Tables often use merged cells for visual grouping. A common pattern is merging the company name cell across all incidents belonging to that company, or merging a coverage type header across its associated rows. In the rendered PDF, these appear as single cells spanning multiple rows or columns. OCR may report the merged cell's content only once, leaving subsequent rows with empty values for that field. Extraction models must propagate the merged value to all spanned rows, recognizing that an empty cell indicates inheritance from above rather than a missing value.

Incident #	Reference #	Company	Coverage	Status	Policy #	Loss Date	State
#30001	L230001	Delta Express Group	Physical Damage	Closed	L23A9665	02/06/2023	NY
#30002	L240002	Franklinview Express		Closed	L24A1572	11/13/2024	GA
#30003	L240003	Liberty Cargo LLC	Inland Marine	Open	L24A4867	02/13/2024	MO
#30004	L230004	South Rogerton Haulin		Open	L23A5476	10/03/2023	AR

Figure 5: Example of a table with merged and spanning cells.

3.4 Dataset scale

The released dataset (`longlistbench-v1`, version 1.0.1) contains 80 PDFs (40 detailed, 40 table) with 6,828 incident rows. Difficulty tiers are configured as 15 easy instances (10 claims/PDF), 12 medium (25 claims/PDF), 8 hard (50 claims/PDF), and 5 extreme (100 claims/PDF nominal). In the extreme tier, enabling `large_doc` expands each document to 500 incidents. Enabling `duplicates` injects additional duplicate rows (up to 5 per document), causing the number of incident rows to exceed the nominal tier size.

Across the 80 documents, the most common injected issues are multi-row entities (62/80), page breaks (56/80), duplicates (56/80), and multiple tables (40/80).

4 Evaluation

We evaluate systems on the task of extracting a list of incident records from the OCR text of each PDF. The benchmark provides both the OCR text and a structured JSON ground truth for each document.

4.1 OCR

All PDFs are converted to Markdown using a Gemini vision model. Each page is rendered to an image and converted to text with a system prompt that emphasizes preserving layout, spacing, and tables. In particular, tables are emitted in a CSV-like form inside Markdown, and the output is concatenated across pages.

4.2 LLM extraction protocol

We provide a lightweight, zero-shot evaluation harness that applies the same extraction prompt to multiple LLM providers and requires the model to return a JSON list of incident objects conforming to the full incident schema. The prompt includes a JSON Schema serialization of the target Pydantic model and is executed at temperature 0. Where supported, we request native structured outputs (e.g., response schemas) to reduce formatting errors.

To ensure schema conformance, model outputs are validated and normalized against a Pydantic schema before scoring (see Appendix A for full schema definitions and scoring rules). Predictions and aggregate reports are stored as machine-readable JSON (with an additional Markdown summary for convenience).

4.3 Chunking and merging for long documents

Hard and extreme documents can contain hundreds of incidents, and the OCR text may exceed practical context limits. The evaluation harness therefore supports chunked extraction: the OCR text is split into overlapping chunks using simple incident-number markers, targeting at most eight incidents per chunk. Each chunk is extracted independently, and chunk-level predictions are merged by normalized incident identifier, preferring non-empty fields and combining nested financial breakdown subfields.

4.4 Report regeneration and validation

To support reproducible analysis, the harness can regenerate summary reports offline from saved prediction files and optionally reuse extraction-time values from a previous report. A companion checker recomputes metrics from the saved predictions and the golden data, and flags schema violations or report inconsistencies.

4.5 Field-level matching and metrics

For scoring we use the incident number as the record identifier. Incident numbers are normalized by stripping common prefixes (e.g., `#`, `Incident #`). Let G be the list of ground-truth records and P be the list of predicted records. We compute micro precision/recall/F1 over field-value pairs, after canonicalizing each incident under the schema.

Canonicalization strips whitespace from strings, maps empty optional strings to null, sorts claimant lists, and rounds monetary values in nested financial breakdowns to two decimal places. Metrics are computed per document and then averaged across documents for tier- and format-level summaries.

For each incident, we flatten its fields into a multiset of canonicalized triples (`incident_id`, `field_path`, `value`) (including nested financial breakdown fields). We then define:

$$\text{found} = |\mathcal{F}(G) \cap \mathcal{F}(P)|, \tag{1}$$

$$\text{recall} = \frac{\text{found}}{|\mathcal{F}(G)|}, \tag{2}$$

$$\text{precision} = \frac{\text{found}}{|\mathcal{F}(P)|}, \tag{3}$$

$$\text{F1} = \frac{2\,\text{precision}\,\text{recall}}{\text{precision} + \text{recall}}, \tag{4}$$

where $\mathcal{F}(\cdot)$ denotes the multiset of flattened field-value pairs across incidents. We additionally report missing and extra incident identifiers and count exact record matches for incidents whose canonicalized objects match exactly.

Table 1: OCR identifier coverage on the full dataset (80 documents).

Identifier	Mean coverage	Min coverage
Incident number	100.0%	100.0%
Reference number	100.0%	100.0%

Table 2: Zero-shot LLM baseline results across the full benchmark (80 documents) under schema-conformant, field-level scoring (computed from released evaluation reports).

Model	Samples	Avg Recall	Avg Precision	Avg F1
Gemini 2.5	80	80.4%	83.4%	81.9%
GPT-5.2	80	76.8%	79.6%	78.1%

5 Results

We summarize results for (i) OCR fidelity and (ii) baseline extraction performance.

5.1 OCR identifier coverage

We measure how often key identifiers from the ground truth appear verbatim in the OCR text. Across the full dataset (80 OCR texts), incident numbers and reference numbers exhibit 100% coverage (mean and minimum). These results indicate that, for primary identifiers, our OCR step rarely drops information and that most downstream failures are attributable to extraction rather than OCR errors.

5.2 Zero-shot LLM extraction baseline

We evaluate two LLMs using the shared prompt and released evaluation harness. We report schema-conformant, field-level scoring across the full benchmark (80 documents: 40 detailed, 40 table) using the released per-tier evaluation reports. Averaged across all documents, Gemini 2.5 achieves 81.9% average F1 (80.4% recall, 83.4% precision), and GPT-5.2 achieves 78.1% average field-level F1 (76.8% recall, 79.6% precision) (Table 2). Across all models, the detailed format is substantially easier than the table format (Table 3), and performance varies meaningfully across difficulty tiers (Table 4).

Qualitatively, errors often manifest as local field-level deviations (e.g., missing optional strings, numeric drift in financial breakdowns, or small identifier formatting mistakes) spread across an otherwise correct long list.

These findings suggest that recovering identifiers is largely deterministic under our OCR pipeline, while the main open challenge for long-list extraction is robustly segmenting and populating full per-incident records under layout disruptions (page breaks, multi-column order, irrelevant tables, merged cells) and scale (hundreds of incidents).

Table 3: Baseline F1 by document format aggregated across all tiers (computed from released evaluation reports).

Model	Detailed F1	Table F1
Gemini 2.5	89.8%	73.9%
GPT-5.2	83.5%	72.8%

Table 4: Baseline F1 by difficulty tier (average across documents within each tier; computed from released evaluation reports).

Tier	Samples	Gemini 2.5 F1	GPT-5.2 F1
Easy	30	85.1%	80.2%
Medium	24	80.5%	76.7%
Hard	16	78.1%	76.0%
Extreme	10	81.6%	78.9%

6 Limitations and Future Directions

6.1 Limitations

LongListBench is designed as a measurement tool for long-list extraction under controlled layout and OCR noise. The current release prioritizes reproducibility and targeted stressors over exhaustive coverage of document variability. Table 5 summarizes what LongListBench v1 covers and what it does not.

In addition, our primary record identifier is the incident number. This choice simplifies evaluation and enables stable alignment at scale, but it can understate performance when a model extracts most fields correctly while corrupting identifiers. It also complicates the interpretation of results on documents containing exact duplicate incidents.

6.2 Future directions

We view LongListBench as an extensible benchmark and evaluation harness. The most valuable extensions are those that broaden the noise distribution, provide stronger signals for layout reconstruction, and benchmark extraction protocols that remain robust at hundreds of records.

Broader OCR conditions and layout supervision. An immediate next step is to expand OCR conditions beyond a single VLM-based OCR output. This includes scanned variants (blur, noise, skew, and resolution changes), classical OCR baselines, and prompt variations. For a small subset, releasing page-level supervision (e.g., table cell boxes or reading-order annotations) would enable controlled evaluation of layout-aware reconstruction methods.

Table 5: Scope of LongListBench v1.

Covered in v1	Not covered / out of scope
Claims-style long lists (loss runs) in two renderings (detailed and table)	Other long-list families (invoices, purchase orders, medical billing, financial statements) and non-English documents
Programmatic layouts with seven injected phenomena	Scan artifacts (skew, blur, stamps, handwriting), complex typography, and highly idiosyncratic templates
VLM-based OCR text with strong identifier retention	Broader OCR stacks and prompt variants; OCR bounding boxes and reading-order supervision
Schema-conformant, field-level micro scoring with canonicalization	Semantic equivalence, downstream task-based metrics, and duplicate-aware entity matching

Protocol benchmarks for long contexts. The extreme tier (500 incidents) is intended to pressure extraction protocols, not only base model quality. A natural extension is to benchmark chunking and merge strategies, retrieval-augmented extraction, and layout-aware segmentation under a unified interface, and to report cost and latency alongside accuracy.

Richer evaluation views. Field-level micro F1 is a stable aggregate, but it hides systematic error patterns. Future releases should include per-field breakdowns, record-level exact match rates, and duplicate-aware matching that treats repeated incidents as first-class entities rather than an edge case of identifier collisions.

Broader document families. Finally, expanding the benchmark to additional long-list domains and templates would test whether methods generalize beyond claims-style tables, and would make LongListBench a more comprehensive measurement suite for long-list extraction.

7 Conclusion

LongListBench targets a persistent gap in document understanding evaluation: extracting long lists of repeated entities from semi-structured business documents under realistic layout and OCR noise. We presented a benchmark construction pipeline that produces paired (PDF, OCR, JSON) artifacts and systematically injects common long-list failure modes.

7.1 Summary

Our main contributions are:

- A reproducible benchmark generation pipeline for semi-structured documents with long incident lists spanning two formats and four difficulty tiers.

- A taxonomy of seven problem types that frequently break long-list extraction systems, including duplicates, page breaks, multi-row entities, multi-column layout, and merged cells.
- An evaluation harness and baseline results that quantify the gap between near-perfect OCR identifier retention and imperfect end-to-end extraction.

7.2 Practical takeaways

We intend LongListBench to be useful as a measurement tool for both research and engineering workflows. Two practical takeaways are worth emphasizing. First, identifier retention in OCR is near-perfect (Table 1), so most end-to-end failures should be attributed to downstream parsing, segmentation, and field population rather than OCR errors. Second, even with schema-conformant structured outputs and a shared prompt, field-level extraction across the full benchmark remains materially below perfect (Table 2), with a large gap between detailed and table formats (Table 3) and meaningful variation across difficulty tiers (Table 4), indicating substantial headroom for methods that explicitly model reading order, table structure, and long-range consistency.

7.3 Recommended reporting

For comparability across papers and systems, we recommend that LongListBench results report (i) OCR identifier coverage, (ii) schema-conformant field-level precision/recall/F1 under the released evaluator, and (iii) the extraction protocol used for long documents (e.g., full-context vs chunking, chunk sizes, and merge strategy). The extreme tier, in particular, is intended to stress scaling behavior: methods that succeed on short lists may fail due to context limits, brittle segmentation, or accumulated small errors across hundreds of records.

7.4 Future Work

We view the benchmark as a foundation for studying scalable, layout-robust extraction. Immediate next steps include improved handling of duplicates and merged cells, and evaluation of methods that can reliably extract hundreds of incidents in a single document (Section 6).

References

[1] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, *Layoutlm: Pre-training of text and layout for document image understanding*, 2020. DOI: 10.1145/3394486.3403172. arXiv: 1912.13318 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1912.13318.

[2] G. Kim et al., *Ocr-free document understanding transformer*, 2022. arXiv: 2111.15664 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2111.15664.

[3] G. Jaume, H. K. Ekenel, and J.-P. Thiran, *Funsd: A dataset for form understanding in noisy scanned documents*, 2019. arXiv: 1905.13538 [cs.IR]. [Online]. Available: https://arxiv.org/abs/1905.13538.

[4] Z. Huang et al., *Icdar2019 competition on scanned receipt ocr and information extraction*, 2021. DOI: 10.1109/ICDAR.2019.00244. arXiv: 2103.10213 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2103.10213.

[5] Š. Šimsa et al., *Docile benchmark for document information localization and extraction*, 2023. arXiv: 2302.05658 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2302.05658.

[6] Z. Wang, Y. Zhou, W. Wei, C.-Y. Lee, and S. Tata, "Vrdu: A benchmark for visually-rich document understanding," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, arXiv:2211.15421, 2023. DOI: 10.1145/3580305.3599929. [Online]. Available: https://arxiv.org/abs/2211.15421.

Appendix A: Evaluation Schemas

A recurring challenge with existing document extraction benchmarks is incomplete or ambiguous schema documentation. Without clear specifications, it can be difficult to understand expected field formats, handling of optional values, or normalization rules. Researchers attempting to reproduce results must often reverse-engineer these details from examples or evaluation scripts, leading to inconsistent implementations and incomparable metrics.

To ensure full reproducibility, we provide complete schemas with explicit type annotations, default values, and field descriptions. The evaluation script validates model outputs against these schemas before scoring, ensuring that format errors are caught early rather than silently degrading metrics.

A.1 Financial Breakdown Schema

Each incident contains four financial breakdown objects (`bi`, `pd`, `lae`, `ded`) with the following fields:

Field	Type	Default	Description
`reserve`	float	0.0	Amount reserved for potential payout
`paid`	float	0.0	Amount already paid
`recovered`	float	0.0	Amount recovered (e.g., deductible)
`total_incurred`	float	0.0	Reserve + Paid - Recovered

A.2 Loss Run Incident Schema

The primary entity schema representing a single insurance claim incident:

Field	Type	Default	Description
`incident_number`	string	required	Incident number (e.g., #12345)
`reference_number`	string	required	Reference ID (e.g., L240123)
`company_name`	string	required	Trucking company name
division	string	General	Company division
`coverage_type`	string	required	Coverage type (Liability, Physical Damage, Inland Marine, Cargo)
status	string	required	Open or Closed
`policy_number`	string	required	Policy identifier
`policy_state`	string	required	Policy state abbreviation
`cause_code`	optional string	null	Internal cause code
description	string	required	Detailed incident description

Field	Type	Default	Description
handler	string	Claims Adjuster	Claims handler
`unit_number`	optional string	null	Vehicle/truck unit ID
`date_of_loss`	string	required	Date incident occurred
`loss_state`	string	required	State where loss occurred
`date_reported`	string	required	Date reported to insurance
agency	optional string	null	Insurance agency name
insured	string	required	Insured party name
claimants	list of strings	[]	List of claimants
`driver_name`	optional string	null	Driver name at time of incident
bi	financial breakdown	{}	Bodily Injury
pd	financial breakdown	{}	Property Damage
lae	financial breakdown	{}	Loss Adjustment Expense
ded	financial breakdown	{}	Deductible
`adjuster_notes`	optional string	null	Additional adjuster notes

A.3 Extraction Output Schema

Models are expected to return a JSON object matching the following structure:

Field	Type	Default	Description
incidents	list of LossRunIncident	required	List of extracted incident records

A.4 Field Scoring Rules

We compute schema-conformant, field-level precision/recall/F1 by canonicalizing records under the schema and comparing multisets of field-value pairs.

Step 1: Canonicalize each incident (schema validation + normalization). Both predictions and ground truth are parsed as full `LossRunIncident` objects and then normalized:

- **Incident identifier**: we compute a normalized incident ID by stripping common prefixes from `incident_number` (e.g., `Incident #`, `#`, `INC`) and trimming whitespace.
- **String fields**: all strings are trimmed. For optional string fields, the empty string is treated as `null`. Optional string fields are:
  - `cause_code`
  - `unit_number`
  - `agency`
  - `driver_name`
  - `adjuster_notes`
- **Claimants**: coerced to a list; each entry is trimmed; empty entries are dropped; the list is sorted.
- **Financial breakdowns**: for each breakdown object (`bi`, `pd`, `lae`, `ded`), we ensure it is an object and normalize each numeric field by converting to float, rounding to two decimals, and mapping negative zero to zero. Unparseable values are treated as 0.0. Breakdown numeric fields are:
  - `reserve`
  - `paid`
  - `recovered`
  - `total_incurred`

Step 2: Flatten incidents into a multiset of field-value pairs. For each canonicalized incident, we emit a list of triples (`incident\_id`, `field\_path`, `value`). Nested financial fields are represented with dotted paths (e.g., `bi.reserve`).

Step 3: Compare using multiset intersection (supports duplicates). Let $\mathcal{F}(G)$ be the multiset of flattened triples from the ground truth and $\mathcal{F}(P)$ the multiset from predictions. We compute

$$\text{found} = |\mathcal{F}(G) \cap \mathcal{F}(P)|, \tag{5}$$

$$\text{recall} = \frac{\text{found}}{|\mathcal{F}(G)|}, \tag{6}$$

$$\text{precision} = \frac{\text{found}}{|\mathcal{F}(P)|}, \tag{7}$$

$$\text{F1} = \frac{2\,\text{precision}\,\text{recall}}{\text{precision} + \text{recall}}. \tag{8}$$

The multiset formulation means that if a document contains exact duplicate incidents (same normalized `incident\_id`), they are counted with multiplicity, and the score only credits matches up to the minimum count in each multiset.

Additional diagnostics. We also report:

- **Missing/extra incident IDs**: computed from the set of normalized incident IDs present in each list.
- **Exact record matches**: the number of fully-matching incident objects, computed as a multiset intersection over canonicalized incident JSON objects grouped by incident ID.