ExtractBench: A Benchmark for Schema-Guided Enterprise Document Extraction

Boyang Zhang Adrian Lyjak Eli Stewart Zhaoqi Li Simon Suo

{boyang, adrian, eli, zhaoqi, simon}@runllama.ai

Abstract

Enterprise workflows increasingly rely on agents for *schema-guided extraction*: given a document and a user-defined schema, the agent faithfully follows the schema to produce the correct output with source evidence as grounding metadata. We present ExtractBench, a benchmark for schema-guided extraction, and, to our knowledge, conduct the first evaluation to report value accuracy, record completeness at scale, grounding, and measured cost together. The evaluation system contains 4,869 pages across 370 enterprise documents, 8 business domains, and 67 document types, with clear tags differentiating their challenge scenarios. The scalable schema and ground-truth curation pipeline combines independent-system agreement and adjudication for real documents, known values for synthetic lists, and human verification for forms. We report order-insensitive value F1 for value accuracy, plus two grounding metrics for source traceability: word- and page-level F1. Commercial VLMs perform well on short documents but often truncate record lists on long ones, while coding agents retain higher accuracy at much higher cost. LlamaExtract Agentic Plus ranks first on all three metrics, with accuracy comparable to coding agents at a fraction of the cost. Dataset and evaluation code are available on HuggingFace and GitHub.

Technical report · July 2026

Dataset Evaluation code

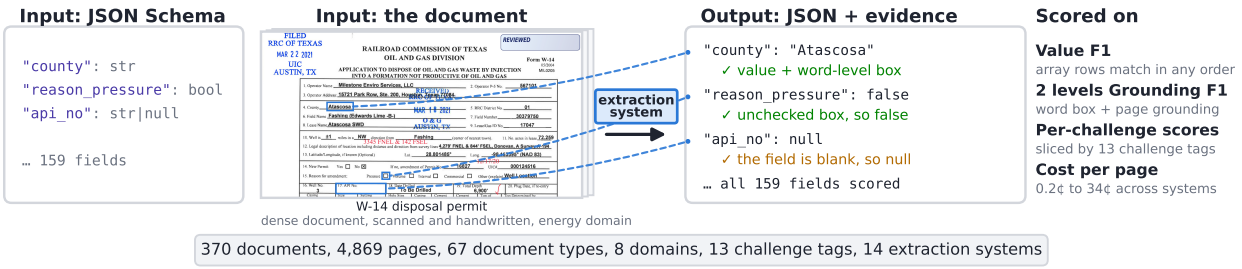


Figure 1: ExtractBench scores schema-guided extraction on real enterprise documents: given a document and its schema, a system returns schema-valid JSON with evidence, and is measured on value accuracy, grounding, per-challenge tags, and cost.

1 Introduction

Until recently, extracting structured data from business documents was performed by humans: knowledge workers read financial filings, insurance claims, purchase orders, and government forms, then keyed the relevant values into a system of record for downstream workflows. This work is usually highly manual and repetitive, and any mistakes can be costly [5, 18]. With the recent development of large language models and autonomous agents, we see fast-growing demand from enterprises to deploy agents to complete extraction-focused document workflows historically performed by humans.

Schema definition is at the center of the extraction workflow. A schema defines one extraction task, shared across all documents of the same type. For example, one invoice schema covers invoices from every vendor regardless of how different each invoice may look. Given that enterprises write a new schema for almost every new workflow, a system cannot be tuned to just one fixed template. We define the extraction task as *schema-guided extraction* (defined precisely in Section 2.1): given a document and a user-defined schema as input, the agent faithfully follows the schema to produce the correct output

	Benchmark	Corpus	Schemas	Domains	Real docs	Long records	Scans / handwriting	Grounding	Cost
FIXED KIE	SROIE [19]	1,000	fixed	1	✓		○	○	
	DocILE [39]	6,680	fixed	1	✓		○	✓	
	RealKIE [42]	1,867	fixed	5	✓	○	○		
SCHEMA GUIDED	Contextual EB [12]	35	5	5	✓	○			
	Extend LongArray [9]	45	3	3		✓			
	Micro1 LongExtract-50 [26]	50	per-doc	7	✓	✓			
	VAREX [4]	1,798	per-doc	1					
	ExtractBench (ours)	**370**	**67**	**8**	✓	✓	✓	✓	✓

✓ = covered and scored ○ = partial or incidental coverage; blank means absent.

Table 1: Comparison of representative fixed-ontology document-IE benchmarks (upper block) and modern schema-guided extraction benchmarks (lower block). ExtractBench is the only benchmark that jointly evaluates long-record completeness, real scans and handwriting, word- and page-level grounding, and measured cost. The full capability matrix appears in Table 7 (Section A.4).

along with source evidence as grounding metadata.

In real-world use cases faced by enterprises, there are many sources of challenges and failure cases in an extraction workflow, with common ones including missing rows in long lists, selecting the wrong occurrence of a sparse fact, overfilling dense forms, and confusing similar dates, identifiers, or amounts. There are also particular challenges in accurately understanding the structure of the document, which we call *perception challenges*: scan or handwriting noise, hierarchical headers, cross-page continuation, and large or irregular tables. Length creates a separate problem: a system can read local values correctly but still truncate a long schedule.

Visual grounding and traceability are another critical element for making agent-powered extraction effective and reliable at production scale for enterprises. Given that there are always inevitable failures — such as when an agent fails in reconciling fund holdings because a long schedule is truncated and rows are missing from the output — it requires a human in the loop to use visual grounding signals to quickly identify and correct the issues. Additionally, the cost of extraction per page also matters at production volume [23]. In high-volume, document-intensive enterprise workflows, a cost difference of one cent per page may determine if an AI initiative is financially viable.

Although there have been attempts from multiple existing benchmarks to tackle this challenge, they all have critical limitations (Table 1; Section 4 and Section A.4 give the detailed comparison). Classic information extraction filled fixed, hand-built templates with per-task systems [16]. Fixed KIE benchmarks, such as SROIE [19] and DocILE [39], do not handle user-specified schemas. More recent schema-guided benchmarks [4, 12] cover only a narrow dimension of the problem. The three closest benchmarks each cover one slice of these requirements: schema-conformant JSON against enterprise-scale schemas [12], row completeness on synthetic rendered arrays [9], and long statistical reports and filings [26]. None of the three measures cost, scores grounding, or contains a scanned or handwritten document. For example, Contextual AI's ExtractBench [12][1] does not cover any handwritten documents, nor does it take visual grounding or cost into consideration.

To bridge the gap, we introduce ExtractBench, a comprehensive benchmark for schema-guided enterprise document extraction that carries broad task coverage, evaluates traceability, and measures cost. ExtractBench contains 370 documents (4,869 pages) across 8 business domains and 67 document types. Each document type has one schema shared across its documents. Each document is tagged by task challenge, perception challenge, table structure, domain, and length (Section 2.2). The benchmark is composed of real born-digital documents, synthetic long lists based on real layouts, and real regulatory and tax forms with schemas authored from blank templates. To establish high-quality ground truth at scale, we design a scalable pipeline: independent-system proposals are adjudicated for real documents, values are set before rendering for synthetic lists, and humans verify both values and grounding on scanned forms (Section 2.3). We evaluate accuracy with order-insensitive value F1 over the values in the extracted JSON. To evaluate visual grounding ability, we also score whether a correct value points to its source for fields with human-verified boxes, so reviewers can audit the answer without searching the document (Section 2.4).

We evaluate 14 frontier methods spanning commercial VLMs, open-source extraction, coding agents, and specialized APIs. We noticed significant performance variance in out-of-the-box frontier models across different challenge dimensions. For example, Gemini 3.5 Flash ac-

[1] The unrelated academic benchmark by Contextual AI shares the ExtractBench name [12]; Section 4 details how the two differ in scope.

curacy dropped significantly from 87.9% on short documents to 27.9% on long ones (Section 3). LlamaExtract Agentic Plus shows much more consistent performance, with 96.6% on short and 94.4% on long documents. It also outperforms Codex GPT-5.5 (95.6% versus 93.6%) at a much lower cost (8.1 ¢/page versus 27.8 ¢/page). Additionally, commercial VLMs and coding agents do not return word-level boxes, so workflows that require source evidence need specialized extraction APIs (Section 3.4).

Our contributions include:

- **A challenge-tagged benchmark with broad coverage.** 370 documents and 4,869 pages span 8 business domains and 67 document types, with tags for task challenge, perception challenge, table structure, domain, and length that support per-challenge analysis.
- **A scalable pipeline for schema and ground-truth curation.** To produce high-quality, well-specified schema–ground-truth pairs without labeling every field by hand, we combine frontier-model ensembles for real documents, programmatic generation for synthetic long lists, and human verification for scanned forms.
- **A broad evaluation of frontier extraction methods.** We compare 14 systems spanning commercial VLMs, OSS pipelines, coding agents, and specialized APIs, reporting accuracy, grounding, cost, and the quality–cost tradeoff.

2 ExtractBench

This section defines schema-guided extraction precisely, then describes how ExtractBench applies it to build a challenge-tagged corpus.

2.1 Task Definition

Given a document and a schema, a system returns structured data with evidence (Figure 1):

$$f : (\text{document}, \text{schema}) \longmapsto (\text{structured data}, \text{evidence}).$$

Input. The input is a full document, born-digital or scanned, and a schema written by the user. The user specifies the extraction task via a schema: it lists the fields to extract, and each field has a name, a type, and a natural-language description of what belongs in it. It is expressed as a *JSON Schema*, the industry-standard way to specify structured output, and may contain scalar fields, nested objects, arrays of records, nullable fields, and value constraints. A schema defines one extraction task and guides all documents of the same type, even though the documents can vary significantly in structure, layout, and styling. For example, insurance claims from different providers vary greatly in length and layout. A document usually holds more than the schema asks for — and sometimes less: any field the document leaves unanswered must come back as null.

Output. The output is a schema-valid JSON object, with the source page and a bounding box for each value as *evidence*. It must return correct, exhaustive values (including repeated records), correctly use null for absent information, and ground each extracted value. Section 2.4 makes these expectations precise.

2.2 Taxonomy and Coverage

Existing benchmarks for document extraction often report one aggregate score over a narrow set of document types. An aggregate score does not show whether a system missed a third of a list or got one label wrong, and it cannot distinguish a hard extraction task from a bad scan. ExtractBench instead tags each document along five independent axes: task challenge (what makes extraction hard), perception challenge (how the page was captured), table structure, length, and business domain. Because the axes are independent, a low score can be traced to its actual cause. Figure 2 shows how the corpus distributes over the five axes. The paragraphs below briefly explain each axis; Table 4 in Section A.1 gives additional details and representative document types for every tag.

Task challenges. A *task challenge* defines the nature of the extraction task and what makes it difficult.

- **T1: long-list completeness.** Recover *every* record of a repeated structure that can span many pages. Typical failures are truncation, duplicated or merged rows, hallucinated records, and values attached to the wrong record.
- **T2: needle-in-haystack.** Find a small number of requested facts in a long document. T2 has few target records but many plausible mentions, only one of which is canonical; failures are missed targets, wrong occurrences, and unnormalized paraphrases. It is also the only task challenge that asks for far less than the document holds: a median of just 1.6 fields per page (Section A.1.1).
- **T3: dense documents.** Fill many fields from a document dense with labels, blanks, checkboxes, handwriting, and scan artifacts. The characteristic failure is over-extraction, inventing a value for a field that is actually blank, compounded by missed checkboxes and mislabeled fields. A dense document also repeats identifiers, dates, and amounts of the same format, so a

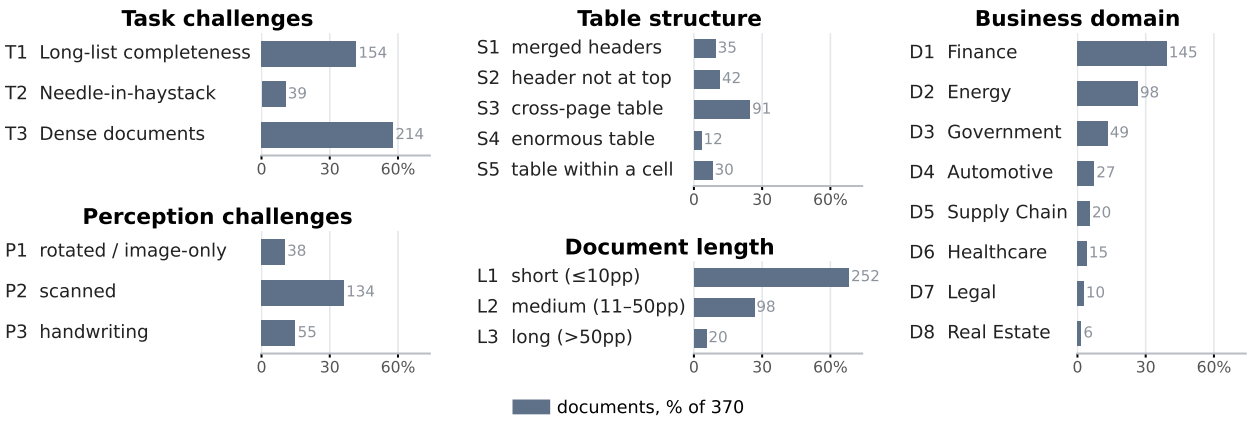


Figure 2: ExtractBench coverage across the five tag axes. Each bar is the share of the 370 documents carrying the tag, with its document count shown beside the bar. The task panel reports the three task challenges; a document tagged with several of one challenge's sub-tags counts once. Tags may overlap across panels; Table 4 (Section A.1) defines every tag and sub-tag.

plausible value can end up in the wrong field. Dense forms are the most common case (T3.a); receipts, invoices, and regulatory filings belong here too.

Perception challenges. A *perception challenge* records how the page was captured. The tags are rotated or image-only capture (P1), scanned page images (P2), and handwriting (P3) (Table 4). They are assigned independently of the task challenge, so the same extraction task can appear under more than one perception challenge.

Table structure. Tables earn a dedicated axis for two reasons. First, most of the values enterprises extract live in tables, from holdings schedules to invoice line items. Second, tables fail in a way no other page element does: a complex table can be read correctly value by value and still be assembled into the wrong structure, a failure that the task and perception axes cannot isolate. The structure tags mark the layouts where this happens: merged or hierarchical headers (S1), a header that does not sit above its data (S2), a table that continues across pages (S3), a table beyond a thousand rows (S4), and a table packed inside a single cell (S5). Each layout has its own failure: a merged header attaches values to the wrong column, a pivoted header transposes the record, a cross-page table loses its continuation, a very large table stops early, and a packed cell comes back as one string instead of its fields.

Document length. Documents fall into three length buckets: short (L1, up to 10 pages), medium (L2, 11 to 50), and long (L3, more than 50). Length gets its own axis for the same reason the other axes are separate: the same task challenge can appear at any length, and length adds a failure of its own, since a system can read every value on a page correctly and still stop before the end of a long schedule.

Business domains. An enterprise extraction system needs to work across domains, for two reasons: teams want one system rather than a separate tool per document type, and businesses often do not control what arrives and must process whatever their customers, vendors, and regulators send. ExtractBench therefore spans 8 domains and 67 document types (Table 4): finance and fund holdings (D1), energy-sector regulatory forms (D2), government procurement and customs (D3), auto valuation (D4), supply-chain and other transactional documents (D5), healthcare remittance (D6), legal and bankruptcy filings (D7), and real-estate closing disclosures (D8). Prior benchmarks for structured extraction usually cover fewer domains and a handful of real document types (Table 6).

2.3 Schema and Ground-Truth Construction

To properly evaluate a schema-guided extraction system, the extraction task itself needs to be well specified: each schema must be coherent with the documents it applies to, and every field must have a clear expected value. If a schema is poorly aligned with its document family, or its instructions leave the extraction goal ambiguous, errors can no longer be attributed and a low score may reflect a defect in the benchmark rather than in the system being tested.

Creating well-defined schemas and ground truth that corresponds to them is labor-intensive, particularly when

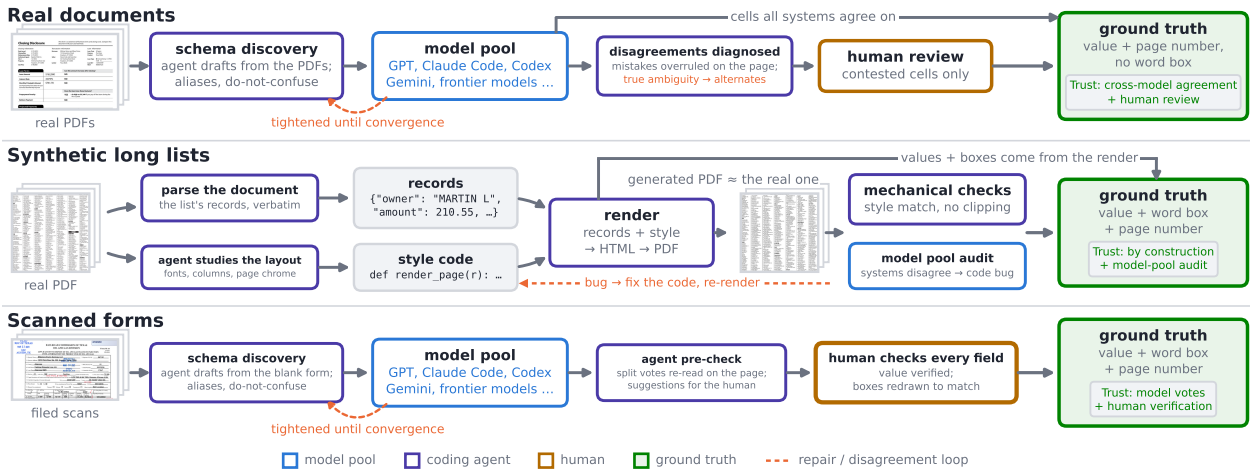


Figure 3: How ground truth is constructed, one strip per source type. A card's border marks who runs the step (blue: the extraction-model pool; violet: coding-agent pipeline code; amber: a human); the filled green card is the resulting ground truth, and orange dashes mark repair and disagreement loops.

documents are long and the data is dense. Checking every field by hand is prohibitive in time and cost at this scale, and using a single extractor's output as ground truth would repeat its mistakes and bias the results toward that extractor. This motivates a scalable pipeline that produces high-quality schema and ground-truth pairs without fully manual annotation.

To this end, we combine three sources of documents, each annotated by the method that fits it: frontier-model ensembles for real documents, programmatic generation for synthetic long lists, and human labelers for scanned forms (Figure 3). Real documents supply the layouts, scan noise, and domain range we want to test, but drawing a box on every one of their fields is prohibitively slow. Synthetic long lists cover documents too large to label by hand: thousands of similar records are slow to annotate and easy to misread. Scanned forms are real documents that need a person to decide each value, because handwriting is unclear and a mark can sit between two fields. Documents from these sources are also re-captured as degraded scans, which needs no new annotation: the values do not change, so the clean document's ground truth carries over. We use this methodology to build ExtractBench, but it scales readily to much larger datasets. Section A.3 gives the full procedures.

Schemas. A document type is a family of documents that carry the same kind of information — SEC 13F filings, utility bills, mortgage closing disclosures — however much their layouts differ. In ExtractBench, each document type has exactly one schema, shared by all of its documents. The schema takes the form a user writes in production: field names, types, and a natural-language description for each field (Section 2.1). How each schema is authored depends on its source and is described with each pipeline below. Every field is written to have a deterministic expected value in the document, so the ground truth is the same no matter which system is being scored.

Real documents. The schema is drafted from sample documents, then several extraction systems from different model and pipeline families run against this candidate schema (Figure 3, top strip): no single extractor is reliable enough, and same-family systems share mistakes. A value on which every system agrees, including null for absent fields, becomes candidate ground truth. Disagreements are classified by cause: if more than one reading of the field is defensible, the schema is at fault, and we tighten its description with aliases, format requirements, location hints, and do-not-confuse guidance until reruns converge; if only one reading is defensible, it is a model failure, and a reviewer settles the contested cells against the page.

Synthetic long lists. We build each synthetic document backwards, data first and document second, so the ground truth is born trusted and no human ever labels a value. From a real filing (a fund schedule, holdings register, or creditor matrix), we produce the records, parsed verbatim or generated in its style, and rendering code that a coding agent writes after studying the layout's fonts, columns, and page chrome (Figure 3, middle strip); the family keeps the real filing's schema. That code renders the records into a PDF closely matching the real one, with page breaks placed by measurement. Every value is known before the PDF exists, and its page and

word-level box are read back from the render, so the ground truth stays exact however long the list grows. Mechanical checks catch style mismatches and clipping, and an extraction-system pool audits the finished document, its disagreements exposing rendering-code bugs that are fixed before the family ships.

Scanned forms. Scanned forms are the one source where a person checks every field (Figure 3, bottom strip). The schema is authored against the blank form template and frozen before any document is labeled. An ensemble of up to five systems votes on every schema leaf; contested votes go to an adjudication agent that must inspect the page before ruling. A designated pipeline proposes a box per field, and a human annotator accepts, edits, nulls, or redraws each one. This yields 169 human-verified documents, with 84% of verified fields carrying a human-placed box; the rest are mostly blank fields, with nothing on the page to box.

The three pipelines back their ground truth differently: real-document values are confirmed by agreement across independent systems, synthetic values and boxes are exact by construction, and form values and boxes are checked by a person. This determines which metrics each document supports: values are scored everywhere, box-level grounding only where the boxes are verified (Section 2.4).

2.4 Metrics

ExtractBench measures two things. Value accuracy asks whether a system returned the right values, and is scored with the unified value F1 on every document. Grounding asks whether the system can show where each value came from, and is scored only on documents whose box ground truth is verified (Section 2.3).

Value accuracy. The unified value F1 scores whether the extracted *values* match the expected output, under one definition for scalar fields and arrays of records. Each output is flattened into cells, one per scalar field and per aligned record subfield, and a cell is correct when it matches its expected counterpart after normalization. Precision, recall, and F1 are computed over these cells per document, and slices report unweighted document means (Section B gives the exact scoring rules).

- **Array alignment.** A repeated structure is compared as an unordered set of records: records are paired by the Hungarian algorithm to minimize mismatched cells, following how prior extraction benchmarks align line items and arrays [39]. Unmatched expected records lower recall; extra predictions lower precision.
- **Normalization.** Values are normalized before comparison: dates to ISO format, strings by collapsing whitespace; everything else requires exact equality, with no numeric tolerance and no LLM judge. The few exceptions are listed in Section B.1.
- **Missing values.** An omitted key scores as an explicit null, so every scalar field counts toward both precision and recall, and a correct null on a blank field is credited (Table 9 lists every case). Only repeated records move precision and recall apart, so a gap between them points to dropped or extra records rather than wrong values (Section D.1 tabulates both per system and length slice).

Grounding. For fields with a verified ground-truth box, ExtractBench also reports word-level grounding precision, recall, and F1. A field counts as grounded only when its value is correct and its predicted box overlaps an accepted box for that field, at a fixed IoU threshold of 0.5: a well-placed box around a wrong value earns no credit. Page-level grounding F1 asks the weaker version of the same question, requiring only the correct source page rather than a box, which many systems satisfy even when they return no boxes at all. Section B.4 gives the details.

3 Experiments

3.1 Setup

We evaluate 14 extraction systems[2] across three high-level approaches:

- **VLMs** treat extraction as direct multimodal generation: they receive the document and schema and generate structured output in a single model call. We evaluate GPT-5.4 Nano [29] and Google Gemini 3.5 Flash [13], called through constrained structured-output APIs; and Lift 9B [7], NuExtract3 [27], Qwen3.6 35B-A3B [34], and Gemma4 26B [14], which are self-hosted.
- **Coding agents** extract through an iterative tool-use loop: they can inspect the document, write and run parsing code, validate results, and revise the final output. We evaluate Claude Code Opus 4.8 [2] and Codex GPT-5.5 [30], which receive the document and schema with filesystem and tool access (tool configuration in Section C.1).
- **Specialized APIs** provide a managed document workflow that handles preprocessing, parsing, and schema-guided extraction, sometimes with source grounding. We evaluate Reducto Deep Extract [36], Extend Max

[2]Models and prices reflect those available as of July 1, 2026.

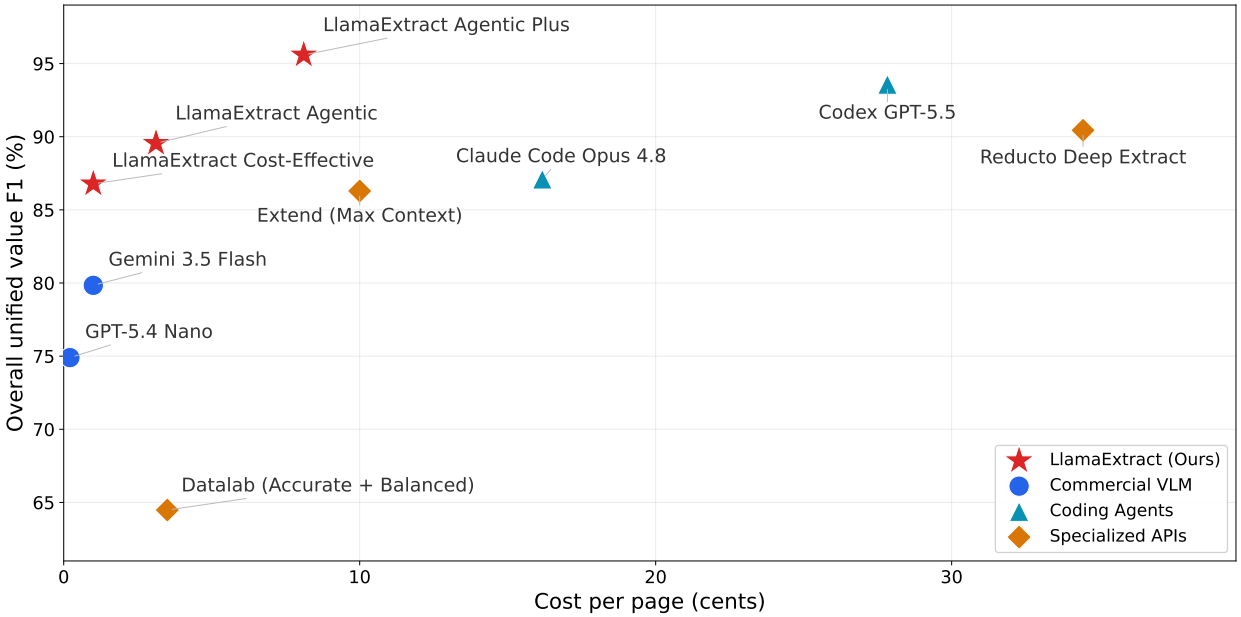


Figure 4: Overall unified value F1 versus mean document-level cost per page, pooled over every scored document. Marker shape and color follow the system grouping of Section 3.1. The four open-weight pipelines have no vendor API price and are omitted. Per-length results appear in Section D.1.

Context [10], Datalab Accurate Parse + Balanced Extract [6, 8], and three LlamaExtract tiers [24] (Cost-Effective, Agentic, and Agentic Plus).

All systems receive the same document–schema pairs and are evaluated without benchmark-specific tuning; all runs took place in June–July 2026. We report unweighted document-level means for value F1, word- and page-level grounding F1, and cost, using the metrics defined in Section 2.4. Per-page costs apply published provider rates to actual token or credit consumption (Section C.2). Because the four self-hosted VLMs have no directly comparable API price, we omit them from cost comparisons.

3.2 Quality–Cost Frontier

Enterprise extraction workloads often span millions of pages, making per-page cost differences substantial. At one million pages, each cent per page adds $10,000. Figure 4 compares overall value F1 with measured per-page cost.

The evaluated system families occupy distinct regions of this tradeoff. The VLMs evaluated here occupy the low-cost region of the quality–cost tradeoff ($\leq$1.0 ¢/page), but neither exceeds 80% F1. Coding agents reach 87–94% F1, but cost more than 15 ¢/page. Specialized APIs span a much wider range. Within this group, LlamaExtract traces the quality–cost frontier: Cost-Effective reaches 86.8% F1 at 1.0 ¢/page, Agentic reaches 89.5% at 3.1 ¢/page, and Agentic Plus reaches 95.6% at 8.1 ¢/page. Agentic Plus outperforms both coding agents while costing no more than half as much. These results show why extraction quality and cost must be evaluated jointly: greater spending does not necessarily produce greater accuracy. A broader comparison of commercial VLMs is provided in Section D.3.

3.3 Results Across Dimensions

Overall F1 makes it easy to compare systems, but a single aggregate score cannot show which document characteristics drive their successes and failures. To expose these failure modes, we use ExtractBench's challenge tags to break down performance across five axes: document length, task challenge, perception challenge, table structure, and business domain (Table 2).

Document length. Most systems perform well on short documents, with more than half scoring above 90%, but the spread widens as documents grow longer. On long documents the commercial VLMs fall below 40%, while Claude Code Opus 4.8 (88.1%) and Reducto Deep Extract (92.0%) remain close to their short-document scores. LlamaExtract Agentic Plus is the strongest across all three lengths and never drops below 90% (96.6/93.3/94.4). The long-document failure is concentrated in recall: entire records are dropped rather than misread (Section D.1 reports precision and recall separately). We attribute this to

	Specialized APIs						Coding Agents		OSS				Commercial VLM	
Dimension	***LE Agentic Plus***	***LE Agentic***	***LE Cost-Eff.***	***Datalab A+B***	***Extend Max***	***Reducto Deep***	***Codex GPT-5.5***	***CC Opus 4.8***	***Gemma4 26B***	***Qwen3.6 35B-A3B***	***NuExtract3***	***Lift 9B***	***GPT-5.4 Nano***	***Gemini 3.5 Flash***
Overall	**95.6**	89.5	86.8	64.5	86.3	90.4	93.6	87.1	66.2	87.3	47.9	77.3	74.9	79.8
Document Length														
L1 Short (≤10 pp)	**96.6**	92.0	90.8	62.8	92.0	94.2	95.7	90.1	80.5	93.1	54.4	87.2	77.4	87.9
L2 Medium (11–50 pp)	**93.3**	85.4	80.1	73.8	78.8	80.5	91.2	79.2	40.5	84.8	39.3	62.6	76.4	69.8
L3 Long (>50 pp)	**94.4**	78.6	69.2	40.5	51.3	92.0	78.9	88.1	12.2	26.8	8.9	25.3	35.8	27.9
Task Challenge														
T1 Long-list completeness	**96.1**	85.9	81.8	80.2	87.2	94.8	91.7	93.6	51.1	79.0	31.8	68.6	72.2	78.8
T2 Needle-in-haystack	**93.6**	88.3	82.3	73.9	90.3	92.5	91.7	89.1	63.0	85.3	25.2	78.0	74.0	87.9
T3 Dense documents	**95.5**	92.1	90.5	54.4	85.7	87.5	95.4	82.4	76.8	93.1	58.7	82.9	76.4	80.5
Perception Challenge														
P1 Rotated / image-only	**95.9**	88.2	85.0	78.9	89.0	93.9	81.0	91.2	66.5	86.8	28.9	80.7	64.7	88.6
P2 Scanned	**93.9**	89.8	87.8	47.6	80.9	81.1	93.4	74.2	69.1	92.0	62.6	76.0	67.4	71.1
P3 Handwriting	**93.8**	90.9	87.6	47.2	93.8	92.3	93.6	74.7	73.9	92.3	75.8	86.0	67.7	74.2
Table Structure														
S1 Merged headers	94.5	78.5	79.4	82.8	91.0	94.2	**95.0**	94.0	44.6	81.8	33.3	69.9	68.9	80.2
S2 Pivoted / header not at top	95.0	86.1	87.2	84.4	91.4	**95.3**	94.9	94.3	63.0	89.6	20.9	76.4	77.7	88.4
S3 Cross-page table	**95.8**	84.3	79.0	78.5	85.1	94.4	89.4	92.5	40.5	73.8	37.6	64.5	72.3	73.6
S4 Enormous table	**95.9**	78.1	67.8	32.7	24.8	95.3	78.9	87.8	0.0	1.3	3.7	1.2	7.2	1.5
S5 Table within a cell	**97.2**	87.3	78.2	71.7	75.9	95.4	86.8	93.9	37.1	56.6	50.1	51.3	67.3	53.4
Business Domain														
D1 Finance	**96.3**	91.3	87.2	62.1	79.2	85.1	96.2	84.7	59.5	85.9	48.4	71.8	77.5	75.4
D2 Energy	**95.0**	90.7	88.5	49.0	94.5	93.7	94.1	82.9	78.7	92.4	76.6	87.9	73.0	78.9
D3 Government	**93.4**	83.4	83.0	77.9	86.5	92.5	92.7	91.0	58.1	83.7	18.0	71.5	74.3	81.4
D4 Automotive	97.9	95.0	95.2	85.0	91.9	97.3	95.2	98.0	83.9	96.6	14.8	85.9	79.3	**98.0**
D5 Supply Chain	97.9	93.2	92.0	82.5	96.8	96.1	95.9	**99.0**	87.4	95.4	31.6	93.5	83.7	98.2
D6 Healthcare	92.6	74.1	70.7	77.9	93.3	90.3	82.8	**95.1**	39.4	69.0	34.3	61.8	55.2	74.5
D7 Legal	**96.7**	81.5	69.1	66.6	56.1	92.8	62.0	74.3	17.9	58.8	56.0	42.2	51.7	58.3
D8 Real Estate	94.0	93.7	93.0	73.8	93.2	**95.9**	93.8	93.4	90.8	93.6	37.8	89.8	85.8	94.7

Table 2: Unified value F1 (%) by dimension and system. Models are columns, grouped by system type, and dimensions are rows. Within each row, **bold** and underlined mark the highest and second-highest scores. Red shading marks drops of more than 5/15/25 points from each system's overall score (darker means larger). Overall aggregates each system's evaluated documents. Table 14 (Section D.2) reports every sub-tag.

context limits: most systems cannot work through a long document in a single pass, and those without a strategy for iterating over it stop early, truncating the remaining records.

Task challenge. Long-list completeness (T1) and needle-in-haystack (T2) mirror the document-length results: LlamaExtract Agentic Plus and Reducto Deep Extract are the top two systems on both challenges. Dense documents (T3) reorder the ranking. This challenge concentrates many fields into a single large schema, and systems that handle such schemas poorly fall behind: Reducto Deep Extract drops to 87.5%, Claude Code Opus 4.8 to 82.4%, and Datalab Accurate Parse + Balanced Extract to 54.4%. LlamaExtract Agentic Plus (95.5%) and Codex GPT-5.5 (95.4%) lead on this challenge. Section D.2 reports every sub-tag separately.

Perception challenge. The perception axis exposes system-specific blind spots. Codex GPT-5.5 handles rotated or image-only capture (P1) poorly, dropping to about 80% from roughly 93% on the other perception chal-

System	Word-level grounding F1				Page-level grounding F1			
	Overall	Short	Medium	Long	Overall	Short	Medium	Long
LE Agentic Plus	**46.4**	**43.7**	**54.0**	**54.7**	**84.9**	**89.7**	**72.2**	**87.1**
LE Agentic	44.1	42.3	50.5	45.7	66.1	69.7	56.6	67.6
LE Cost-Eff.	40.4	40.2	42.3	36.7	64.2	68.9	53.7	56.5
Datalab A+B	2.0	2.7	0.2	0.0	48.5	56.9	38.6	0.0
Extend Max	25.1	33.9	0.2	0.0	48.9	61.7	27.7	0.0
Reducto Deep	43.3	42.8	45.6	41.1	71.7	72.6	70.4	67.3
All other systems	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 3: Grounding score (%). Word-level grounding F1 requires a correct value and word-level box at IoU 0.5; page-level grounding F1 requires a correct value and the source page. The final row spans all systems not named above. **Bold** marks each column's best value, underlined the second best. LE Agentic and LE Cost-Effective return word-level boxes only when the caller enables the granular bounding-box option; both are run with it on.

lenges. Reducto Deep Extract shows the complementary weakness: it stays above 90% on rotated or image-only capture and on handwriting (P3), but falls to about 81% on scanned pages (P2). Qwen3.6 35B-A3B is stronger on scanned pages and handwriting than the other VLMs, scoring above 92% on both. LlamaExtract Agentic Plus remains the strongest system across all three perception challenges.

Table structure. Enormous tables (S4, beyond a thousand rows) produce the sharpest separation in Table 2. Most systems stop early and return only a small fraction of the records: every VLM scores below 10% on this slice, and Datalab Accurate Parse + Balanced Extract (32.7%) and Extend Max Context (24.8%) also fall sharply. By contrast, LlamaExtract Agentic Plus (95.9%), Reducto Deep Extract (95.3%), and Claude Code Opus 4.8 (87.8%) are the top three systems. Cross-page tables (S3) pose a milder version of the same failure, where the difficulty is carrying the table structure across page breaks. Pivoted layouts (S2) are the least discriminative of the structure slices, because most leading systems handle them well.

Business domains. Domain difficulty largely reflects the mix of task challenges inside each domain (Table 2). Finance (D1) and government (D3) carry the long-list (T1) and needle-in-haystack (T2) tasks; energy (D2) is dominated by scanned dense forms (T3); and legal (D7) and healthcare (D6) hold long record lists, including creditor matrices, sanctions lists, and clinical event logs. A domain's score is therefore mostly a reweighting of its task-challenge results, and we read the domain axis as a check on coverage rather than as an independent source of difficulty.

3.4 The Grounding Gap

Grounding makes extraction auditable by letting a reviewer trace each predicted value back to its source. ExtractBench measures this capability explicitly, whereas existing schema-guided extraction benchmarks overlook it (Table 1). We consider a field grounded only when both the extracted value and its citation are correct, at the page- or word-level. We highlight the grounding gap in Table 3.

- **VLMs and coding agents** do not return evidence by default; they therefore score zero at both grounding levels. Users who need auditable outputs must add a separate evidence-localization component or use an extraction API with grounding built in.
- **Granularity challenge.** Locating the exact word is much more challenging than finding the correct page. LlamaExtract Agentic Plus achieves 84.9% page-level grounding F1 but only 46.4% word-level F1. Datalab shows a larger gap, at 48.5% versus 2.0%. Page evidence narrows the search, but still leaves reviewers to locate the value among similar candidates.
- **Robustness to length.** Extend Max Context falls from 61.7% page-level grounding F1 on short documents to 0.0% on long documents, and Datalab follows the same pattern. Reducto Deep Extract is more stable, declining from 72.6% to 67.3%, while LlamaExtract Agentic Plus remains the strongest system overall.

Even the best overall word-level grounding F1 remains below 50%. Systems are increasingly capable of extracting values and often identifying their source pages, but reliably connecting each value to its exact supporting evidence remains an open problem.

4 Related Work

4.1 Benchmarks for Document Extraction

Document extraction benchmarks cover two main settings: fixed-ontology extraction, where fields are defined in advance, and *schema-guided extraction*, where the user supplies a schema defining the fields and output structure at inference time. We briefly review both settings below.

Fixed-ontology document IE. Fixed-ontology benchmarks ask how reliably a system can recover a known set of fields as the documents become more challenging. They have progressively expanded document diversity from forms and receipts to multilingual layouts and enterprise domains [19, 21, 33, 42, 44]. Other work increases structural and contextual complexity through line items, tables, long documents, and unfamiliar templates [17, 20, 39, 40, 43]. Some benchmarks additionally annotate spatial positions or study localization [39, 41]. Across these settings, however, the target fields remain fixed by the benchmark: they test robustness within a known ontology rather than whether a system can follow a new user-supplied schema.

Schema-guided extraction benchmarks. Recent document extraction benchmarks have focused on the schema-guided setting, where users specify the extraction task at inference time without retraining the system [12, 22, 38]. Work in this setting has progressively increased task scale and complexity, testing more complex schemas, longer documents, and larger outputs. ContextualAI's ExtractBench [12] stresses schema complexity, with schemas containing up to 369 fields, but evaluates only five shared schemas; LongExtractBench-50 [26] and VAREX [4] use a different schema for every document, so they do not test whether one extraction task transfers across diverse document appearances. LongArrayExtract [9] and LongExtractBench-50 [26] instead stress completeness over long documents and repeated records, though their public test sets contain only dozens of documents. Several benchmarks [4, 9, 22] use synthetic construction to scale these evaluations, but generated documents do not capture the visual variability and perception challenges found in real enterprise data. This fragmented coverage makes it difficult to compare system families comprehensively or diagnose why they fail.

To our knowledge, ExtractBench provides the broadest combined coverage of these dimensions, spanning real document families and targeted synthetic stress tests across 8 business domains (Table 1). ExtractBench is designed around production requirements at scale, jointly measuring value accuracy, source grounding, and per-page cost to capture whether outputs are correct, traceable, and economical to produce. We also stratify the dataset with challenge tags to ensure coverage across task and perception difficulties and diagnose where different system families fail.

4.2 Methods for Document Extraction

Modern schema-guided extraction systems fall into three broad families: general-purpose vision-language models that generate outputs directly, coding agents that inspect documents iteratively with tools, and specialized extraction systems designed around document processing workflows. These approaches make different tradeoffs in completeness, visual robustness, grounding, and cost.

Vision-language models. General-purpose vision-language models are multimodal reasoners that accept text and images and generate flexible outputs. Document extraction can therefore be reformulated as multimodal generation: document pages are rendered as images, the schema is expressed as text instructions, and the model returns extracted values either as prompted text or as schema-compliant output enforced through a structured-output API. Systems in this family include closed general-purpose models [13, 29], general-purpose open-weight models [14, 34], and models tuned specifically for extraction [7, 27]. All three groups accept new schemas without task-specific retraining. This direct, one-pass workflow is simple and efficient, but can miss values in long repeated structures, and the systems evaluated here do not return source evidence.

Coding agents. Coding agents such as Codex [28, 30] and Claude Code [1, 2] approach extraction through an iterative tool-use loop: given the document and schema as files, they can inspect pages, write parsing code, run checks, and revise the final JSON. This loop is more flexible than one-pass generation and can help with long documents or repeated records, where the agent can revisit the document rather than rely on a single model response. The same flexibility creates cost and reliability risks: even short documents may trigger many inspection, coding, debugging, and validation steps, and unconstrained agents can run for many steps before producing a small extraction. Coding agents also require an agent runtime with filesystem access and validation, depend on how documents are rendered and which tools are available, and are not designed around extraction-specific grounding metadata.

Specialized document extractors. Commercial platforms [6, 10, 24, 36] expose extraction as a purpose-built API rather than a raw model prompt or coding environment. As managed services, they let users upload files

directly and handle format support, preprocessing, document parsing, and schema-guided extraction with little configuration. They can also expose visual grounding and other extraction metadata, such as source pages and, in some cases, word-level boxes, making outputs easier to audit than raw model responses. However, specialized APIs still differ substantially in completeness, robustness, grounding quality, and cost.

5 Conclusion

We introduced ExtractBench, a challenge-tagged benchmark that brings the core requirements of real schema-guided extraction into one evaluation: correct and complete outputs, source traceability, robustness across document challenges, and cost at scale. By measuring these dimensions together, ExtractBench shows not only which systems perform well, but where and why they fail.

Direct VLM extraction is inexpensive but often truncates long record lists; coding agents are more robust on these workloads but substantially more expensive. Specialized APIs span the quality–cost frontier, with LlamaExtract Agentic Plus achieving the strongest overall performance at a lower cost than the coding agents. Challenge-tagged results further show that systems degrade differently on long documents, dense schemas, scans, handwriting, and enormous tables.

Grounding remains the clearest area for improvement. The evaluated VLMs and coding agents do not return source evidence by default, while word-level grounding F1 remains below 50% even for specialized systems that return boxes. Together, these results set a clear bar for reliable extraction: complete outputs, traceable evidence, and sustainable cost at scale.

References

[1] Anthropic. Claude code, 2026. URL https://claude.com/product/claude-code. Accessed 2026-07-01.

[2] Anthropic. Claude opus 4.8, 2026. URL https://www.anthropic.com/news/claude-opus-4-8. Accessed 2026-07-01.

[3] Anthropic. Claude api pricing, 2026. URL https://platform.claude.com/docs/en/about-claude/pricing. Accessed 2026-07-01.

[4] Udi Barzelay, Ophir Azulai, Inbar Shapira, Idan Friedman, Foad Abo Dahood, Madison Lee, and Abraham Daniels. VAREX: A benchmark for multi-modal structured extraction from documents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 7368–7376, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026W/MMFM5/html/Barzelay_VAREX_A_Benchmark_for_Multi-Modal_Structured_Extraction_from_Documents_CVPRW_2026_paper.html.

[5] Brian Chivers, Mason P Jiang, Wonhee Lee, Amy Ng, Natalya I Rapstine, and Alex Storer. Ants: a framework for retrieval of text segments in unstructured documents. In *Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing*, pages 38–47, Hybrid, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.deeplo-1.5. URL https://aclanthology.org/2022.deeplo-1.5/.

[6] Datalab. Structured data extraction api overview, 2026. URL https://documentation.datalab.to/docs/recipes/structured-extraction/api-overview. Accessed 2026-07-01.

[7] Datalab. Lift: Open-source document extraction pipeline. GitHub repository, 2026. URL https://github.com/datalab-to/lift. Accessed 2026-07-01.

[8] Datalab. Datalab pricing, 2026. URL https://www.datalab.to/pricing. Accessed 2026-07-01.

[9] Extend AI. LongArray-Extract: Open-source array extraction benchmark. Hugging Face dataset, 2026. URL https://huggingface.co/datasets/Extend-AI/LongArray-Extract. Accessed 2026-07-01.

[10] Extend AI. Extend extraction configuration, 2026. URL https://docs.extend.ai/extraction/configuration. Accessed 2026-07-01.

[11] Extend AI. Extend pricing, 2026. URL https://www.extend.ai/pricing. Accessed 2026-07-01.

[12] Nick Ferguson, Josh Pennington, Narek Beghian, Aravind Mohan, Douwe Kiela, Sheshansh Agrawal, and Thien Hang Nguyen. ExtractBench: A benchmark and evaluation methodology for complex structured extraction. *arXiv preprint arXiv:2602.12247*, 2026. URL https://arxiv.org/abs/2602.12247.

[13] Google. Gemini 3.5 flash, 2026. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.5-flash. Accessed 2026-07-01.

[14] Google. Gemma 4 model overview, 2026. URL https://ai.google.dev/gemma/docs/core. Accessed 2026-07-01.

[15] Google. Gemini api pricing, 2026. URL https://ai.google.dev/gemini-api/docs/pricing. Accessed 2026-07-01.

[16] Ralph Grishman and Beth Sundheim. Message understanding conference-6: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 466–471, 1996.

[17] Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. CUAD: An expert-annotated NLP dataset for legal contract review. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

[18] Xavier Holt and Andrew Chisholm. Extracting structured data from invoices. In *Proceedings of the Australasian Language Technology Association Workshop 2018*, pages 53–59, Dunedin, New Zealand, December 2018. URL https://aclanthology.org/U18-1006/.

[19] Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and C. V. Jawahar. ICDAR2019 competition on scanned receipt OCR and information extraction. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, pages 1516–1520, 2019. doi: 10.1109/ICDAR.2019.00244.

[20] Goeric Huybrechts, Srikanth Ronanki, Sai Muralidhar Jayanthi, Jack Fitzgerald, and Srinivasan Veeravanallur. Document haystack: A long context multimodal image/document understanding vision LLM benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 4121–4129, October 2025. URL https://openaccess.thecvf.com/content/ICCV2025W/MRR%202025/html/Huybrechts_Document_Haystack_A_Long_Context_Multimodal_ImageDocument_Understanding_Vision_LLM_ICCVW_2025_paper.html.

[21] Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. FUNSD: A dataset for form understanding in noisy scanned documents. In *Proceedings of the International Conference on Document Analysis and Recognition Workshops (ICDARW)*, pages 22–26, 2019. doi: 10.1109/ICDARW.2019.10029.

[22] Yifan Ji, Zhipeng Xu, Zhenghao Liu, Zulong Chen, Qian Zhang, Zhibo Yang, Junyang Lin, Yu Gu, Ge Yu, and Maosong Sun. UNIKIE-BENCH: Benchmarking large multimodal models for key information extraction in visual documents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6331–6352, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.287. URL https://aclanthology.org/2026.acl-long.287/.

[23] Yiming Lin, Mawil Hasan, Rohan Kosalge, Alvin Cheung, and Aditya G Parameswaran. Visual template inference for data extraction from documents. *Proceedings of the ACM on Management of Data*, 3(6):1–27, 2025. doi: 10.1145/3769840. URL https://doi.org/10.1145/3769840.

[24] LlamaIndex. Llamaextract documentation, 2026. URL https://developers.llamaindex.ai/llamaparse/extract/. Accessed 2026-07-01.

[25] LlamaIndex. Llamaparse pricing, 2026. URL https://developers.llamaindex.ai/llamaparse/general/pricing/#extraction. Accessed 2026-07-01.

[26] micro1. LongExtractBench-50: Structured extraction on long, table-heavy documents. Hugging Face dataset; representative public subset of the 225-document benchmark commissioned by Reducto, 2026. URL https://huggingface.co/datasets/micro1-inc/longextract-bench-50. Accessed 2026-07-01.

[27] NuMind. NuExtract3, 2026. URL https://huggingface.co/numind/NuExtract3. Accessed 2026-07-01.

[28] OpenAI. Codex, 2026. URL https://openai.com/codex/. Accessed 2026-07-01.

[29] OpenAI. Introducing gpt-5.4 mini and nano, 2026. URL https://openai.com/index/introducing-gpt-5-4-mini-and-nano/. Accessed 2026-07-01.

[30] OpenAI. Introducing gpt-5.5, 2026. URL https://openai.com/index/introducing-gpt-5-5/. Accessed 2026-07-01.

[31] OpenAI. Gpt-5.5 model, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.5. Accessed 2026-07-01.

[32] OpenAI. Openai api pricing, 2026. URL https://developers.openai.com/api/docs/pricing. Accessed 2026-07-01.

[33] Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. CORD: A consolidated receipt dataset for post-OCR parsing. In *Proceedings of the NeurIPS Workshop on Document Intelligence*, 2019. URL https://github.com/clovaai/cord.

[34] Qwen Team. Qwen3.6-35B-A3B-FP8, 2026. URL https://huggingface.co/Qwen/Qwen3.6-35B-A3B-FP8. Accessed 2026-07-01.

[35] Reducto. Credit usage, 2026. URL https://docs.reducto.ai/reference/credit-usage#extract-endpoint. Accessed 2026-07-01.

[36] Reducto. Deep extract, 2026. URL https://docs.reducto.ai/configs/extract/deep-extract. Accessed 2026-07-01.

[37] Reducto. Pricing, 2026. URL https://reducto.ai/pricing. Accessed 2026-07-01.

[38] Mathieu Sibue, Andrés Muñoz Garza, Samuel Mensah, Pranav Shetty, Zhiqiang Ma, Xiaomo Liu, and Manuela Veloso. Exstructiny: A benchmark for schema-variable structured information extraction from document images. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5669–5688, Rabat, Morocco, March 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.eacl-long.265. URL https://aclanthology.org/2026.eacl-long.265/.

[39] Štěpán Šimsa, Milan Šulc, Michal Uřičář, Yash Patel, Ahmed Hamdi, Matěj Kocián, Matyáš Skalický, Jiří Matas, Antoine Doucet, Mickaël Coustaty, and Dimosthenis Karatzas. DocILE benchmark for document information localization and extraction. In *Document Analysis and Recognition – ICDAR 2023*, volume 14188 of *Lecture Notes in Computer Science*, pages 147–166. Springer, 2023. doi: 10.1007/978-3-031-41679-8_9. URL https://doi.org/10.1007/978-3-031-41679-8_9.

[40] Tomasz Stanisławek, Filip Graliński, Anna Wróblewska, Dawid Lipiński, Agnieszka Kaliska, Paulina Rosalska, Bartłomiej Topolski, and Przemysław Biecek. Kleister: Key information extraction datasets involving long documents with complex layouts. In *Document Analysis and Recognition – ICDAR 2021*, volume 12821 of *Lecture Notes in Computer Science*, pages 564–579. Springer, 2021. doi: 10.1007/978-3-030-86549-8_36. URL https://doi.org/10.1007/978-3-030-86549-8_36.

[41] Matthew Toles, Isaac Song, Rattandeep Singh, and Zhou Yu. FormGym: Doing paperwork with agents. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3771–3785, Rabat, Morocco, March 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.eacl-long.175. URL https://aclanthology.org/2026.eacl-long.175/.

[42] Benjamin Townsend, Madison May, Katherine Mackowiak, and Christopher M. Wells. RealKIE: Five novel datasets for enterprise key information extraction, 2024. URL https://arxiv.org/abs/2403.20101. Version 2, revised 2025.

[43] Zilong Wang, Yichao Zhou, Wei Wei, Chen-Yu Lee, and Sandeep Tata. VRDU: A benchmark for visually-rich document understanding. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5184–5193. Association for Computing Machinery, 2023. doi: 10.1145/3580305.3599929. URL https://doi.org/10.1145/3580305.3599929.

[44] Yiheng Xu, Tengchao Lv, Lei Cui, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, and Furu Wei. XFUND: A benchmark dataset for multilingual visually rich form understanding. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3214–3224. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-acl.253. URL https://aclanthology.org/2022.findings-acl.253/.

Appendix Contents

A Benchmark Details

A.1 Taxonomy and Coverage Reference

The consolidated taxonomy table defines every tag: what it stresses, representative document types, and tagged document and page coverage. The tags are shared with the capability comparison (Table 7) and with every result slice in Section 3; Figure 2 in the main text shows the challenge-level coverage as a distribution.

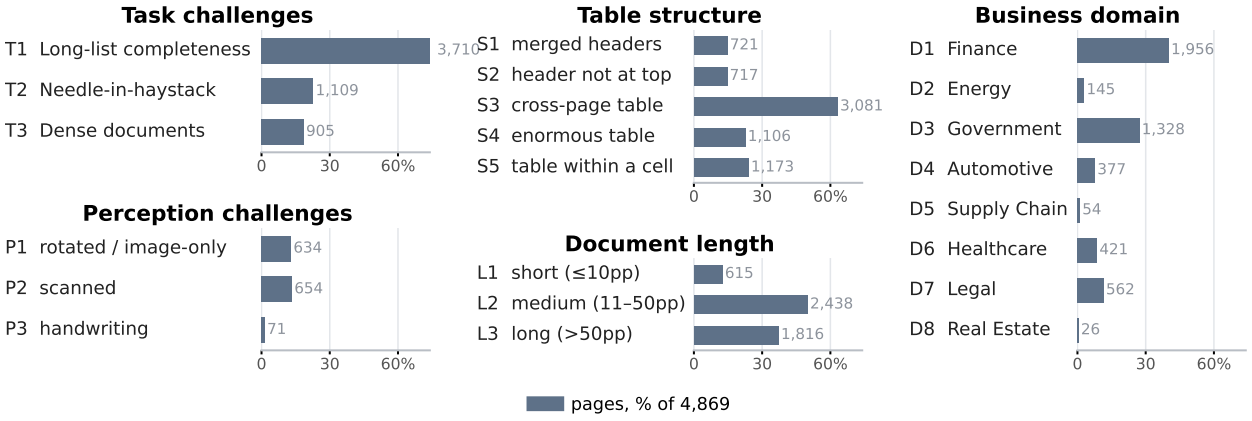


Figure 5: ExtractBench coverage by page share. Each bar is the share of the 4,869 corpus pages carrying the tag, with its page count shown beside the bar. Tags may overlap across panels; Table 4 defines every tag and sub-tag.

Task-challenge notes. Partial credit can hide F1 errors: a system can emit a well-formed array that is missing a third of its rows. T1 covers a large share of the benchmark and holds records at large scale: a real SEC 13F table with 3,063 holdings rows, a bankruptcy creditor matrix with 8,624 address-block records, and an unclaimed-property list with 26,725 rows. T2 failures include selecting the canonical value of a KPI that recurs many times under paraphrase, or a target field buried in procurement narrative. On T3 documents, field localization remains a major error source even with dedicated tools [41]. The T3.d slice is 13 administratively reviewed W-14 filings, whose twin schema asks for the original filer value and the later regulator annotation separately rather than merging the two.

Tag	What it stresses	Representative document types	Docs	Pages
Task Challenges				
T1.a single long table	one homogeneous table spanning pages → array	SEC 13F holdings, fund schedules, registers	99 (26.8%)	2681 (55.1%)
T1.b cross-page continuation	records continue past page breaks under repeated headers	GSA labor schedules, IRS Schedule I, clinical logs	91 (24.6%)	3081 (63.3%)
T1.c repeated complex region	each record a multi-field block, not a table row	OFAC SSI list, service lists, bankruptcy E/F	27 (7.3%)	492 (10.1%)
T1.d pivoted / matrix	entities down and values across columns, with the header not at top	auto valuations, census cross-tabs, election pivots	42 (11.4%)	717 (14.7%)
T1.e packed / multi-row cell	one record spans sub-rows, or one cell packs many fields	FTX / iMedia creditor matrices, CBP 7501	30 (8.1%)	1173 (24.1%)
T2.a sparse in narrative	few target fields buried in long prose	DD1155, CLIN schedules, SF1449	15 (4.1%)	617 (12.7%)
T2.b paraphrased match	pick the canonical occurrence of a paraphrased value	earnings decks, investor presentations	12 (3.2%)	356 (7.3%)
T2.c dedup across modalities	reconcile base vs. modified copies	contract modification sets	5 (1.4%)	222 (4.6%)
T2.d cross-ref / reconciliation	combine / check values across sections	SEFA schedules, audit reconciliations	24 (6.5%)	492 (10.1%)
T3.a dense form	labeled cells, checkboxes, blanks on a short form	RRC oil-and-gas forms, CBP 7501, closing disclosures	194 (52.4%)	798 (16.4%)
T3.b receipt / invoice	line-item business documents	invoices, receipts, purchase orders	17 (4.6%)	46 (0.9%)
T3.c classify then extract	route by document class before extracting	brokerage statement families	3 (0.8%)	61 (1.3%)
T3.d filer-reviewer separation	separate original filer entries from later regulator annotations	administratively reviewed W-14 filings	13 (3.5%)	13 (0.3%)
Perception Challenges				
P1 rotated / image-only	rotated or skewed page image, no text layer	scan-degraded re-captures	38 (10.3%)	634 (13.0%)
P2 scanned	scanned page image	scanned regulatory forms	134 (36.2%)	654 (13.4%)
P3 handwriting	handwriting on the page	hand-completed form fields	55 (14.9%)	71 (1.5%)
Table Structure				
S1 merged headers	hierarchical headers	banded financial statements	35 (9.5%)	721 (14.8%)
S2 header not at top / pivoted	pivoted layout	valuation and census matrices	42 (11.4%)	717 (14.7%)
S3 cross-page table	continues across pages	long procurement schedules	91 (24.6%)	3081 (63.3%)
S4 enormous table	very large table	13F holdings, unclaimed-property lists	12 (3.2%)	1106 (22.7%)
S5 table within a cell	nested table	creditor address blocks	30 (8.1%)	1173 (24.1%)
Document Length				
L1 short	up to 10 pages	receipts, single-page forms, short filings	252 (68.1%)	615 (12.6%)
L2 medium	11–50 pages	multi-page statements, mid-size filings	98 (26.5%)	2438 (50.1%)
L3 long	more than 50 pages	registers, holdings, long schedules	20 (5.4%)	1816 (37.3%)
Business Domain				
D1 Finance	financial disclosures, holdings, and tax records	13F / N-PORT, fund schedules, 1040, W-2, K-1, 1099-B	145 (39.2%)	1956 (40.2%)
D2 Energy	regulatory forms and filer-reviewer annotations	Texas RRC W-1, W-2, W-14, 2A, P-4, P-18, H-5	98 (26.5%)	145 (3.0%)
D3 Government	procurement, customs, and public reporting	CBP 7501, GSA labor, IRS-990, SEFA, CLIN / SF-1449	49 (13.2%)	1328 (27.3%)
D4 Automotive	valuation reports and comparison tables	CCC / Mitchell total-loss valuations	27 (7.3%)	377 (7.7%)
D5 Supply Chain	transactional documents and itemized records	invoices, receipts, rate cards, product specs, utility bills	20 (5.4%)	54 (1.1%)
D6 Healthcare	remittance and clinical-event records	remittance advice, adverse-event / deviation logs	15 (4.1%)	421 (8.6%)
D7 Legal	filings, creditor matrices, and entity lists	bankruptcy schedules, creditor matrix, sanctions list	10 (2.7%)	562 (11.5%)
D8 Real Estate	mortgage closing disclosures	TRID mortgage closing disclosure	6 (1.6%)	26 (0.5%)

Table 4: ExtractBench taxonomy and coverage. Each row gives a task challenge, perception challenge, table structure, size, or business-domain slice, with representative documents and tagged document and page coverage. Percentages are shares of the 370-document, 4,869-page benchmark. Tags may overlap.

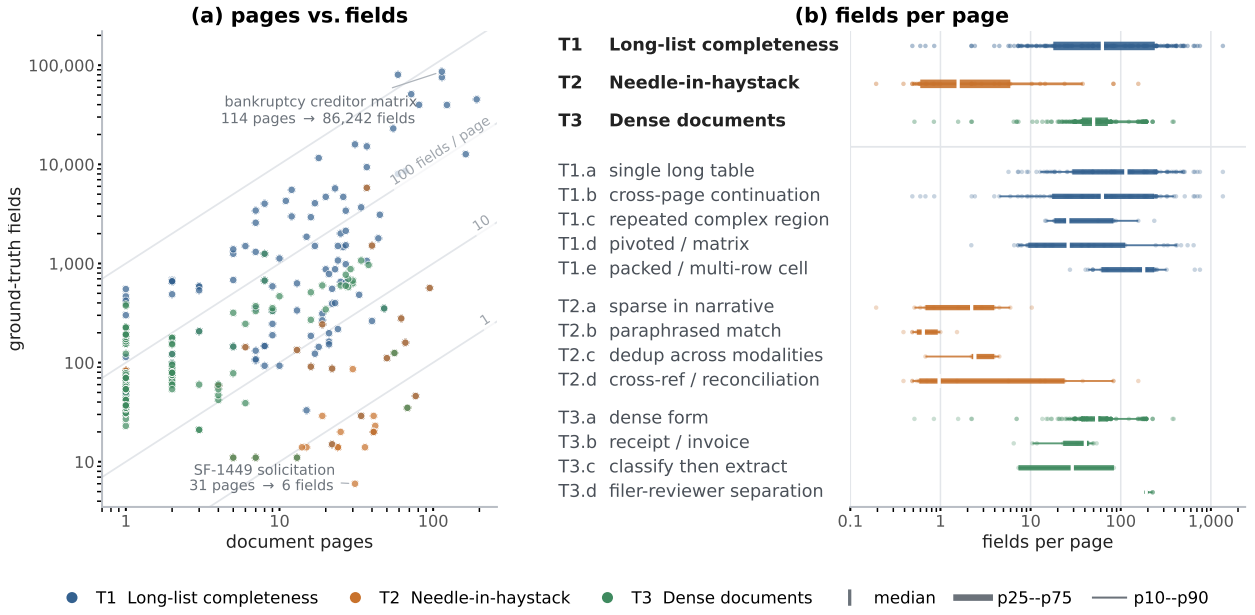


Figure 6: Document size against ground-truth size, by task challenge. A *field* is one cell of the unified value metric: one per scalar field and per record subfield (Section 2.4). A document's field count is therefore the number of cells a system must return correctly for full recall on it. **(a)** One point per document, over the 332 pool documents that are not scan-degraded re-captures; a document tagged with more than one task challenge appears once per challenge. The diagonals mark a constant number of fields per page. **(b)** The same rate by task challenge, then by sub-tag in code order beneath a dividing rule, with the p10–p90 range, the p25–p75 range, and the median; a document carrying several sub-tags appears in each of its rows. Table 4 gives the document count behind every row. Document size is counted in pages rather than in document text because most pages of the form and scan-degraded slices carry no text layer. The 38 scan-degraded re-captures are omitted: a re-capture keeps its original's schema, values, and page count, so it lands on the same point.

Grounding tags. An extraction is auditable only if each value points back to where it came from, so a separate set of tags records the required evidence: a box around each extracted value (G1) and a box on a checkbox together with the boolean read from it (G4). G2 covers a field that expands into many records, and G3 covers deeply nested objects and arrays. These tags constrain the output rather than the document; Table 5 gives the coverage of each. Section 3.4 scores grounding; to support these tags, the ground truth of Section 2.3 pairs every value with a location.

Grounding (G)	What it requires	Docs
G1 value-level box	box at each extracted value	237 (64.1%)
G2 1:N cardinality	one field expands to many records	36 (9.7%)
G3 deep nesting	deeply nested objects and arrays	8 (2.2%)
G4 checkbox / boolean box	read and locate a checkbox	182 (49.2%)

Table 5: Grounding tags: what the output must carry besides the value—a box at each value, cardinality, nesting, and checkbox handling. Document counts are shares of the benchmark and may overlap. Grounding is scored in Section 3.4.

A.1.1 Document and Ground-Truth Size by Task Challenge

The task challenges are defined by what makes extraction hard (Section 2.2). They also separate quantitatively, by how many fields a document yields per page (Figure 6). Across the pool that rate spans a factor of 7,021, from 0.19 fields per page to 1,359.

T2 is the only task challenge that compresses. Needle-in-haystack documents yield a median of 1.6 fields per page: a median of 25 pages read for 35 fields returned, so the system reads a long document and keeps almost none of it. The extreme case is an SF-1449 solicitation whose schema asks for 6 fields across 31 pages. Both other task challenges return more than an order of magnitude more fields per page.

T1 and T3 return dozens of fields per page. Long-list documents yield a median of 62.6 fields per page and dense documents 50.0; the output re-encodes most of the document rather than summarizing it. 48 documents carry more than 1,000 ground-truth fields and 12 more than 10,000, up to 86,242 fields in a 114-page bankruptcy creditor matrix. A system that reads such a document correctly still has to return every one of them, which is the long-list completeness failure of Section 3.3: precision stays high while recall falls.

The rate alone does not define the taxonomy. T1 and T3 overlap in Figure 6b and separate by scale instead: the median T1 document is 17 pages against 1 for T3, so the same field density arrives either as one dense page or as tens of pages of records. This is why the benchmark tags task challenge and document length on independent axes (Section 2.2) rather than collapsing both into a single difficulty score.

A.2 Corpus Composition

This appendix details the corpus by document length. Every document carries exactly one length class, so the regulatory and tax forms, the automotive valuations, and the scan-degraded re-captures are described inside the class their page count puts them in rather than as separate slices. Each re-capture pairs one-to-one with a clean original that also appears in the benchmark, under the same schema and the same expected values (Section A.3.4).

Benchmark	Domains	Schemas/types	Documents	Max pages
ContextualAI EB [12]	5	5	35	218
Extend LongArray [9]	3	3	45	235
Micro1 LongExtract-50 [26]	7	50	50	11,622
VAREX [4]	1	1,798[†]	1,798	1
DocILE [39]	1	1	6,680	3
VRDU [43]	2	2	2,556	12
RealKIE [42]	5	5	1,867	198
Kleister [40]	2	2	3,318	368
CUAD [17]	1	1	510	154
Legacy KIE [19, 21, 33, 44]	2	4	3,565	1
ExtractBench	8	**67**	370	192

Table 6: Dataset scale and diversity. Document and page counts show scale. Domain and schema counts show breadth. The schema figures are not directly comparable: [†] VAREX uses one schema per synthetic single-page form, Micro1 uses a model-drafted schema per document, and ExtractBench reuses a document-type schema across documents.

A.2.1 Short Documents ($\leq$10 pages)

252 documents, 615 pages: 68 multi-domain business documents, 155 regulatory and tax forms, 6 automotive total-loss valuations, and 23 scan-degraded re-captures.

Real families span finance, commodity business documents, government and customs forms, healthcare remittance advice, mortgage closing disclosures, product spec sheets, and auto total-loss valuations. Examples include SEC 13F and N-PORT holdings, invoices and receipts, CBP-7501 entry continuations, and census and budget cross-tabs with header-not-at-top pivots. Synthetic re-renders add a bankruptcy Schedule E/F with contingent/unliquidated/disputed booleans and election statement-of-votes pivots whose positional vote arrays align to per-contest candidate columns. These short documents concentrate same-type identifiers and value–label disambiguation: electric-versus-gas meter IDs on one bill, vendor versus customer versus account numbers on one invoice, and structurally identical money fields whose meaning differs by section. The automotive reports are pivoted comparison documents: the vehicle under appraisal runs down the page while comparable vehicles run across it, with the header beside the data rather than above it. They are the densest concentration of the pivoted/matrix structure (S2) in the benchmark.

The forms are Texas Railroad Commission energy filings (drilling permits, well completions, injection and disposal permits, enhanced-oil-recovery designations, H_2S certificates, pressure tests, plug records, transportation authorities, and skim-oil reports) and federal tax documents (Form 1040 returns, W-2 wage statements, Schedule K-1, and 1099-B pages). They span 594 pages, from 1950s typewritten-on-scan filings to current born-digital output, and carry 17 distinct frozen schemas across thirteen form families, because Form 1040 has a separate schema per tax year and Schedule K-1 has separate partnership and S-corporation schemas. All but the longer Form 1040 bundles are ten pages or fewer; the 14 that are not appear in Section A.2.2. What they stress is dense labelled cells, checkbox banks, blanks that must come back null, handwriting and scan noise on the older filings, and identifiers of the same shape whose meaning differs by section. Every document carries the whole filed form, nothing cropped. A human annotator reviews every form field by field; the tax forms additionally pass through schema-first adjudication and an audit against the rendered pages. Of the 16,534 human-verified evidence rules across these documents, 13,867 ($\sim$84%) carry a human-placed value box. Most of the remainder are fields the form leaves blank, which have nothing on the page to box.

All form documents are public records. The energy forms are regulatory filings served by the Texas Railroad Commission's public records. The tax forms come from public releases: 1040 returns released in full by public officials, W-2 wage statements from a public utility district's employer reference-copy register, and K-1 and 1099-B schedule pages from publicly filed documents. Personal taxpayer identifiers were masked in those releases themselves — employee and taxpayer SSNs appear on the page as masked strings (e.g., `XXX-XX-XXXX`), and the ground truth expects the masked string — while the remaining identifiers, such as employer EINs, are business identifiers printed on public filings.

A.2.2 Medium Documents (11–50 pages)

98 documents, 2,438 pages: 64 multi-domain reports, 14 Form 1040 bundles, 8 automotive valuations, and 12 scan-degraded re-captures.

Real families include financial-KPI documents—earnings decks, annual-report extracts, and press releases—scored against a master KPI schema that requires canonical-occurrence selection and GAAP/non-GAAP reconciliation. Other examples are SEFA single-audit schedules with hierarchical headers, SEC N-PORT holdings with nested detail, IRS Schedule I grant tables with multi-line addresses across pages, GSA labor-rate schedules with isolated pivots, merged brokerage/1099 statements that require classification, and auto total-loss valuations with a bookout matrix and option grid. Synthetic re-renders add clinical protocol-deviation logs with per-entry timestamped comment threads (deep nesting) and county audit lists whose record blocks reconcile to multiple rollup levels. The Form 1040 bundles that land here are the benchmark's largest schemas, carrying optional schedules and absent sub-forms that must come back null.

A.2.3 Long Documents (>50 pages)

20 documents, 1,816 pages: 17 multi-domain filings and registers, and 3 scan-degraded re-captures.

Real documents include an SEC 13F information table (3,063 holdings rows), a government CLIN schedule (62 pages, sparse fields in narrative plus base-versus-modification deduplication), an SF1449 solicitation (95 pages, line items leaking across page breaks), and a DD1155 schedule continuation (56 pages, sparse fields plus handwritten signatures). Synthetic re-renders include a bankruptcy creditor matrix (8,624 street-address blocks, packed cells at scale), an unclaimed-property list (26,725 rows, the benchmark's extreme truncation stress), and a rotated-landscape clinical adverse-event listing (163 pages), all with exact page and word-level box evidence by construction.

A.3 Annotation Methodology in Detail

This appendix gives the complete procedures for the three ground-truth methods in Section 2.3.

Evidence lists. Ground truth records each field as an *evidence list*: the expected value, any alternate defensible readings, and for each reading a source page and, where reviewed, a word-level box. Scoring accepts a prediction matching *any* listed reading (*OR-acceptance*), and every expected record still counts against recall. A field's evidence list holds more than one entry in two cases: a value can be cited from several places (a drug name that appears in the title, the indications paragraph, and a dosage table), or a genuinely ambiguous field has more than one defensible

reading, each carrying its own box. The annotator ratifies the list, and OR-acceptance scores predictions against it (Section 2.4).

A.3.1 Real Documents

Pipeline. For each document family the workflow is as follows. (1) Collect representative PDFs for the family (invoices, remittance advice, KPI reports, ...) and confirm membership. (2) Draft or refine the schema. Every field description carries aliases, format requirements, location hints, and do-not-confuse guidance. (3) Run multiple extraction systems from different model and pipeline families against the same schema. A smaller cross-extract trio drives the schema-discovery loop, and a broader ensemble drives verification. (4) Compare outputs cell by cell, aligning repeated rows by a declared identity key (invoice line ID, claim number, VIN, check number) rather than by list position. (5) Classify each disagreement as schema ambiguity, model failure, or unresolvable. (6) Build candidate ground truth only from agreed or verified cells, blocking cells that remain schema-ambiguous. (7) Re-score cached extraction outputs against the candidate ground truth before promoting it. (8) *Human check and fix*: An annotator takes the cells the models disagreed on and the ones the coding agent could not settle from the page itself, checks each against the PDF, and fixes the value before the family ships.

Two operating modes. The same machinery serves both: *schema discovery* creates a new family, and *ground-truth review* audits an existing one by hunting cells where many pipelines fail against the current truth, verifying them against the source PDF, and patching a copied dataset.

A.3.2 Synthetic Long Lists

Pipeline. (1) Choose a real long-list layout pattern (repeated rows, sectioned registers, continuation pages, totals, nested records); the layout may be a fund schedule, a holdings register, or a creditor matrix sampled from real templates. (2) Build the full structured content first (records, fields, nulls, totals, hierarchy, normalization rules). (3) Render the content into a realistic PDF in that pattern. (4) Paginate by *measurement*: rendered blocks are measured and packed into pages by actual size, so a long record takes more space, a heading stays attached to its item, and no page stops early. Fixed rows-per-page pagination creates artificial page breaks and wrong source-page labels. (5) Derive ground truth from the render: locate every record and field in the final PDF and assign source pages, quotes, and word-level boxes from what is actually visible. (6) Validate consistency across PDF, expected JSON, schema, evidence, page references, and boxes. (7) Audit semantically by running several extraction systems and inspecting aggregated disagreements for schema or generation defects that mechanical checks cannot see.

A.3.3 Scanned Forms

Five stages. (0) *Corpus*: Select documents per form type. (1) *Schema* (agent-only): Draft from the blank form template, check the fit against real scans, tighten over a small refinement loop, and freeze before any document is labeled. A schema revision after aggregation begins requires a version bump and targeted re-review, never silent reinterpretation of old votes. (2) *Aggregate and adjudicate*: An ensemble of up to five systems votes per schema leaf and contested leaves go to adjudication against the rendered page. (3) *Human review*: An annotator reviews the proposed value and box for each field, then accepts, edits, nulls, or redraws it. (4) *Post-QA*: The reviewed ground truth is evaluated against the same systems again to identify remaining inconsistencies.

Value consensus. Because the schema is frozen, every pipeline answers the identical question per leaf. Omitting a path is a null vote, not an abstention. Votes are normalized by the field's comparator (case, whitespace, date, boolean) before grouping, and array rows align across pipelines by declared identity key, never list position. Each leaf lands in a tier: *unanimous*, *majority*, *split* (plurality proposal, queued for adjudication), or *all-null*. A null majority over a real minority value is also queued, since the ensemble is built to catch silent field loss. The ensemble draws on models from several independent families, so no one system decides a value on its own. Adjudication load scales with the size of the schema: small forms converge with no adjudicated fields at all, while the largest need a handful per document.

Adjudication rules. The adjudicator must inspect the rendered page before ruling. Checkboxes are two-state wherever their page is in the filing: the verdict is *true* (a mark is present) or *false* (no mark, including when the box is not printed on this copy), and a *false* verdict needs no supporting quote since there is nothing on the page to cite. A checkbox on a page the filing does not include is null.

One-source boxes. Value votes benefit from redundancy, but agreement among predicted boxes does not verify the cited location. Bounding boxes are never merged across systems. Each field's box comes from a single citation-emitting pipeline in the pool. A field with a nullish value needs no box, and a field with no valid citation is handed to the annotator to draw.

A.3.4 Scan-Degraded Re-Captures

38 documents are degraded re-captures from the sources above. Their expected values stay the same, so the difference between clean and degraded scores measures the effect of capture degradation. This slice is also what populates the P1 rotated / image-only tag of Table 4: each document appears twice, clean and degraded, under the same schema and the same expected values, so only the capture differs.

Pipeline. Each of the 38 documents (30 real and 8 synthetic) is rendered to page images, slightly rotated, given a slight perspective shift, and passed through one scan recipe from a fixed library. The recipes include photocopier and carbon-copy tone curves, fax thresholding, sensor and speckle noise, phone-camera capture, aging and bleed-through, low-resolution resampling, dust, and shadowed copying. Each document's recipe and seed are recorded, so it can be regenerated byte-for-byte. Recipes that warp the page non-rigidly (creases, book curvature, elastic deformation) are excluded, because their effect on a word box cannot be written down in closed form.

Ground truth. Values, schema, and tags are copied from the clean document unchanged. Boxes are stored normalized to the page, which makes them invariant to every photometric effect and to uniform rescaling, so only the rotation and perspective steps move them; those are applied to the box corners in closed form.

A.4 Full Capability Comparison

The capability summary of the introduction merges related taxonomy tags into single rows. The matrix below scores every tag separately, using the tags defined in Section A.1. Benchmark columns are grouped by task specification: schema-guided extraction supplies the target schema at evaluation time, whereas fixed-ontology benchmarks predefine the labels. A filled circle denotes covered and scored, a hollow circle partial or incidental coverage, and a blank absence. Each mark is verified against the benchmark's primary source. Legacy KIE combines FUNSD, SROIE, CORD, and XFUND. Table 1 gives the main-text summary, including domain breadth and measured cost.

	Schema-guided					Fixed ontology					
Dimension	**ExtractBench**	ContextualAI EB [12]	Extend LongArray [9]	Micro1 LongExtract-50 [26]	VAREX [4]	RealKIE [42]	DocILE [39]	VRDU [43]	Legacy KIE [19, 21, 33, 44]	Kleister [40]	CUAD [17]
T1: long-list completeness											
T1.a single long table	●	●	●	●	○	○	●	○	○		
T1.b cross-page continuation	●	○	●	●			○				
T1.c repeated complex region	●	●	○	○	○			●	●		
T1.d pivoted / matrix	●			○							
T1.e packed / multi-row cell	●						○		○		
T2: needle-in-haystack											
T2.a sparse in narrative	●	○		○		●				●	●
T2.b paraphrased match	●	○				○	○	●		●	●
T2.c dedup across modalities	●		○							○	
T2.d cross-ref / reconciliation	●		○						○	○	○
T3: dense documents											
T3.a dense form	●	○			●	○	○	●	●		
T3.b receipt / invoice	●					●	●	○	●		
T3.c classify then extract	●										○
T3.d filer-reviewer separation	●										
Perception challenges											
P1 rotated / image-only	●	○			○	○		○	○	○	
P2 scanned	●					●	○	○	●	●	
P3 handwriting	●					○		○	○	○	
Table structure											
S1 merged headers	●			○				○	○		
S2 header not at top / pivoted	●			○							
S3 cross-page table	●	○	●	●		○	○	●		○	○
S4 enormous table	●	○	●	●							
S5 table within a cell	●								●		
Grounding & output trust											
G1 value-level box	●					○	●	●	●		○
G2 1:N cardinality	●	●	●	●	●	●	●	●	●	○	●
G3 deep nesting	●	○	○	○	○			●	○		
G4 checkbox / boolean box	●										

● = covered and scored ○ = partial or incidental blank = absent.

Table 7: Full comparison of benchmark task specification and capability coverage.

B Metric Details

This appendix gives the exact matching, normalization, and aggregation rules behind the unified value F1 of Section 2.4. Scoring is fully deterministic: the same predictions and ground truth always produce the same score, with no model in the loop.

B.1 Cell Matching and Normalization

Two cells are compared under the following rules, in order; the first rule that applies decides.

1. *Date canonicalization.* Before any comparison, every string on either side that matches one of eight common date formats (`2019-03-28`, `3/28/2019`, `3/28/19`, `3-28-2019`, `March 28, 2019`, `Thursday March 28 2019`, `28 March 2019`, and hyphenated variants) is rewritten to ISO `YYYY-MM-DD`. Guards keep non-dates intact: candidates shorter than 4 or longer than 50 characters, all-digit strings, strings containing a run of ten or more digits, and parses outside 1900–2100 pass through unchanged.
2. *Long-list comparability.* So that our long-list numbers can be read against Extend's LongArray benchmark, we adopt its public reference scorer verbatim, including the two free-text fields it compares by edit-distance ratio rather than exactly [9]. No other field in the benchmark uses fuzzy matching.
3. *Strings.* Whitespace runs collapse to single spaces and outer whitespace is trimmed; the comparison is then exact and case-sensitive. There is no punctuation stripping, unicode folding, or currency/thousands handling.
4. *Everything else.* Plain equality. Numbers carry no tolerance, and a number never equals its string rendering (`"1,000"` $\neq$ `1000`). A list of scalars inside a record compares as one opaque, order-sensitive value; order-invariance applies to records, not to scalar lists.

Two acceptance layers apply after exact matching. Both are declared in the ground truth before scoring and applied identically to every system. First, a scanned-form field can declare *opt-in leniencies* where the printed template makes one strict reading unfair (Table 8). Second, by OR-acceptance over the evidence list (Section A.3), a prediction is correct when it matches the expected value or any recorded alternate reading.

Leniency	Rule
`null_equals_false`	a blank checkbox may read as `false` or as null
`case_insensitive`	casefolded comparison (typewriter and stamp case)
`optional_terminal_punctuation`	one trailing `. , ; :` is ignored on each side
`punctuation_spacing`	whitespace around `. , ; :` is insignificant
`phone_digits`	phone numbers compare by their last ten digits
`lenient_date`	split preprinted years rejoin ("19 55"), two-digit years try both centuries

Table 8: Opt-in, per-field leniencies for scanned forms. Each is declared in the ground truth for specific fields and applies only after the exact match misses, so a leniency can never turn a passing cell into a failure.

B.2 Missing-Value Semantics

Every scalar field of the schema enters both the precision and the recall denominator, and a key absent from the output is scored identically to an explicit null. Table 9 lists every case. Two consequences follow: a hallucinated value on a blank field costs both precision and recall (the cell sits in both denominators), and correctly returning null for a blank field is credited, so a system cannot be hurt by faithful nulls.

Expected	Predicted	Outcome
value	matching value	correct (counts toward P and R)
value	different value	miss in both P and R
value	null or key omitted	miss in both P and R
null	null or key omitted	correct (counts toward P and R)
null	value	miss in both P and R (hallucination)
records	fewer records	each missing record's cells miss in R only
records	extra / duplicated records	each extra record's cells miss in P only
null or `[]`	`[]` or null	no cells contributed

Table 9: Scoring outcome for every (expected, predicted) state. Scalar cells are symmetric between precision and recall; only repeated records move the two sides apart, which is what makes the precision–recall split of Section D.1 a truncation diagnostic.

B.3 Array Alignment

Records pair by a globally optimal one-to-one assignment (`linear_sum_assignment`, the Hungarian family) whose cost between an expected and a predicted record is the number of mismatched declared subfield cells; minimizing total cost is equivalent to maximizing total field agreement over the array. The assignment is not greedy, consults no identity keys, and record order never affects the value score. Rows whose cells match exactly are pre-paired by a hash join as a provably score-preserving fast path. Duplicated predictions pair at most once; the surplus copies count only against precision. A record subfield that is itself a list of records recurses with an independent assignment per matched pair.

B.4 Aggregation and Grounding

The per-document score is micro precision/recall/F1 over the document's cell bag; slice and overall scores are unweighted means of per-document values, so a document with more fields does not weigh more, and a slice's mean F1 is not the harmonic mean of its mean P and R. A document a system fails to return, whether it rejects the schema, errors, or returns no JSON object, scores zero rather than being dropped, so every system is averaged over the same document set and a refusal is penalized like any other miss.

Grounding metrics gate on value correctness: a cell is grounded-correct when its value is accepted and a predicted citation box overlaps any evidence box for that field at IoU ≥ 0.5 on the correct page. The grounding precision denominator counts only gradeable claims (citations on cells aligned to box-bearing ground truth), the recall denominator counts ground-truth cells that carry a verified box, and documents with no verified boxes emit no grounding score at all rather than zero. Page-level evidence replaces the box test with page membership. For each document on which all three metrics are defined, unified value F1 $\geq$ page F1 $\geq$ word-level grounding F1.

C Evaluation Protocol

This section documents the prompts, configurations, and cost rules underlying the evaluation in the main text. Section C.1 specifies how each system is run, while Section C.2 records the cost-accounting rules and provider rates.

C.1 System Configuration

This subsection records the exact prompts and consequential configurations underlying the evaluation setup in Section 3.1. The accompanying benchmark release contains the complete integrations and provider-specific options.

VLM APIs. GPT-5.4 Nano and Gemini 3.5 Flash receive the document and the same benchmark prompts.

> **System prompt.** *"You are extracting structured data from a document according to the provided JSON schema. Return only the JSON that matches the schema. Use null for fields not present in the document. When the schema includes a list field, populate every relevant row visible in the document – do not return an empty list when rows are present."*
>
> **User prompt.** *"Extract every field from the attached document according to the schema. Return JSON only. Use null for fields not present in the document. Whenever the schema declares a list field, enumerate every row visible in the document – do not collapse rows or return an empty list when rows are present."*

GPT-5.4 Nano uses the OpenAI Responses API with the target schema supplied as `text.format=json_schema` and `strict=false`. Gemini 3.5 Flash uses `response_json_schema` with temperature zero and low thinking. For GPT, the harness recursively sets `additionalProperties=false`; for Gemini, it also promotes repeated-record nodes into ordinary JSON-Schema properties. These mechanical transformations do not change the requested fields or descriptions.

Self-hosted VLMs. Lift 9B passes the task schema through the official Lift SDK to vLLM guided-JSON decoding. Qwen3.6 35B-A3B uses vLLM's `json_schema` response format with xgrammar-guided decoding. NuExtract3 receives a mechanical conversion of the task schema into its native extraction-template format rather than a generic JSON-Schema constraint. Gemma4 26B receives the schema in the prompt and uses vLLM's `json_object` mode, which enforces valid JSON but not the complete schema. Qwen3.6 35B-A3B and Gemma4 use the benchmark prompts above; Lift and NuExtract3 use their model-native interfaces.

Coding agents. Claude Code Opus 4.8 and Codex GPT-5.5 run through their vendor CLIs in isolated working directories containing the staged document and target schema. Both may use local computation, are instructed to rely only on the provided materials, and have a 1,200-second per-document timeout. Codex uses low reasoning effort.

The shared task prompt below is quoted verbatim; `[DOCUMENT]` and `[SCHEMA]` mark the staged filename and full task schema inserted for each example.

> **Shared Claude Code and Codex task prompt.** *"Extract structured data from the document file './[DOCUMENT]' in the current directory.*
>
> *Use only local file inspection and shell commands. Do not use web search, network calls, browser tools, or external services. Temporary scratch files inside the current directory are OK.*
>
> *Return a single JSON object conforming to this schema as your final answer:*
>
> `[SCHEMA]`
>
> *Rules:*
>
> *- Use null for fields not present in the document.*
>
> *- For list/array fields, enumerate every relevant row visible in the document; never collapse rows.*
>
> *- For large regular tables, prefer writing and running a local script to parse/enumerate rows.*
>
> *- For forms, prefer direct field extraction from the document content.*
>
> *- Write the resulting JSON object to ./output.json and validate that it is valid JSON before stopping.*
>
> *- Do not print the JSON to your assistant output."*

Specialized extraction APIs. Specialized extraction systems receive the same document and schema without a benchmark-specific prompt. LlamaExtract Cost-Effective and Agentic use their matching parse tiers, explicit parse-first execution, source citations, and word-level bounding boxes; Agentic Plus uses per-document extraction with source citations and confidence scores. Reducto Deep Extract runs with citations and deep extraction enabled. Extend Max Context uses the `extraction_performance` processor with citations, advanced figure parsing, and `large_array_max_context`. Datalab uses Accurate Parse, Balanced Extract, and JSON output.

C.2 Cost Accounting and Pricing

This appendix records the list prices used to reconstruct commercial-system costs in Section 3.2. We apply the public pay-as-you-go or standard-tier rates available as of July 1, 2026 to recorded or reconstructed token, credit, or page usage. Volume, committed-use, and enterprise discounts are not included.

Open-weight pipelines that we self-host (Lift 9B [7], NuExtract3 [27], Qwen3.6 35B-A3B [34], and Gemma4 26B [14]) have no vendor API price and are omitted. The ¢/page figures in Section 3.2 are calculated from the token, credit, or page consumption of the benchmark runs. Because provider pricing can depend on observed usage, these measured costs may differ from a headline per-page rate.

Token-metered systems. Table 10 reports the token rates used for the VLM API and coding-agent runs. Codex GPT-5.5 is priced at the GPT-5.5 rates and Claude Code Opus 4.8 at the Opus 4.8 rates. OpenAI applies a 2× input and 1.5× output surcharge above 272K input tokens [31]. Anthropic's Opus 4.7+ tokenizer emits ∼30% more tokens for the same text [3].

Model	Input	Cached	Output
OpenAI [32]			
GPT-5.4 Nano	0.20	0.02	1.25
GPT-5.5	5.00	0.50	30.00
Google [15]			
Gemini 3.5 Flash	1.50	0.15	9.00
Anthropic [3]			
Opus 4.8	5.00	0.50	25.00

Table 10: Token-metered list prices in USD per one million tokens, using the standard real-time tier. "Cached" denotes cache-hit input.

Managed document extraction APIs. These APIs typically separate document extraction into two stages: (1) parsing (i.e. transcribing) the document into a machine-readable representation, and (2) extracting the target fields from that representation. Table 11 reports the corresponding parse and extract charges.

System	¢/credit	Parse *credits/page*	Extract *credits/page*	¢/page
LlamaExtract [24, 25]				
Cost-Effective	0.125	3	5	1
Agentic	0.125	10	15	3.1
Agentic Plus	0.125	10	50	7.5
Reducto [35, 37]				
Deep Extract	1.5	1–2	$\max(30,\ 4p + 0.1f)$ credits/document	variable
Extend [11]				
Max Context	1.25	2	6	10
Datalab [6, 8]				
A+B	1	1.0	2.5	3.5

Table 11: Page- and credit-metered list prices. Here, p is the number of pages and f the number of returned fields.

The provider-specific pricing rules are:

- *LlamaExtract Agentic Plus.* For large schemas, LlamaIndex applies a schema-size multiplier to the 50-credit *extract* rate [25].
- *Reducto Deep Extract.* Reducto charges $\max(30,\ 4p + 0.1f)$ extract credits per document, plus 1–2 parse credits/page. At the lower parse rate, a one-page document costs at least 31 credits, or 46.5 ¢.
- *Extend Max Context.* Extend's published base rate is 3 extract credits/page plus 2 parse credits/page [11]. Its `large_array_max_context` strategy makes multiple passes at approximately twice the extract credits [10]; accordingly, we double the extract component and leave the parse component unchanged.
- *Datalab A+B.* Datalab prices parse and extract directly in cents/page. To present both components in the same columns, we treat 1 ¢ as one credit.

D Detailed Results

Using the same document-level aggregation and cost accounting as the main text, this section extends the headline results in three directions. Section D.1 examines quality, cost, precision, and recall across document lengths; Section D.2 isolates fine-grained task failures; and Section D.3 compares quality–cost scaling within commercial model families.

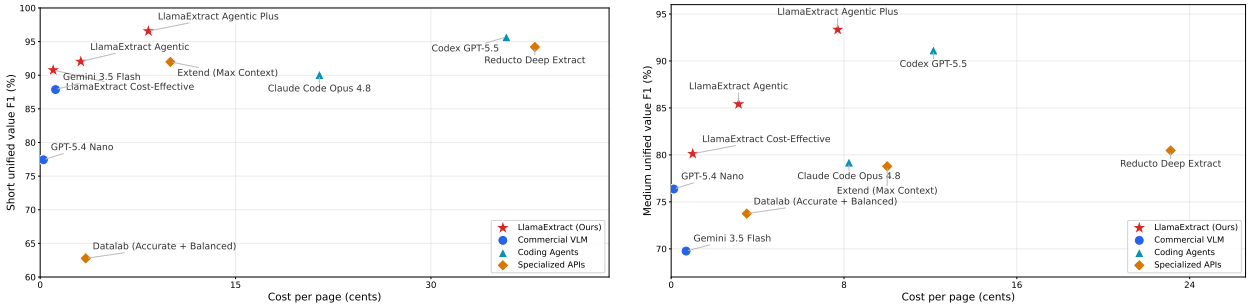
D.1 Document-Length Analysis

Quality and Cost. Figure 7 decomposes the overall quality–cost tradeoff of Section 3.2 by document length. Each panel pairs unified value F1 with mean per-page cost over the documents in that length slice. Table 12 provides the exact costs.

On short and medium documents, the same progression defines the frontier: GPT-5.4 Nano anchors the lowest-cost end, followed by LlamaExtract Cost-Effective, Agentic, and Agentic Plus as quality and cost increase. Agentic Plus remains the high-quality endpoint while costing less than the coding agents and the most expensive specialized APIs.

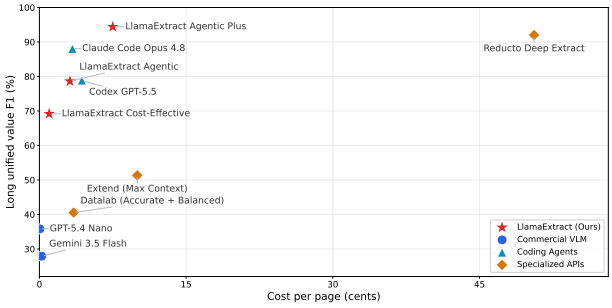
Long documents reshape the tradeoff. The token-metered VLMs and coding agents cost substantially less per page on L3 than on L1, consistent with per-document overhead being spread across more pages. Their quality does not scale uniformly, however: the one-shot VLMs degrade sharply, and Codex also loses ground. Claude preserves more of its short-document quality and becomes a competitive intermediate point, while Agentic Plus continues to anchor the high-quality end. Reducto also remains accurate on long documents, but at a substantially higher per-page cost.

Together, the panels separate two notions of scaling. A system can become cheaper per page as documents grow while extracting a smaller fraction of the requested records; robust long-document extraction requires both favorable cost scaling and stable quality.



(a) Short documents (L1)

(b) Medium documents (L2)



(c) Long documents (L3)

Figure 7: Unified value F1 versus measured per-page cost by document length (L1/L2/L3). Systems without a measured cost on a slice are omitted. The four OSS pipelines are open-weight and self-hosted.

System	Overall	L1: short	L2: medium	L3: long
Specialized APIs				
LlamaExtract Agentic Plus	8.1	8.3	7.7	7.5
LlamaExtract Agentic	3.1	3.1	3.1	3.1
LlamaExtract Cost-Effective	1.0	1.0	1.0	1.0
Datalab (Accurate + Balanced)	3.5	3.5	3.5	3.5
Extend (Max Context)	10.0	10.0	10.0	10.0
Reducto Deep Extract	34.4	38.0	23.1	50.6
Coding Agents				
Codex 5.5	27.8	35.8	12.1	4.3
Claude Code (Opus 4.8)	16.2	21.4	8.2	3.4
Commercial VLM				
GPT-5.4 Nano	0.21	0.25	0.12	0.05
Gemini 3.5 Flash	1.0	1.2	0.69	0.24

Table 12: Mean document-level cost per page (¢/page), overall and by document length. Each column pools the documents in that slice; Overall pools all scored documents. Flat page-priced systems repeat their list price, while token- and credit-metered systems use recorded or reconstructed consumption (Section C.2). The four self-hosted OSS pipelines have no comparable vendor price and are omitted.

Precision and Recall. Precision and recall expose different failures hidden by F1: missing records lower recall, while duplicated or hallucinated records lower precision. Table 13 reports both and the signed gap $\Delta = P - R$ by document length. The large positive L3 gaps for the commercial VLMs show that returned values are often correct but many requested records are missing.

	Overall			L1			L2			L3		
System	P	R	Δ	P	R	Δ	P	R	Δ	P	R	Δ
Specialized APIs												
LlamaExtract Agentic Plus	**95.8**	**95.4**	0.4	**96.6**	**96.5**	0.1	**94.0**	**92.8**	1.2	**94.3**	**94.5**	-0.2
LlamaExtract Agentic	90.4	89.7	0.7	91.4	92.8	-1.4	89.4	84.3	5.1	82.5	77.2	5.3
LlamaExtract Cost-Effective	89.2	86.1	3.1	91.2	90.6	0.6	85.5	79.2	6.3	80.7	63.4	17.3
Datalab (Accurate + Balanced)	64.7	64.5	0.2	63.1	62.9	0.2	73.9	73.6	0.3	40.5	40.5	0.0
Extend (Max Context)	86.0	86.8	-0.8	92.0	91.9	0.1	77.4	80.9	-3.5	51.2	51.5	-0.3
Reducto Deep Extract	90.5	90.5	0.0	94.1	94.4	-0.3	80.8	80.2	0.6	92.0	92.1	-0.1
Coding Agents												
Codex 5.5	95.3	93.2	2.1	96.0	95.5	0.5	93.6	90.2	3.4	94.0	78.6	15.4
Claude Code (Opus 4.8)	87.4	87.1	0.3	90.2	90.2	0.0	79.8	78.9	0.9	89.4	87.5	1.9
OSS												
Gemma4 26B	67.0	66.0	1.0	81.2	80.2	1.0	41.6	40.6	1.0	12.5	11.9	0.6
Qwen3.6 35B	88.9	87.2	1.7	93.2	93.3	-0.1	85.5	84.3	1.2	51.0	25.7	25.3
NuExtract3	64.2	45.0	19.2	66.5	51.0	15.5	61.2	37.4	23.8	50.1	5.8	44.3
Lift 9B	78.8	77.2	1.6	87.4	87.2	0.2	65.1	62.4	2.7	37.4	24.2	13.2
Commercial VLM												
GPT-5.4 Nano	77.8	75.1	2.7	77.4	78.2	-0.8	80.3	75.5	4.8	71.2	33.7	37.5
Gemini 3.5 Flash	84.5	79.5	5.0	88.2	87.7	0.5	75.1	69.3	5.8	83.7	26.5	57.2

Table 13: Value precision (P), recall (R), and signed gap $\Delta = P - R$ (percentage points), overall and by document length. Positive Δ means recall trails precision. Overall weights slices by scored-document count; L1/L2/L3 denote $\leq$10, 11–50, and $>$50 pages. Entries are per-document means, so mean F1 is not implied by mean P and R. Bold and underline mark the top two P and R values; red shading marks gaps above 5/15/25 points (darker means larger).

D.2 Task-Challenge Analysis

Table 14 decomposes the three task-challenge families reported in Table 2 into their individual sub-tags. Because these slices can overlap in documents, the rows should be interpreted individually rather than averaged. The shading highlights where a system falls below its own overall score, making it possible to distinguish broad weaknesses from failures concentrated in a particular subtype.

Challenge sub-tag	Specialized APIs						Coding Agents		OSS				Commercial VLM	
	LE Agentic Plus	*LE Agentic*	*LE Cost-Eff.*	*Datalab A+B*	*Extend Max*	*Reducto Deep*	*Codex GPT-5.5*	*CC Opus 4.8*	*Gemma4 26B*	*Qwen3.6 35B-A3B*	*NuExtract3*	*Lift 9B*	*GPT-5.4 Nano*	*Gemini 3.5 Flash*
Overall	**95.6**	89.5	86.8	64.5	86.3	90.4	93.6	87.1	66.2	87.3	47.9	77.3	74.9	79.8
T1: Long-list completeness														
T1.a: single long table	**97.5**	87.4	83.8	78.0	85.6	96.0	92.4	95.1	47.5	74.8	35.9	66.7	72.6	74.4
T1.b: cross-page continuation	**95.8**	84.3	79.0	78.5	85.1	94.4	89.4	92.5	40.5	73.8	37.6	64.5	72.3	73.6
T1.c: repeated complex region	**93.9**	76.2	75.4	84.2	88.3	92.7	84.1	91.6	55.4	84.3	34.4	72.8	62.2	87.2
T1.d: pivoted / matrix	95.0	86.1	87.2	84.4	91.4	**95.3**	94.9	94.3	63.0	89.6	20.9	76.4	77.7	88.4
T1.e: packed / multi-row cell	**97.2**	87.3	78.2	71.7	75.9	95.4	86.8	93.9	37.1	56.6	50.1	51.3	67.3	53.4
T2: Needle-in-haystack														
T2.a: sparse in narrative	**95.9**	92.9	88.8	79.5	93.7	93.5	94.6	90.2	67.0	92.5	18.1	87.5	83.5	89.9
T2.b: paraphrased match	90.3	87.7	80.7	51.1	84.2	90.8	86.6	85.0	62.7	86.2	44.8	78.9	79.5	**93.0**
T2.c: dedup across modalities	**94.8**	86.4	84.8	86.8	87.6	92.0	90.2	77.8	83.4	82.4	6.6	76.3	81.8	91.0
T2.d: cross-ref / reconciliation	**92.2**	85.5	78.2	70.3	88.2	91.8	89.9	88.5	60.5	80.8	29.6	72.1	68.0	86.6
T3: Dense documents														
T3.a: dense form	**95.4**	92.1	90.3	53.1	84.6	86.5	95.1	81.1	76.0	93.2	59.0	82.3	75.8	78.8
T3.b: receipt / invoice	97.4	92.7	95.0	65.5	96.8	97.2	98.2	**99.2**	89.7	97.1	64.8	92.5	83.8	97.1
T3.c: classify then extract	96.5	90.8	74.9	75.0	93.4	95.5	**97.5**	76.7	52.3	66.2	5.4	68.8	72.7	97.2
T3.d: filer-reviewer separation	85.1	82.2	82.9	50.4	**91.6**	85.1	89.5	0.0	75.8	87.6	74.8	82.2	71.5	0.0

Table 14: Unified value F1 (%) for every challenge sub-tag, with the same system columns as Table 2. Within each row, **bold** and underlined mark the highest and second-highest scores. Red shading marks drops of more than 5/15/25 points from each system's overall score (darker means larger). Sub-tag slices overlap in documents; the task-challenge rows of Table 2 score each challenge's document union instead of averaging these rows.

Long-list completeness. Long-list extraction produces the clearest sustained separation between systems. LlamaExtract Agentic Plus and Reducto Deep Extract remain strong across both regular long tables and more difficult packed or multi-row layouts, whereas several other systems degrade as the row structure becomes less regular. The contrast across the T1 sub-tags suggests that the central difficulty is not simply locating individual values, but preserving record boundaries and associating cells correctly across continuations, repeated regions, and compact layouts.

Needle-in-haystack. The T2 results are less uniform because the sub-tags test different retrieval behaviors. Sparse narrative retrieval is broadly tractable for the strongest systems, while paraphrased matching, cross-modal deduplication, and reconciliation change the system ordering. In particular, Gemini leads on T2.b, whereas LlamaExtract Agentic Plus and Reducto are strongest on T2.c and T2.d. The family-level score therefore conceals meaningful differences between locating a mention, recognizing a paraphrase, and selecting a canonical value.

Dense documents. Dense-document performance likewise depends on the underlying structure. Receipt and invoice extraction is strong across most systems, but classify-then-extract and filer-reviewer separation produce much wider dispersion. These rows show that density alone is not the determining factor: the interaction between document structure and the required extraction operation matters more than the number of visible values.

Pipeline failures on large schemas. The widest gaps in Table 14 are not gradual. On T3.d — the 13 reviewed W-14 filings, whose twin schema asks for the filer value and the reviewer annotation as separate fields — Claude Code Opus 4.8 and Gemini 3.5 Flash both score 0.0 while the other twelve systems all score above 50. The same two systems

return nothing on the Form 1040 bundles, which carry the largest schemas in the benchmark. These are end-to-end failures rather than graded perception or completeness errors: on these schemas the two pipelines return no output, and a missing document scores zero as it does everywhere else in the benchmark. Because both cases also use large schemas, a controlled sweep would be needed to determine whether schema size itself causes the failures.

D.3 Quality–Cost Scaling Within Model Families

We compare nine one-shot commercial VLM configurations: basic, medium, and flagship models from the GPT, Gemini, and Claude families. Every configuration receives the same document and target schema, and overall unified value F1 is pooled over the same 370-document benchmark. Figure 8 plots that quality against measured extraction cost per page; within each family, the line connects the three model tiers in increasing order.

Failure rates. A failed or rejected document remains in the evaluation denominator and contributes zero F1, so the comparison measures end-to-end robustness as well as the quality of successful outputs. GPT and Gemini have scored entries for all 370 documents, whereas every Claude tier rejects the Automotive and K-1 schemas before producing a model response. These 33 rejections (8.9% of the pool) score zero and, because no response is produced, also contribute zero measured extraction cost. Rejection alone does not explain Claude's gap: restricted to the 337 completed documents, Haiku 4.5, Sonnet 4.6, and Opus 4.8 still reach only 32.9, 34.6, and 33.0 F1, respectively.

Quality and cost scaling. Scaling behavior differs sharply by family. GPT is the only family with monotonic quality gains: GPT-5.4 Nano, GPT-5.4 Mini, and GPT-5.5 score 74.9, 85.2, and 88.7 F1 at 0.21, 0.72, and 6.86 ¢ per page. Most of the gain therefore comes from Nano to Mini; moving from Mini to GPT-5.5 costs nearly ten times as much for another 3.5 F1 points. Gemini remains essentially flat as cost rises, moving from 79.6 F1 at 0.17 ¢ per page for Flash Lite to 79.8 at 1.00 ¢ for Flash and 78.2 at 3.83 ¢ for Pro. Claude is similarly non-monotonic: Haiku, Sonnet, and Opus score 29.9, 31.5, and 30.1 F1 as cost rises from 0.80 to 2.40 to 4.89 ¢ per page. Within these nine configurations, the quality–cost frontier is therefore Gemini 3.1 Flash Lite, GPT-5.4 Mini, and GPT-5.5: model tier alone does not predict extraction quality, and the best operating point depends on the desired cost–quality tradeoff.

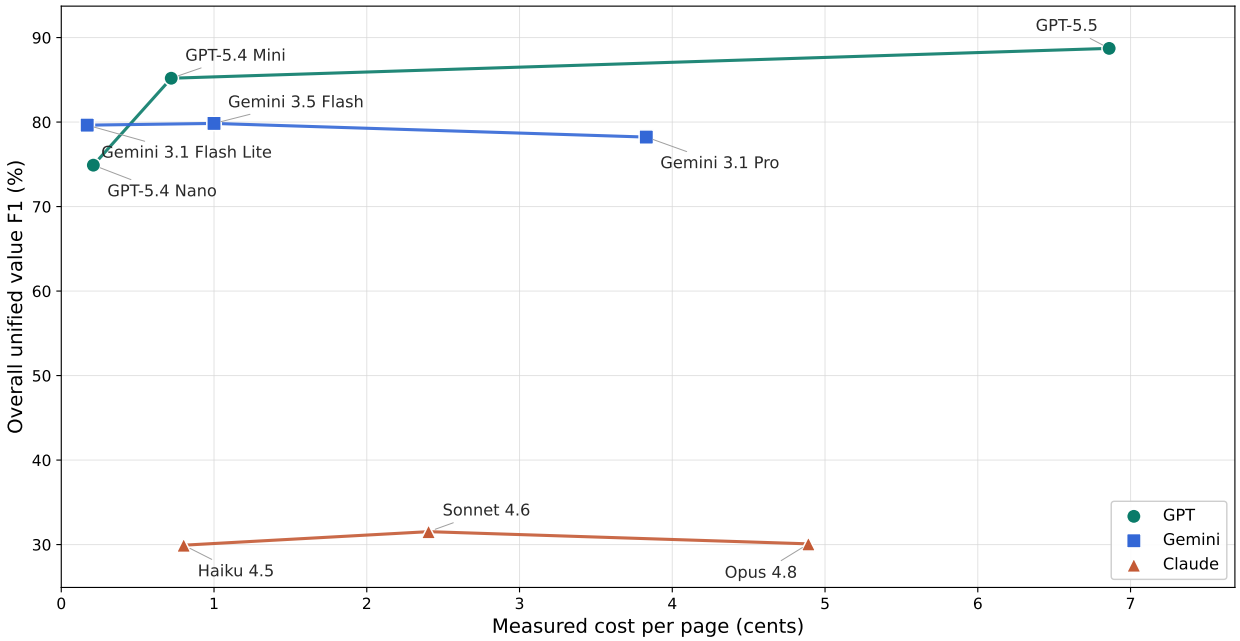


Figure 8: Within-family quality–cost scaling for basic, medium, and flagship models from GPT, Gemini, and Claude. Lines connect the three tiers within each family. Quality is pooled over the full 370-document benchmark; rejected documents score zero, while rejections that produce no model response contribute zero measured extraction cost.