

Silent Structured-Content Corruption in Document Parsing: A Span-Typed Diagnostic and a Released Cross-Parser Resource

Ruiyang Zhang 

FLock.io & Ryonix Labs Inc.

zhangruiyang36@gmail.com

Abstract

Document-AI pipelines increasingly begin by converting a PDF to text with a parser—classical extractors, OCR, layout models, or vision-language models—before any retrieval or reasoning. Overall-quality metrics (edit distance, TEDS) score these parsers as if all characters mattered equally. They do not: a corrupted number, date, identifier, or table cell changes meaning while barely moving a similarity score. We isolate this failure with a span-typed diagnostic over six parsers, ten structured span types, and two conditions (digital and scanned), on a controlled corpus with exact ground truth plus a real arXiv subset. Three findings. (i) Preservation is highly parser- and type-dependent: the best parser preserves 99.4% of spans while others fail specific types catastrophically (Docling drops MARKUP to 6.0%; MarkItDown returns nothing on scanned pages), so “which parser” is a per-span-type risk decision, not a single score. (ii) Document-level text similarity does *not* predict span loss (Spearman $\rho \leq 0.46$ across all parsers and conditions; ≈ 0 for VLMs), so standard metrics are blind to it. (iii) Downstream, corrupted spans are dangerous precisely when the parse *looks* successful: among questions whose evidence span was corrupted but whose document still returned text, 96.5% are answered with a confident wrong value rather than an abstention—an undetectable error—whereas total parse failure is comparatively safe because it forces an abstention. We release the resource behind these numbers: 566,904 per-span verdicts over 660 documents and 7,560 parser outputs, under CC-BY, with an independent verifier (49,720 checks) that re-derives every reported number.

1 Introduction

Retrieval-augmented and document-QA systems rarely see a PDF directly. A *parser* first turns it into text or Markdown—PyMuPDF, Tesseract, layout models such as Docling, or a vision-language

model prompted to transcribe the page—and everything downstream consumes that text. The parser is therefore the point at which document evidence either survives or is silently corrupted before any model reasons over it.

The field evaluates parsers with document-level fidelity metrics: normalized edit distance for text, TEDS (Zhong et al., 2020) for tables. These treat every character as equally important. But in the documents that matter—financial, legal, scientific, clinical—meaning is carried by *structured spans*: numbers, dates, currency amounts, identifiers, URLs, code, markup, quotations, and table cells. Corrupting one of these (a transposed digit, a dropped currency symbol, a mangled ID) changes what the document says while barely denting an edit-distance score, and no downstream component flags it. Existing parser benchmarks (Ouyang et al., 2025; Xu et al., 2026; Li et al., 2025) rank overall quality; a growing line shows OCR quality does not predict downstream RAG performance (Sun et al., 2026). We complement these with a targeted, *span-typed* audit of the specific failure mode—silent, meaning-changing corruption—and release the per-span verdicts so parsers can be audited on this axis.

Contributions. (i) A controlled, span-typed measurement of structured-content preservation across six parsers, ten span types, and digital/scanned conditions, with exact ground truth (§3). (ii) Evidence that document similarity does not predict span loss—metric blindness (§4). (iii) A downstream QA probe showing that the dangerous regime is *partial* corruption behind a successful-looking parse (96.5% confident-wrong), not total failure (§5); plus a corruption taxonomy, an in-table penalty, and honest OCR-resolution and real-PDF controls (§6). (iv) A released CC-BY resource of 566,904 verdicts and a 49,720-check verifier.

type	inj. recall	prose prec.	prose rec.
DATE	1.00 (5408/5408)	1.00	0.96
ID	1.00 (3473/3473)	0.91	1.00
CURRENCY	1.00 (6912/6912)	1.00	1.00

Table 1: Detector validation for the added span types (hand-audited). Recall on injected spans is exact; prose precision/recall are on held-out human-labelled samples.

2 Setup

Corpus and ground truth. We render a controlled corpus of 600 documents from text carrying *known* structured spans, to digital PDF and to a rasterized “scanned” image PDF (so the OCR path is exercised). Ground truth is the exact injected span strings and their loci; frozen detectors (validated below) recover them from parser output. A real subset of 60 public arXiv PDFs provides ecological validity, with spans taken from each paper’s own text layer.

Parsers. Four local, free, CPU-only tools—PyMuPDF4LLM (Artifex Software, 2024), MarkItDown (Microsoft, 2024), Docling (Auer et al., 2024), Tesseract (Smith, 2007)—and two vision-language parsers, GPT-4.1 and GPT-4.1-mini prompted to transcribe each page. Both VLMs get one frozen prompt at temperature 0, and both conditions are downscaled to a common pixel budget so the conditions differ in image *quality* and not in how many pixels the model is handed; the prompt is reproduced verbatim in Appendix C. No GPU was used; total API cost was $\approx$ \$14.

Span types and matcher. Ten types: NUMBER, DATE, CURRENCY, ID, URL, EMAIL, CODE, MARKUP, QUOTE, TABLE_CELL. A span is *preserved* iff it appears intact in the output under a pinned normalization; we also report format-, digit-, and delimiter-lenient levels (Table 3 uses strict). The frozen detectors that recover spans from parser output are hand-validated (Table 1): perfect recall on injected spans and 0.91–1.00 precision on held-out prose, so a missed or spurious span cannot masquerade as parser damage. Appendix A gives the provenance, scoring criterion and leniency sensitivity of all ten detectors—including why TABLE_CELL needs no detector at all.

Discipline and cost. The design was pre-registered (thresholds and decision rules fixed before results); deviations are logged. `verify.py`

parser	cond.	docs	USD
GPT-4.1	digital	600	4.75
GPT-4.1	scanned	600	4.78
GPT-4.1-mini	digital	600	1.58
GPT-4.1-mini	scanned	600	1.59
VLM real-PDF subset (60 each)		120	1.42

Table 2: API cost. Local parsers (PyMuPDF4LLM, MarkItDown, Docling, Tesseract) are free; total VLM spend $\approx$ \$14.

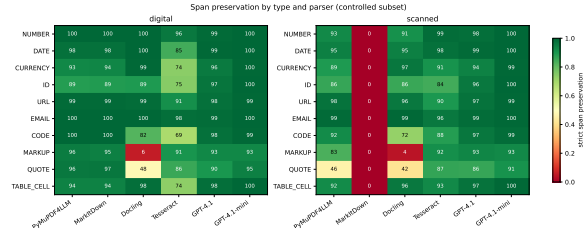


Figure 1: Span preservation by parser $\times$ type $\times$ condition (all ten types). The best parser preserves 99.4%; the damage concentrates in specific parser/type cells, not uniformly.

re-derives every number in this paper from the raw parser outputs and passes all 49,720 checks. The four local parsers are free and CPU-only; the two VLM parsers cost \$14 in total (Table 2), so the whole study is reproducible for the price of a few API dollars and no GPU.

3 Preservation is parser- and type-dependent (H1)

Table 3 and Figure 1 give the core result. Aggregate preservation ranges from 99.4% (GPT-4.1-mini, digital) to 0% (MarkItDown, scanned), but the aggregate hides the point: each parser has *specific* span types it destroys. Docling, a strong layout model on text and tables, drops MARKUP to 6.0% and QUOTE to 48.4%; MarkItDown silently returns nothing on scanned input (it has no OCR path); Tesseract is weakest on ID and CODE. The VLM parsers are strongest overall and most uniform across types. Choosing a parser is thus a decision about *which span types you can afford to lose*, which an all-type quality number cannot express. Damage persists under every matcher-lenient level, each scored against its own denominator: pooled strict preservation is 85.7%; relaxing digit formatting raises the digit-bearing spans to 89.5%; relaxing quotation delimiters raises QUOTE from 71.9% to 78.5%. Leniency can only make a span *easier* to count as preserved, so every rate here

cond.	parser	NUM	DATE	CUR	ID	URL	MAIL	CODE	MRK	QT	CELL	ALL
digital	PyMuPDF4LLM	99.8	97.8	93.2	88.5	99.2	99.8	99.7	95.6	96.4	94.2	95.0
	MarkItDown	99.8	97.9	93.6	88.7	99.2	99.8	99.6	94.9	96.7	94.5	95.2
	Docling	99.7	99.9	99.5	89.1	98.7	99.8	82.1	6.0	48.4	98.2	93.9
	Tesseract	96.4	84.7	74.0	74.7	91.5	98.1	68.8	90.6	85.9	73.6	78.2
	GPT-4.1	99.2	99.5	95.7	97.2	97.9	98.7	97.8	93.4	90.1	98.1	97.6
	GPT-4.1-mini	99.8	99.8	99.5	99.8	99.5	99.8	99.4	93.4	94.7	99.7	99.4
scanned	PyMuPDF4LLM	92.8	94.8	89.2	86.3	97.5	99.2	92.4	83.5	46.2	92.3	90.6
	MarkItDown	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Docling	91.1	94.9	96.6	85.6	96.4	99.3	72.3	4.3	41.5	96.4	90.6
	Tesseract	98.6	97.7	90.9	84.5	89.7	96.2	88.4	91.5	86.6	92.7	92.3
	GPT-4.1	98.4	99.1	94.4	96.3	97.3	98.6	96.8	93.4	86.0	97.4	96.7
	GPT-4.1-mini	99.7	99.7	99.4	99.7	99.3	99.6	98.6	92.8	90.7	99.6	99.1

Table 3: Strict span preservation (%) by parser $\times$ type $\times$ condition, controlled corpus (600 docs). Parsers fail *specific* types catastrophically (bold): Docling on MARKUP/QUOTE; MarkItDown returns nothing on scanned pages (no OCR path). The all-type mean (last column) hides this: choosing a parser is a decision about which span types you can afford to lose.

is a lower bound on damage; the per-type ladder is in Appendix A.

4 Similarity does not predict span loss (H2)

Does a cheap document-level similarity score warn you? We correlate per-document round-trip text similarity with per-document span-preservation rate. Pre-registered rule: $|\rho| \geq 0.6$ would make similarity a usable predictor.

Within a parser and condition—the setting in which a practitioner would actually use such a check—it never comes close. The largest $|\rho|$ over all parser $\times$ condition cells is 0.46 (PyMuPDF4LLM, scanned). For the VLM parsers it is near zero, and worth stating precisely rather than as “ ≈ 0 ”: GPT-4.1 gives $\rho = 0.07$ ($p = 0.11$) and $\rho = 0.00$ ($p = 0.98$), while GPT-4.1-mini gives $\rho = -0.12$ in both conditions at $p < 0.005$ —small, and where statistically reliable, pointing the *wrong way*. Pooled over the controlled corpus, $\rho = 0.39$ for similarity and 0.57 for chrF2 (Popović, 2015).

One configuration does cross the bar, and we report it rather than bury it: pooling all six parsers on the real arXiv subset gives $\rho = 0.61$. That is an artifact of pooling a catastrophic parser with working ones—MarkItDown contributes 60 documents at exactly (0, 0)—and deleting it drops the real pooled figure to 0.33 and the controlled one from 0.39 to 0.19. A correlation pooled across parsers answers “can similarity tell a dead parser from a live one”, which no one needs a metric for. The guardrail question is within a parser, and there the answer is no.

High text similarity coexists with heavy structured-span loss (Fig. 2). Edit-distance/similarity-style metrics are therefore blind to exactly the corruption that changes meaning—consistent with, and mechanistic for, the OCR-quality-vs-downstream gap reported by Sun et al. (2026). The blindness is worst for the VLM parsers, whose outputs are fluent Markdown that scores high on text similarity whether or not a digit was silently changed. A cheap similarity check is not a usable guardrail: it will pass corrupted documents.

5 The dangerous regime is partial corruption (H3)

We attach a QA probe whose answer *is* a structured span, and ask what happens when the parser corrupted the evidence. When the span is preserved, accuracy is 99.1% (3156/3185); when corrupted, 0.2% (1/415). That contrast is near-definitional under exact match; the informative question is the *failure mode*. Of 415 corrupted-span questions, 303 are abstentions (NOT_FOUND, detectable) and 111 are confident wrong values (undetectable). Crucially, restricting to documents that *did* return text—where the parse looked successful and the corruption is partial rather than total—111 of 115 (96.5%) come back as a confident wrong value. Over the item pairs observed in both arms the contrast is paired-significant (McNemar with continuity correction: 299 discordant pairs favouring preserved against 0 favouring corrupted, $\chi^2 = 297.0$, $p = 1.5 \times 10^{-66}$). Total parse failure is comparatively *safe*: an empty document forces an abstention a pipeline can catch. Partial corruption behind

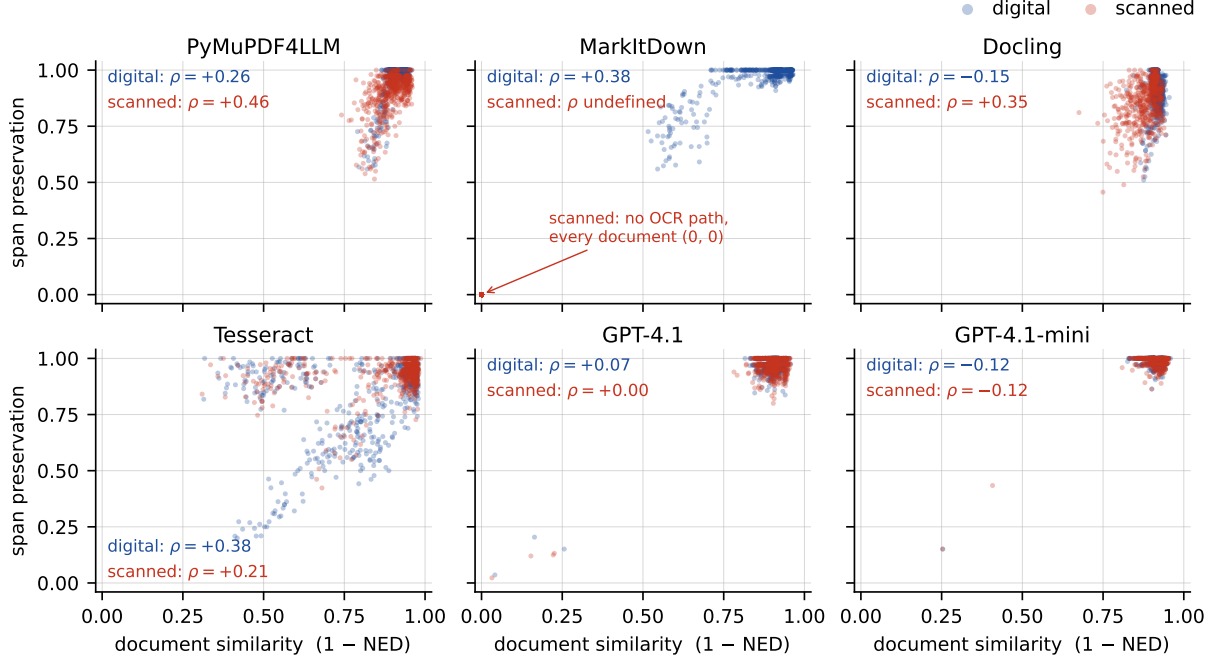


Figure 2: Per-document round-trip text similarity vs. span-preservation rate, per parser and condition. High similarity coexists with heavy span loss, and no parser $\times$ condition cell reaches the pre-registered $|\rho| \geq 0.6$ usability bar (largest 0.46; both VLM parsers within 0.12 of zero, GPT-4.1-mini negative). MarkItDown has no OCR path, so its scanned arm is a single point at the origin and its ρ is undefined.

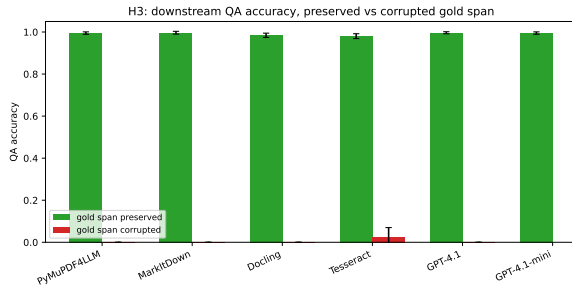


Figure 3: Downstream QA. When the evidence span is preserved, accuracy is 99.1%; when corrupted, 0.2%. The informative split is *how* it fails: on documents that still returned text, 96.5% of corrupted-span questions get a confident wrong answer rather than an abstention.

a successful-looking parse is the mode that silently poisons downstream answers. The reader, prompt, the exact definition of a confident wrong answer, and why no retrieval step is involved are in Appendix B.

6 Taxonomy, tables, and honest controls

How spans break. Table 4 gives the failure taxonomy. Deletion dominates (64.3%), then truncation/split (15.1%); but a non-trivial slice is *value-changing*—digit alteration (3.1%), delimiter substitution (2.3%), entity escaping (2.3%)—the kind

mode	%	mode	%
deletion	64.3	digit alteration	3.1
truncation/split	15.1	delimiter subst.	2.3
other alteration	7.1	entity escaped	2.3
line-break split	5.3	repeat dropped	0.3
		casing / cell-misalign	0.2

Table 4: Corruption taxonomy over all corrupted spans (controlled, all parsers). Loss dominates, but $\sim 8\%$ is value-changing.

that produces *plausible wrong* answers rather than obvious gaps.

Tables cost extra. The same span type is preserved less well inside a table than in prose (Table 5): ID -6.2 , CURRENCY -3.4 , DATE -2.1 points. Table *row integrity*—all cells of a row emitted on one output line—varies sharply by parser (Table 6), from 99.6% (GPT-4.1-mini) to 60.9% (Tesseract, digital) and 0% (MarkItDown, scanned): the parsers that preserve individual cells do not necessarily preserve the row structure that makes them interpretable.

OCR: resolution, not noise. Our two conditions differ in two things at once—digital is a clean 300 dpi render, scanned a degraded 200 dpi one—so a Tesseract subsample fills in the miss-

	NUM	DATE	CUR	ID	MAIL	CODE
prose	89.7	90.3	88.5	86.2	90.7	78.1
table	88.4	88.2	85.1	80.1	90.7	88.4
Δ	-1.3	-2.1	-3.4	-6.2	-0.1	+10.3

Table 5: Preservation (%) of the same span type in prose vs. inside a table. Identifiers suffer most inside tables.

parser	digital	scanned
PyMuPDF4LLM	92.7	87.8
MarkItDown	75.9	0.0
Docling	95.5	90.9
Tesseract	60.9	80.0
GPT-4.1	95.5	94.5
GPT-4.1-mini	99.6	99.3

Table 6: Table row integrity (% of rows whose cells stay on one output line).

ing cell of that 2×2 , clean at 200 dpi, and separates them (Table 7). The difference is almost entirely resolution, and it runs in the *unexpected* direction: moving $300 \rightarrow 200$ dpi *raises* CODE by 20.5 and TABLE_CELL by 20.2 points, whereas adding realistic scan degradation at fixed resolution costs at most 1.8. Tesseract’s better showing on the scanned condition in Table 3 is therefore a rendering-resolution effect, not evidence that degradation is harmless; we report the contrast as a difference in the imaging path as a whole and do not call it “OCR corruption”.

Real PDFs, read carefully. On 60 arXiv papers the parser ordering broadly reproduces (Table 8; MarkItDown 0%, VLMs and PyMuPDF strong on NUMBER). We flag one confound rather than pool it: the DATE column is dominated by arXiv’s submission stamp, printed *rotated* down the left margin; Tesseract’s page segmentation recovers it (95.5%) while layout/VLM parsers discard it (Docling 0%, GPT-4.1-mini 12.1%)—a real behavioral difference, but about rotated marginal text, not date handling, so we report it separately and do not pool it with the controlled corpus.

7 Discussion

The three findings compose into a single risk model. Span loss is (i) *type-specific*—a parser that is excellent overall can still destroy one structured type—(ii) *invisible to similarity metrics*, and (iii) *downstream-confident*: when the parse looks successful, the corrupted value is answered rather than abstained on. A document pipeline built on an aggregate fidelity score therefore has no signal for

type	300dpi	200dpi	degr. 200	res. eff.
CURRENCY	71.6	91.1	89.9	+19.4
CODE	67.8	88.2	88.5	+20.5
TABLE_CELL	72.1	92.3	90.9	+20.2
DATE	85.3	98.4	97.6	+13.1
ID	71.9	81.3	79.8	+9.4

Table 7: Tesseract diagnostic (hardest five types). The $\text{digital} \leftrightarrow \text{scanned}$ difference is a resolution effect—large, and favouring the *lower* resolution—not a degradation effect (≤ 1.8 points, not shown).

parser	NUM	ID	URL	CODE	QT
PyMuPDF4LLM	98.4	97.9	83.9	90.5	16.1
MarkItDown	0.0	0.0	0.0	0.0	0.0
Docling	93.5	93.6	43.7	71.4	12.9
Tesseract	96.4	91.5	77.0	61.9	85.5
GPT-4.1	94.7	89.4	63.2	85.7	77.4
GPT-4.1-mini	94.9	93.6	51.7	81.0	83.9

Table 8: Real arXiv subset (60 PDFs, scanned; DATE excluded, see text). Small, uneven per-type n ; ordering reproduces the controlled corpus.

the errors that most change meaning, and its most dangerous inputs are precisely the ones that pass a superficial quality check.

Which parser for which task? The results give concrete guidance. If the target evidence is *markup or quotation structure* (e.g., parsing code documentation, or preserving nested quotations in legal text), Docling—strong on numbers and tables—is a poor choice, dropping MARKUP to 6%. If inputs are *scanned*, MarkItDown is disqualified outright (it has no OCR path and returns nothing, silently). If *table structure* matters, row integrity, not just cell content, must be checked (Table 6): Tesseract preserves many cells but loses 39% of rows on digital input. The VLM parsers are the most uniform and the strongest on the hard types, but they cost real money (Table 2) and are still the weakest parsers on MARKUP/QUOTE, so they are not a universal answer.

Why the failure is silent. The corruption taxonomy (Table 4) explains the downstream result. Most damage is deletion, which downstream shows up as a missing value—recoverable as an abstention. But the $\sim 8\%$ that is value-changing (digit alteration, delimiter substitution, entity escaping) produces a *plausible* wrong span, and a QA model asked for that value returns it with confidence. The pipeline never sees an error because, at every stage, something was returned: the parser returned text,

the retriever returned a chunk, the reader returned an answer. Nothing is null, so nothing is flagged.

8 Recommendations

(1) **Report span-typed preservation alongside TEDS/edit distance**; the aggregate hides which structured content a parser destroys. (2) **Choose a parser by the span types your task depends on**, not by a single quality score. (3) **Treat a successful-looking parse as the higher-risk case**: add span-level validation (checksums on IDs, format checks on dates/currency) because partial corruption will not raise an error and will be answered confidently.

9 Related Work

Document-parsing benchmarks score overall fidelity across parsers, layouts, and languages (Ouyang et al., 2025; Xu et al., 2026; Li et al., 2025); VLM and OCR parsers proliferate (Auer et al., 2024; Poznanski et al., 2025; Smith, 2007). Sun et al. (2026) show OCR quality does not predict downstream RAG performance. We are complementary: rather than a fidelity leaderboard, we isolate the *silent*, *span-typed* corruption these metrics miss, connect it to a downstream failure mode, and release per-span verdicts. Our resource follows datasheet norms (Geburu et al., 2021).

10 Conclusion

Document parsers do not fail uniformly; they silently corrupt *specific* structured span types, invisibly to text-similarity metrics, and the damage is most dangerous when the parse looks successful—producing confident wrong answers rather than catchable abstentions. Parser choice and evaluation for high-stakes documents should be span-typed. We release the verdicts and verifier so the community can audit parsers on this axis.

Limitations

The controlled corpus is rendered from injected spans, giving exact labels at some cost in realism; the arXiv subset mitigates this but is one genre, at 60 documents, with small and uneven per-type denominators and at least one layout confound we report separately rather than pool. The results should therefore be read as a diagnostic demonstration, not as a settled real-world benchmark: the natural next step, which we have not taken, is the same span-typed audit across the genres where structured

spans actually carry the risk—filings, contracts, forms, invoices, and scientific tables.

We test six parsers, and the two VLM parsers at a single frozen prompt and a single model version. VLM transcription is known to be prompt-sensitive, so the VLM rows should be read as one prompt’s behaviour rather than a property of the model family; we report the prompt (Appendix C) so it can be varied, but we did not run a prompt-sensitivity sweep, and a different prompt could plausibly move the MARKUP and QUOTE columns in particular, which is where both VLMs are weakest.

The QA probe is deliberately narrow: one reader, exact-match scoring, and the whole parser output in context with no retrieval step (Appendix B). Exact match makes the preserved/corrupted accuracy gap near-definitional—so we report the *failure mode* split, which is not implied by the design—and the no-retrieval setting is the most favourable one for the reader, so the confident-wrong rate we measure is if anything a floor for a full RAG stack, which adds retrieval error on top. Ground-truth detectors have high but not perfect precision (reported per type, Appendix A).

Ethics Statement

This work improves the reliability of document pipelines in high-stakes settings (finance, law, health). It uses public documents and released tools; the released resource contains parser outputs and per-span verdicts over public or synthetically rendered documents under CC-BY, with a datasheet.

References

- Artifex Software. 2024. PyMuPDF4LLM: PDF extraction for LLM and RAG environments. <https://pymupdf.readthedocs.io/en/latest/pymupdf4llm/>.
- Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Nikolaos Livathinos, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Fabian Lindlbauer, Kasper Dinkla, et al. 2024. Docling technical report. *arXiv preprint arXiv:2408.09869*.
- Timnit Geburu, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. 2021. *Datasheets for datasets*. *Communications of the ACM*, 64(12):86–92.
- Zichao Li, Aizier Abulaiti, Yaojie Lu, Xuanang Chen, Jia Zheng, Hongyu Lin, Xianpei Han, Shanshan Jiang, Bin Dong, and Le Sun. 2025. *READoc: A*

unified benchmark for realistic document structured extraction. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 21889–21905, Vienna, Austria. Association for Computational Linguistics.

Microsoft. 2024. MarkItDown: A python tool for converting files to Markdown. <https://github.com/microsoft/markitdown>.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, et al. 2025. OmniDocBench: Benchmarking diverse PDF document parsing with comprehensive annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24838–24848.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395.

Jake Poznanski, Aman Rangapur, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, Christopher Wilhelm, Kyle Lo, and Luca Soldaini. 2025. olmOCR: Unlocking trillions of tokens in PDFs with vision language models. *arXiv preprint arXiv:2502.18443*.

Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, pages 629–633.

Lin Sun, Dexian Wang, Jingang Huang, Linglin Zhang, Change Jia, Zhengwei Cheng, and Xiangzheng Zhang. 2026. When good OCR is not enough: Benchmarking OCR robustness for retrieval-augmented generation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 884–894, San Diego, California, USA. Association for Computational Linguistics.

Long Xu, Binghong Wu, Tinghao Yu, Hao Feng, Zhenyu Huang, Haoqing Jiang, Yunhao Wang, Shuo Huang, and Feng Zhang. 2026. MORE: A multilingual document parsing benchmark and evaluation. In *International Conference on Machine Learning (ICML)*.

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. 2020. Image-based table recognition: Data, model, and evaluation. In *European Conference on Computer Vision (ECCV)*, pages 564–580.

A Span detectors: provenance, validation, and leniency sensitivity

Reviewers asked for detector validation beyond the three types in Table 1. Table 9 gives all ten. Three things separate them, and they are not validated the same way.

Six inherited, three added, one constructed.

URL, EMAIL, NUMBER, CODE, MARKUP and QUOTE are inherited unchanged from the prior study this matcher comes from, and are frozen here by hash (FREEZE.json) so the two studies’ numbers stay comparable; we did not re-adjudicate their prose precision, and we do not claim new validation for them. DATE, ID and CURRENCY are added by this study and hand-adjudicated here (Table 1): recall on injected spans is *computed, not sampled* — the generators record every structured record they write, giving 5,408/5,408, 3,473/3,473 and 6,912/6,912 — and precision is adjudicated on held-out natural prose at 1.00, 0.91 and 1.00. TABLE_CELL is the exception that needs no detector: cells are read from the authored table geometry, so a cell is ground truth by construction and detector error cannot apply to it. It exists only in the controlled subset for that reason.

Scoring criterion. Nine types are scored *verbatim*: a document parser is not licensed to re-render anything, so a parser that turns March 5, 2026 into 2026-03-05 has changed the document. QUOTE is scored on delimiter integrity instead, inherited unchanged so the figure stays comparable with the prior study.

Leniency sensitivity. The last two columns of Table 9 give the digit-lenient rate beside the strict one. Read them with their denominators: the digit-lenient level is defined only over the digit-bearing spans of each type, so for MARKUP and QUOTE it is computed over a small and easier subset (1,932 and 3,240 spans) and the apparent +20-point gap is mostly subset selection, not leniency. The one clean same-denominator comparison is QUOTE under delimiter leniency, over all 14,364 spans: 71.9% strict against 78.5% lenient. Every normalisation in the matcher can only make a span easier to count as preserved, so all preservation rates in this paper are lower bounds on damage.

B QA probe protocol

Reader and context. The reader is meta-llama/Llama-3.3-70B-Instruct-Turbo at temperature 0, 64 max tokens, fixed seed. It is a *text* model and never sees the page — only the parser’s output, so nothing it answers can come from the image.

No retrieval. The entire parser output for the document is placed in context (truncated at 24,000

type	origin	criterion	strict		digit-lenient	
			<i>n</i>	%	<i>n</i>	%
NUMBER	inherited	verbatim	33,720	89.6	33,696	90.3
DATE	added	verbatim	68,688	88.8	68,688	89.8
CURRENCY	added	verbatim	85,392	85.5	85,392	89.5
ID	added	verbatim	45,660	82.5	45,660	88.8
URL	inherited	verbatim	19,872	88.8	15,300	91.4
EMAIL	inherited	verbatim	11,820	90.7	312	91.0
CODE	inherited	verbatim	22,632	83.0	1,680	90.5
MARKUP	inherited	verbatim	12,996	69.9	1,932	90.2
QUOTE	inherited	delimiter	14,364	71.9	3,240	90.1
TABLE_CELL	constructed	verbatim	243,120	86.4	168,504	89.2

Table 9: All ten span detectors: where each comes from, how it is scored, and how sensitive its preservation rate is to matcher leniency (controlled corpus, all parsers and conditions pooled). The digit-lenient columns carry their own *n* because that level is defined only over digit-bearing spans.

characters, above our page sizes), with no chunking, embedding or ranking step. This is deliberate: it isolates parser corruption from retrieval error, and it is the *most favourable* setting for the reader, because whenever a span survived the parse it is guaranteed to be in context. A deployed RAG stack adds retrieval error on top, so the confident-wrong rate we report is a floor rather than an estimate for such a stack.

System prompt (frozen, verbatim).

You answer questions about a document using ONLY the extracted text provided. Reply with the exact value copied from the text, and nothing else. If the value is not present, reply exactly NOT_FOUND.

Questions and ground truth. Each question asks for a value that *is* a structured span, addressed by the label printed in the document (e.g. “What is the account IBAN?”). The labels and values are authored by the corpus builder, so the gold answer is ground truth by construction rather than model output. 3,600 items in total.

What counts as a confident wrong answer. Every corrupted-span item falls into exactly one of three buckets: it *abstained* if the raw answer contains NOT_FOUND; it *recovered* if the answer is correct despite the span being damaged, which measures whether the document carried enough redundancy; otherwise the reader returned a value and the value was wrong — a *confident wrong answer*, the case a downstream pipeline cannot detect. Pooled over all 415 corrupted-span items: 303

abstentions, 1 recovery, 111 confident wrong. Restricted to documents where the parser returned some text — where the parse looked successful — the same split is 3 / 1 / 111 of 115, i.e. 96.5% confident wrong.

Paired test. Over the (document, question) pairs observed in both the preserved and corrupted arms, McNemar with continuity correction gives 299 discordant pairs favouring preserved against 0 favouring corrupted, $\chi^2 = 297.0$, $p = 1.5e - 66$.

C VLM transcription prompt and image normalisation

Both vision-language parsers receive one frozen prompt, identical across models and conditions, at temperature 0 with a 4096-token cap:

Transcribe this document page into Markdown exactly as it appears. Preserve every number, date, identifier, currency amount, URL and email address character for character, including separators and formatting. Reproduce tables as Markdown tables with one row per line. Output only the transcription, with no commentary.

Image normalisation. The rendered pages are 300 dpi (digital) and 200 dpi (scanned). Both are downscaled to the same pixel budget — longest side 1800 px, JPEG quality 88 — before being sent, so the two conditions differ in image *quality*, which is the manipulation, and not in how many pixels the model is handed, which would be a confound.

What this does and does not establish. These are one prompt’s results. We report the prompt

so it can be varied, but we did not run a prompt-sensitivity sweep, and the VLM rows of Table 3 should be read accordingly — as the behaviour of these models under this instruction, not as a property of the model family.