ParseBench-2: Severity-Aware Evaluation of Document Parsing

Pierre-Loïc Doulcet Boyang Zhang Sebastián G. Acosta Sacha Bron
Daniel B. Ospina Antonio Jimeno Yepes Nutchapol Dendumrongsup Simon Suo
{pierre, boyang, sebas, sacha, daniel, antonio, nutchapol, simon}@runllama.ai LlamaIndex

Abstract

Document-parsing metrics can give high scores to outputs that change important facts and penalize changes that preserve meaning. People can often judge these consequences more easily than they can design a metric to measure them. We introduce a method for developing parsing metrics from severity policies that specify how each error type should affect the score. We express these policies through test scenarios that specify how a metric should behave. We then optimize the metric implementations against these fixed expectations, using coding agents to refine comparison rules, error detectors and weights under human review. This makes stated human priorities an explicit objective of metric design. We apply this method in **ParseBench-2**, a comprehensive document-parsing benchmark that combines broad capability coverage with metrics calibrated to error severity. It covers 4,317 pages across six dimensions of parsing quality: text, formatting, tables, charts, visual grounding, and forms. We use detailed annotations and targeted difficult cases to check for changed facts, broken structural relationships, and missing content. We take the lower of a parsing score that measures content recovery and a severity cap that penalizes detected errors, so recovering most of a page cannot hide a serious error. Across six severity test suites, our metrics more closely match the expert judgments expressed by the scenario targets than existing metrics and penalize at most 3.4% of the tested benign variations. We conduct a comprehensive evaluation spanning frontier vision-language models, specialized document parsers and open-weight systems, comparing their overall rankings, performance across parsing capabilities and tradeoffs between accuracy and cost.

November 2026

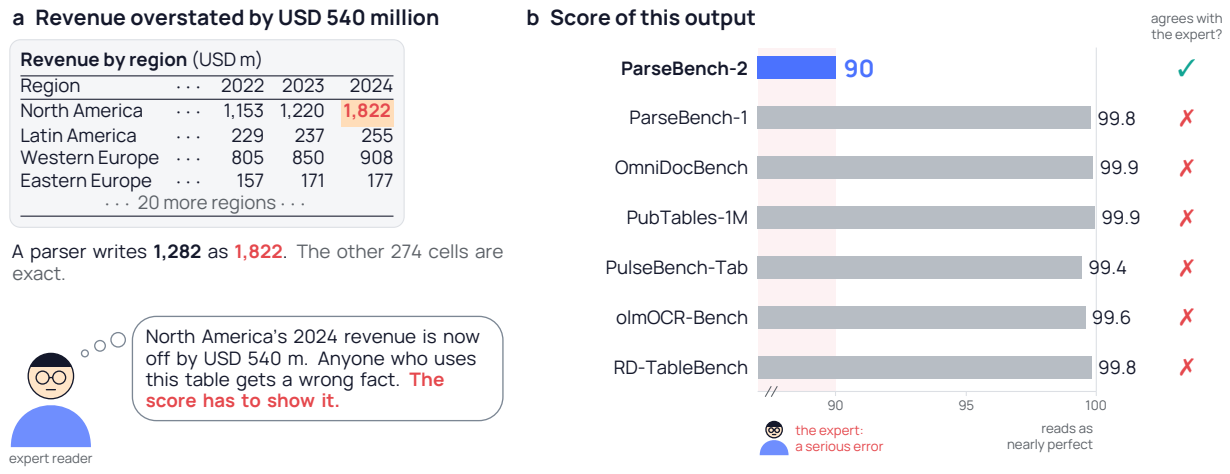


Figure 1 ParseBench-2 aligns parsing scores with human judgments of error severity.

1 Introduction

Document parsers supply the text, tables, and structure that retrieval systems and language-model agents use. Errors in this output can propagate: a changed number in a financial table can become a wrong fact in a generated answer. A useful evaluation should reflect the consequences of these errors while accepting

different representations that preserve the information needed by the application.

Existing metrics do not consistently capture this distinction. Similarity metrics average differences across an output, so a wrong digit can barely affect the score of a long page (Figure 1). Yet these metrics can penalize a table encoded with different HTML tags even when its values and header relationships are unchanged [28]. Rule-based checks make individual requirements explicit. But a pass rate gives every failed check equal weight, so a cosmetic mistake can count as much as a wrong number [16, 24]. These limitations make it difficult to tell whether a high-scoring output preserves the information that matters.

People can often judge the consequence of a parsing error more easily than they can design a metric that measures it. They may agree that writing 1,250 as 1,350 deserves a substantial penalty and that writing it as 1250 should cost nothing. Implementing these expectations consistently across text, tables, charts, and other content requires many interacting rules. We therefore start by defining a policy for how errors should affect the score, then develop metrics that implement it.

We call this a *severity policy*. It specifies how each *error type*, such as changed numbers or missing fields, should affect the score. We use test scenarios to check whether a metric follows this policy [19, 20]. Each test scenario applies one change to a correct output on a particular page. The policy specifies how much lower the changed output should score than the correct output. With these targets fixed, coding agents revise each metric's comparison rules, error detectors and weights. We review the changes and keep those that bring score drops closer to the targets. We then investigate why the metric still fails to meet the targets on other test scenarios. This makes stated priorities an explicit objective of metric design, with the targets fixed and the implementation free to change (Section 2).

We apply this method in **ParseBench-2**, a comprehensive benchmark with scores designed to reflect the impact of parsing errors. Building on ParseBench-1's enterprise document coverage [24], it combines detailed annotations and targeted difficult cases across 4,317 pages. We evaluate six dimensions of parsing quality: text, formatting, tables, charts, visual grounding, and forms. Our severity policies require a single critical error to lower an otherwise correct output's score by at least ten points out of 100, regardless of page length. Benign variations must leave the score unchanged. A high score should require both recovering the page's content and avoiding serious errors. We therefore take the lower of a parsing score, which measures content recovery, and a severity cap, which limits the score according to the severity of detected errors. We report detected errors alongside scores, so users can adjust error penalties for their own applications without rerunning the parsers (Section 3).

We first compare parsing metrics on six test suites to measure how closely they follow our severity policies. Our metrics achieve the highest agreement with the severity policies in all six test suites (Section 4.1). For example, our Formatting metric meets the required score drop in 96% of scenarios with critical errors, compared with 5% for ParseBench-1's formatting metric. We then evaluate 44 configurations of document parsers, direct vision models, and open-weight systems across the six dimensions. The best document parser scores 90.1 overall, compared with 85.7 for the best direct vision model. Our severity-aware evaluation distinguishes parsers that recover similar amounts of content but differ in how often they change facts or lose important information.

Our contributions are:

- A method for constructing parsing metrics from severity policies and test scenarios through iterative refinement by coding agents, with human review (Section 2).
- ParseBench-2, a comprehensive document parsing benchmark that combines broad document coverage and targeted difficult cases with metrics calibrated to the impact of errors (Section 3).
- An evaluation of metric behavior and parsing systems, with error diagnostics, capability comparisons, and cost measurements (Section 4).

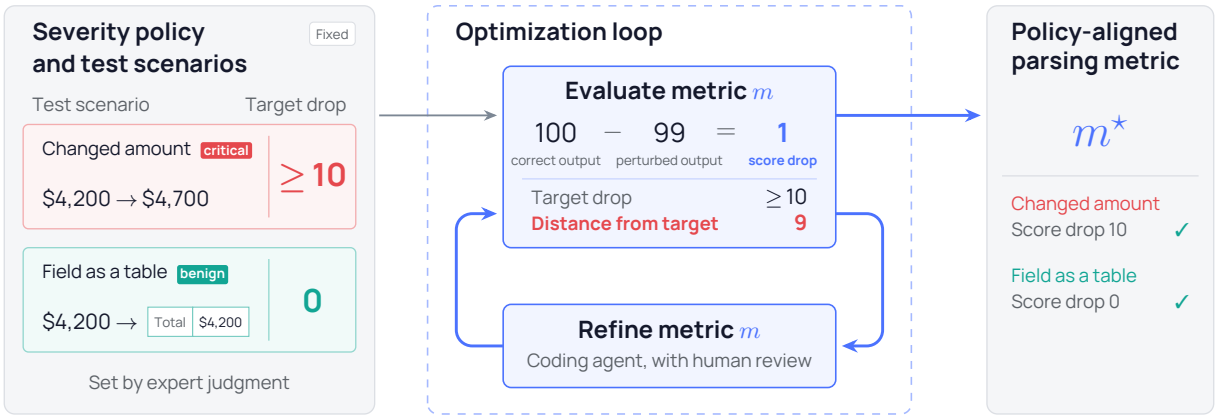


Figure 2 Experts express their severity policy through test scenarios, and we refine parsing metrics to follow that policy (Sections 2.1 to 2.3).

2 Aligning Parsing Metrics with Severity Policies

Specifying how errors should affect the score is easier than designing a metric that meets those expectations. A useful metric must reflect such judgments consistently across many kinds of content and errors. Translating these expectations into scoring rules requires repeated implementation, testing and revision. We therefore treat metric design as an optimization problem: we hold the expected score changes fixed and refine the metric implementation to meet them. A *severity policy* specifies how each error type should affect the score. We record detected errors separately from their penalties so users can change the penalties and recompute scores without repeating error detection. We express these requirements through test scenarios with target score changes (Figure 2).

2.1 Severity policies and test scenarios

Which parsing errors matter depends on how the extracted information will be used. We group parsing errors into *error types*, such as changed numeric values, omitted fields, or incorrect header associations. A *severity policy* π assigns each error type or benign variation t a severity level $\pi(t)$, based on its consequences for the intended application. We use four severity levels: critical, major, minor, and benign. Benign variations preserve the information relevant to the application and receive no penalty. The policy specifies the score change required for each severity level, such as a minimum drop for critical errors and no change for benign variations.

Assigning severity levels. A severity policy is expert judgment, not a measurement. We assigned severity levels based on how each change could affect someone using the parsed content. Users can assign different severity levels for their own applications.

Constructing test scenarios. We make the policy testable with a set of test scenarios $\mathcal{S}$. A *test scenario* $s \in \mathcal{S}$ consists of a page and its correct output, a controlled change to that output (a *perturbation*), and the score change required by the policy. One error type thus yields one scenario per page. Each scenario s has a target score-drop interval $I_\pi(s)$, the range of score drops the policy accepts. The policy specifies a target interval for each severity level. We use the interval for the severity level of the change being tested.

Three principles guide the construction. A perturbation changes one thing, so the score drop measures the metric's response to that change alone. We test whether a metric gives errors too small a penalty or penalizes benign variations that should receive none. A metric's response to the same change can depend on the page's content and length. We therefore apply each perturbation to many pages to check that the metric follows the policy consistently.

An example. Suppose the policy classifies changed amounts as critical and requires a score drop of at least ten points on a 0–100 scale. We construct a test scenario that changes a reported amount from $4,200 to $4,700 on a particular form page. Its target follows directly from the policy: the score must fall by at least ten points. A second scenario presents the same field and value in a two-column table. If the policy treats this as a benign variation, its target is no score change. A target constrains the final score drop, not one parameter of the metric: a single change can be recorded as several errors and also lower the parsing score, so a ten-point target does not mean a ten-point weight.

We next evaluate how well a parsing metric meets these requirements.

2.2 Evaluating parsing metrics under a severity policy

We evaluate a parsing metric by checking whether its score drops meet the severity policy's targets across the test scenarios.

What the metric measures. A parsing metric assigns a score to a parsed output by comparing it with the page's annotations. We write $m(y, r)$ for the score of output y against annotations r, with higher scores indicating better parsing quality.

Measuring a metric's response. For each test scenario s, we score the correct output y_s and perturbed output $\tilde{y}_s$ against the same annotations r_s. The score drop produced by metric m is

$$\Delta m(s) = m(y_s, r_s) - m(\tilde{y}_s, r_s). \tag{1}$$

Distance from the policy target. We measure how far the observed drop falls outside the target score-drop interval $I_\pi(s)$ defined in Section 2.1:

$$d_\pi(m, s) = \operatorname{dist}(\Delta m(s),\, I_\pi(s))\,. \tag{2}$$

The distance is zero when the score drop lies within the target interval. Otherwise, we measure the gap to the nearest boundary. These tests identify where the metric meets or misses the policy's requirements and guide its subsequent refinement.

2.3 Refining metrics through scenario feedback

We use the test scenarios to guide metric development. A coding agent proposes changes to the implementation, and we test each revision against the same policy targets to decide which changes to keep.

Metric implementation. We consider a family of candidate parsing metrics $\mathcal{M}$. Candidates can differ in how they compare outputs, match corresponding content and aggregate discrepancies. These choices determine how a metric responds to the errors and benign variations in the test scenarios. The policy, the test scenarios and their targets are not part of the search and stay fixed. Section 3.2 describes how we combine parsing scores and severity caps in ParseBench-2.

Optimization objective. We minimize the sum of squared distances defined in Section 2.2:

$$m^\star \in \arg\min_{m \in \mathcal{M}} \sum_{s \in \mathcal{S}} d_\pi(m, s)^2. \tag{3}$$

Refinement process. We cannot search $\mathcal{M}$ exhaustively, so we solve the problem approximately by local search. We start with a metric that already follows the policy well (Section 4.1). A coding agent then inspects the scenarios that miss their target, proposes a revision to the metric and evaluates it on the full set of test scenarios. We review each change and keep it when it lowers this sum, breaks no scenario that already met its target, and does not overfit to a particular scenario. We stop when we can no longer find a revision that improves the objective while meeting these conditions. The result follows the policy more closely than the metric we started from.

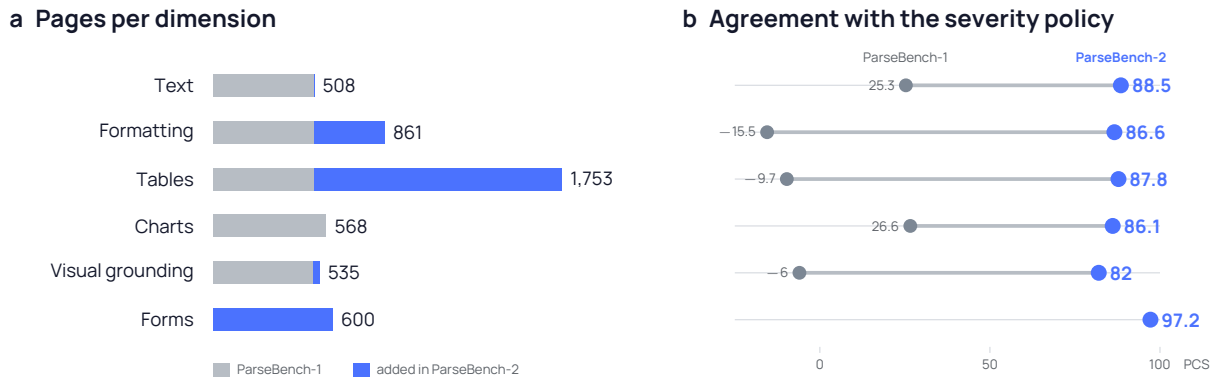


Figure 3 ParseBench-2 covers more pages than ParseBench-1, and its metrics follow the severity policy more closely (Table 1).

An illustrative iteration. One Tables test inserts a decimal point into a numeric cell, such as changing 35 to 3.5. The metric treated changes to numeric labels as rewording and assigned too small a penalty. We revised the comparison rule to recognize the changed number. Across the 45 tested tables, the share meeting the ten-point target rose from 56% to 98%.

3 ParseBench-2: Broad Coverage and Severity-Aware Scoring

ParseBench-2 combines broad coverage of document types and difficult cases with metrics that reflect the severity of parsing errors (Figure 3). We expand ParseBench-1 to 4,317 pages across six dimensions, including cases that test whether parsers preserve table structure, strikethrough and empty form fields (Section 3.1). We also provide more complete annotations, including full data tables for all 1,025 charts.

We use the method in Section 2 to develop metrics from our severity policy (Section 3.2). In preliminary tests, these metrics follow the policy more closely than the compared benchmark metrics in all six suites (Table 1). Our metrics detect errors such as changed values, incorrect header associations, and missing or invented content. We report scores and detected errors for each dimension so users can see what a parser gets wrong and why it loses points.

3.1 Dataset coverage and annotation quality

We evaluate six parsing capabilities as separate dimensions: Text, Formatting, Tables, Charts, Visual grounding and Forms. We group annotated pages into collections, each focused on a parsing capability or a specific difficult case. ParseBench-2 contains 13 collections with 4,317 pages from 2,645 source documents (Appendix A.1).

We use a core collection for each dimension to evaluate how well parsers handle a broad range of documents. We build on ParseBench-1's enterprise document coverage, including scanned, multi-column, multilingual and handwritten pages.

We add targeted collections for Formatting, Tables and Forms to test difficult cases that broad document coverage alone may miss. We add targeted collections of financial statements, insurance rate filings, redlined text and blank public forms.

Expanding dataset coverage. We retain 2,024 of ParseBench-1's 2,078 pages and add 2,293 pages. The new Forms core collection contains 500 government-form pages from 30 countries. We also add 86 pages for Visual grounding, one for Text and 1,706 in targeted collections. We use the targeted collections to check whether parsers attach table values to the correct headers, preserve strikethrough that changes meaning, and retain empty form fields without inventing values.

Improving annotation quality. We revise the annotations of the retained pages so that the metrics can compare complete content. ParseBench-1 checked up to ten data points per chart. We provide a full data table for each of the 1,025 charts so the metric can detect changed or missing values beyond those selected points. Text annotations transcribe the whole page as printed, including running heads, page numbers and formatting. Layout annotations have boxes tight to the ink and a box for every word.

We use frontier models to further refine the annotations, especially for charts and forms. We review and approve the proposed corrections. We check that each page's annotations receive a perfect score when used as both the expected content and the parser output. We also correct any confirmed annotation errors. We manually reviewed each parser's lowest-scoring pages in at least three rounds to identify annotation and scoring errors.

3.2 Severity-aware scoring

Severity policy. The policy reflects our own expert judgment of how an error affects a reader who uses the output. We classify an error as critical when it can mislead a reader, such as a wrong form value. Major errors leave a visible gap, such as a missing value with its caption intact. Minor errors preserve the meaning, such as a typo. Benign variations preserve the information while changing its representation. We require a score drop of at least ten points for a critical error, at least five for a major error, and 1 to 25 for a minor error. Benign variations must leave the score unchanged.

Scenarios. We build six severity test suites, one for each dimension. Together, they contain 330 perturbations covering 240 error types and 90 benign variations (Table 4). They range from swapped column headers and a flipped negation (critical) to extra HTML markup (benign). We list the error types, their severity levels, and the benign variations in Appendix B.1.

Resulting metrics. For each dimension, we provide annotated document pages and a metric for scoring parser outputs. A severity test suite checks whether that metric follows the policy.

We use the method in Section 2 to optimize each metric for agreement with the severity policy, revising its scoring rules as needed. For example, the metrics detect table values linked to the wrong headers, lost strikethrough that changes meaning, and incorrectly extracted chart values. They allow small differences when chart values must be estimated from an axis. The code defines the resulting metrics, and Appendix A.3 provides agent-generated explanations of how they work.

Adding a severity cap. We optimize the parsing metric $m(y, r)$ introduced in Section 2.2 against the severity policy's targets. To enforce the severity policy more strictly, we supplement the parsing metric with a severity cap c for Text, Formatting, Tables, Charts and Forms. The cap limits how high the score can be when serious errors are detected, even if most of the page is correct. The cap is 1 minus the combined penalties for detected errors, with a minimum of zero. The final benchmark score is

$$S = \min(m(y, r), c)\,. \tag{4}$$

3.3 Score aggregation

Dimension scores. We want each dimension to reflect both broad parsing ability and performance on difficult cases. Averaging all pages together would give larger collections more influence on the dimension score.

We therefore compute a separate average for the core collection and for each group of targeted cases (Section 3.1). We then combine these averages using fixed weights, so adding pages to a group does not increase its influence:

$$D \;=\; \sum_{g} w_g\, \bar{S}_g. \tag{5}$$

Here, D is the dimension score, $\bar{S}_g$ is the average page score in group g, and w_g is that group's weight. The weights sum to one.

We choose the weights to balance broad parsing ability with performance on difficult cases (Table 3).

Overall score. We average the six dimension scores, giving each parsing capability equal influence on the overall score. We report the dimension scores alongside the overall score so users can see where a parser is strong or weak.

4 Experiments

We evaluate both the scoring metrics and the parsing systems. We use test scenarios to compare how well the metrics align with the severity policy (Section 4.1). We then evaluate parsing systems on ParseBench-2 to identify their strengths, diagnose serious errors, and compare accuracy and cost (Section 4.2).

4.1 Metric evaluation

We test how closely the metrics developed with our method follow the severity policy in Section 3.2. We compare them with the ParseBench-1 metrics [24] and other baseline metrics on the six severity test suites in Table 4.

Evaluation protocol. We evaluate each metric on the same test scenarios, which cover parsing errors and benign variations. We measure the rate of policy violations at each severity level. We summarize policy agreement with a *policy-compliance summary* (PCS), which starts at 100 and decreases as test scenarios violate the policy:

$$\mathrm{PCS} = 100\,[1 - (0.5\,v_{\mathrm{critical}} + 0.3\,v_{\mathrm{major}} + 0.2\,v_{\mathrm{minor}} + v_{\mathrm{benign}})]\,. \tag{6}$$

Here, v_ℓ is the fraction of scenarios that violate the policy at severity level ℓ. For errors, a violation means that the score drop falls outside the required interval (Section 2.1). For benign variations, it means a drop of 0.5 points or more on the 0–100 page-score scale.

We give benign variations the same total weight as the three error levels combined. This prevents a metric from scoring highly simply by penalizing every difference. All metrics are evaluated on the same PCS scale: higher is better, 100 is the ideal value, and scores can be negative.

We evaluate the metrics on test scenarios generated by applying each perturbation to correct outputs from 100 pages in each of the five real-page suites. We evaluate the metrics on 16,059 graded scenarios across these five suites.

To measure each component's contribution, we compare PCS for the parsing metric m, the severity cap c and their combination S.

Results. The ParseBench-2 metrics have the highest PCS among the benchmark metrics in all six suites (Table 1). The full scenario results show that our metrics meet most policy targets at every severity level in each suite (Appendix B.1). For example, the Formatting metric meets the target on 96% of critical cases, while the ParseBench-1 Semantic Formatting metric meets it on 5%.

Without the severity cap, the parsing metrics already achieve higher PCS than all compared baselines on Text, Formatting and Charts. The severity cap makes a large contribution to policy agreement. For Tables, adding the cap raises PCS from 50.0 for the parsing metric alone to 87.8 for the combined score.

The metrics penalize at most 3.4% of tested benign-variation scenarios in any suite. For Tables, our metric penalizes none, compared with 80% for ParseBench-1's GTRM metric.

Additional results: LLM-as-judge. We also evaluate LLM-as-judge as an alternative to our scoring metrics. We use Claude Haiku 5.5 with the PAIR prompt [17], which evaluates how faithfully outputs preserve a page's information. We test it on the error types and benign variations from the five suites. The judge sees a page and two outputs, then chooses the better output or declares a tie. We use one page per error type and one judgment per comparison.

	ParseBench-2			ParseBench-1 [24]		Best other baseline	
Test suite	Parsing metric m	Severity cap c	Final score S	Score	Metric	Score	Metric
Text	52.2	88.3	**88.5**	25.3	Content Faithfulness	29.5	OmniDocBench 1−NED [12]
Formatting	40.7	86.5	**86.6**	−15.5	Semantic Formatting	−4.7	FinTOC harmonic mean (adapted) [3]
Tables	50.0	87.5	**87.8**	−9.7	GTRM	58.2	PulseBench-Tab T-LAG [13]
Charts	46.4	83.3	**86.1**	26.6	ChartDataPoint Match	30.6	EpiCurveBench [1]
Forms	20.4	97.2	**97.2**	–	–	37.4	Generic exact-match field F1 (ours)
Visual grounding	–	–	**82**	−6	EPR	–	–

Table 1 Policy-compliance summary (PCS; higher is better) for ParseBench-2 and baseline metrics.

The judge usually prefers a correct output over one containing an error. When both outputs contain errors, however, it selects the output with the more severe error in 29% of comparisons. It also often favors one output over another when they differ only by a benign variation. These results motivate defining how errors and benign variations should affect the score, then refining metrics to meet those expectations.

We use the published PAIR prompt as a starting point. Future work could explore model selection and prompt refinement to better align LLM judgments with severity policies.

4.2 Parser evaluation

Evaluation protocol. We evaluate 19 document-parser configurations, 16 frontier VLM configurations and 9 open-weight model configurations. We compare these systems on 4,317 pages across six dimensions and rank them within each group. Appendix C.1 describes the full evaluation protocol.

Table 2 shows the ten highest-scoring systems across all three groups. Appendix C.3 reports results for every system.

Key findings.

- **Document parsers lead overall.** LlamaParse (Agentic Plus) scores 90.1 overall, compared with 85.7 for the best frontier VLM and 77.1 for the best open-weight model.
- **Strengths differ across dimensions.** Frontier VLMs have higher median scores on Text and Tables, while document parsers have higher medians on Charts and Visual grounding.
- **Charts reveal large capability gaps.** Scores range from 1.2 to 92.6. Several default configurations describe charts without extracting their data.

Cost and accuracy Costs vary 30-fold among document parsers, but higher-priced systems do not consistently score better (Figure 4). The highest-scoring parser costs less than half as much per page as the best frontier VLM. Frontier VLMs also offer competitive choices at the low-cost end. Among parsers on the cost–accuracy frontier, the first step above the cheapest option gives the largest score gain. Each later step costs two to four times as much for a few additional points.

Revealing severe errors in parser outputs. Severity-aligned evaluation distinguishes outputs that existing metrics score alike. On one text page, Mistral OCR 4.1 (annotation) invents a percentage table absent

System	Type	Overall	Scores by dimension						Cost
			Text	Format.	Tables	Charts	Visual grounding	Forms	$/page
LlamaParse (Agentic Plus)	Parser	**90.1**	90.3	**93.6**	**94.6**	**92.6**	87.5	**81.8**	0.028
LlamaParse (Agentic)	Parser	86.0	87.0	92.7	89.9	77.4	88.3	80.9	0.013
Claude Opus 5.5	Frontier VLM	85.7	**92.7**	89.4	94.1	74.1	86.2	77.8	0.065
LlamaParse (Cost Effective)	Parser	84.6	85.5	91.9	87.2	73.1	**88.6**	81.2	0.004
Gemini 4 argon (low)	Frontier VLM	84.2	89.7	89.2	92.8	68.2	88.1	77.2	0.027
Reducto	Parser	83.9	81.9	82.8	84.9	89.1	84.1	80.9	0.030
Extend (Parse 2)	Parser	83.2	80.2	83.3	84.2	85.1	84.8	81.5	0.025
Pulse (Ultra 2)	Parser	82.7	86.1	88.7	84.7	86.8	73.5	76.4	0.015
Extend (Parse Light)	Parser	81.6	78.3	78.7	81.4	85.4	84.7	81.2	0.006
Datalab (Accurate)	Parser	81.3	91.1	92.5	90.3	67.8	70.1	75.8	0.010

Table 2 Ten highest-scoring systems on ParseBench-2 using default configurations. Bold marks the best score in each column across all ranked systems.

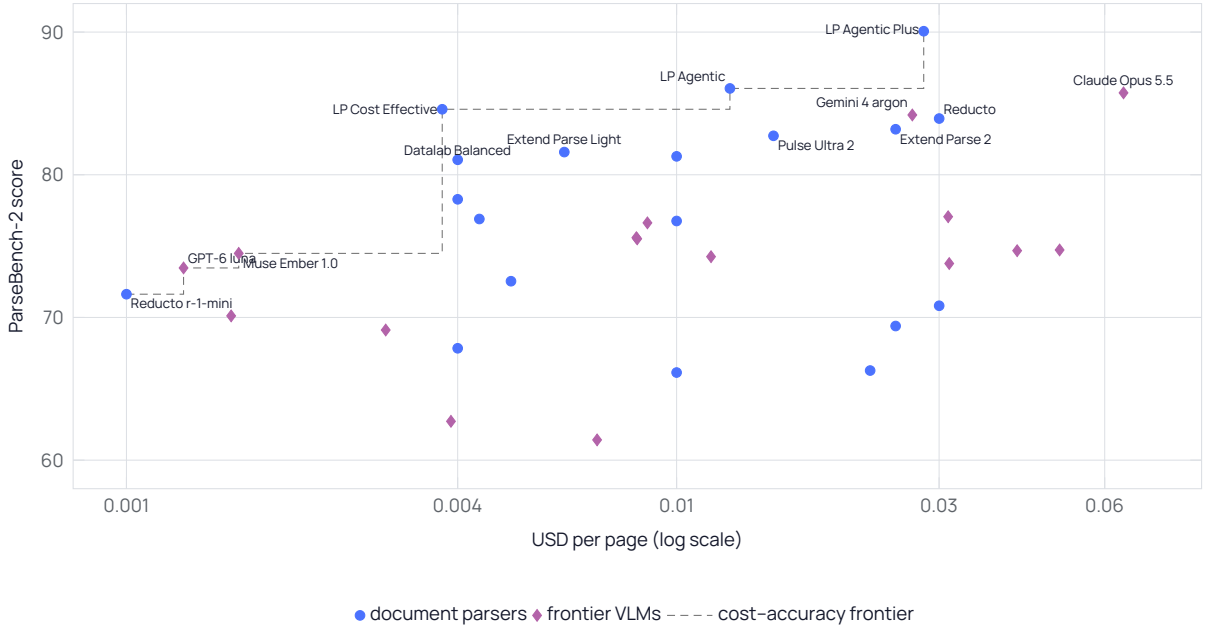


Figure 4 ParseBench-2 score versus average cost per page for 19 document parsers and 16 frontier VLMs. The dashed line shows the highest score achieved within each cost budget. LP denotes LlamaParse.

from the document, while LlamaParse (Agentic Plus) preserves the annotated content. Content Faithfulness assigns both outputs 98.6. ParseBench-2 identifies the invented content and scores the outputs 58.7 and 100, respectively (Figure 5). Appendix C.2 reports error counts and severity-cap results for every system.

a Source page


FLYER

Ikonologie der „Flüchtlingskrise“ 2015/16

10. September 2015

Selfie eines syrischen Flüchtlings mit Angela Merkel beim Besuch der Bundeskanzlerin in der Erstaufnahmeeinrichtung Berlin-Spandau; Foto: Bernd von Jutrczenka/ dpa.

Die vielfach reproduzierte Aufnahme dokumentiert die Beliebtheit und Dankbarkeit gegenüber Angela Merkel infolge ihrer Entscheidung vom 04. September 2015, die in Budapest festsitzenden und von der ungarischen Regierung sich selbst überlassenen Flüchtlinge in Deutschland aufzunehmen.

Ironisch ist das Foto aber auch in Bezug auf die oft kommentierte, nicht selten durch Gehässigkeiten über den Wohlstand der Asylbewerber verzerrt wahrgenommene weite Verbreitung von Smartphones unter den Flüchtlingen.

Zu der überlebenswichtigen Kommunikation und Abstimmung mittels Smartphone über große Räume hinweg gehören auch diese bei den Asylsuchenden sehr begehrten und das Ende der gefahrvollen Flucht besonders wirksam dokumentierenden Selfies mit Angela Merkel.

b LlamaParse

Agentic Plus

FLYER

Ikonologie der „Flüchtlingskrise“ 2015/16

10. September 2015

Selfie eines syrischen Flüchtlings mit Angela Merkel beim Besuch der Bundeskanzlerin
. . .

Die vielfach reproduzierte Aufnahme dokumentiert die Beliebtheit und Dankbarkeit gegenüber Angela Merkel . . .

c Mistral OCR 4.1

annotation

FLYER

INVENTED CONTENT

Category	Percentage
Category A	50%
Category B	50%

Ikonologie der „Flüchtlingskrise“ 2015/16

10. September 2015

Selfie eines syrischen Flüchtlings mit Angela Merkel beim Besuch der Bundeskanzlerin
. . .

Content Faithfulness	98.6	98.6
ParseBench-2 Text	**100**	**58.7**

Figure 5 Content Faithfulness gives both outputs the same score despite the invented percentage table in Mistral's output. ParseBench-2 identifies the error. Output excerpts are rendered with image placeholders omitted and the invented content highlighted.

5 Related Work

5.1 Benchmarks for document parsing

Subtask benchmarks. Early benchmarks each target one parsing capability on a narrow set of documents. PubTabNet [28] and FinTabNet [26] evaluate table structure recognition on academic and financial documents, DocLayNet [15] evaluates layout detection but not content extraction, and ChartQA [11] tests chart reasoning through question answering rather than extraction of the plotted data.

End-to-end benchmarks. OmniDocBench [12], OCRBench v2 [5] and READoc [9] evaluate several capabilities together, mostly on academic and general documents, and score structured elements mainly with similarity metrics. olmOCR-Bench [16] uses binary property tests rather than whole-output similarity, drawing many of its pages from arXiv papers. ParseBench-1 [24] evaluates five capabilities on enterprise documents and checks charts at selected values.

Gaps addressed by ParseBench-2. Most of these benchmarks either focus on narrow document types or check only selected content, so many errors on real enterprise pages go untested. ParseBench-2 closes this gap by extending ParseBench-1's enterprise coverage with forms, revised annotations, complete chart data tables and targeted hard cases (Section 3).

5.2 Evaluating parse quality

Similarity metrics. Edit distance and BLEU [14] compare text, TEDS [28] compares table trees, and GriTS [22] compares table grids. They give partial credit, but a changed digit adds little over a long page, while equivalent markup can cost points.

Rule-based tests. Rule-based benchmarks such as olmOCR-Bench and ParseBench-1 [16, 24] make each requirement an explicit check, but a pass rate counts every check alike, and a strict rule can fail an output that gives the same content in another valid format.

Model-based judges. LLM judges trade hand-written rules for model judgment. They accept equivalent formats and reworded content without a rule for each case, and Horn and Keuper [7] report stronger agreement with human judgments than TEDS and GriTS for tables. This flexibility comes at a cost: strong judges approach human agreement, yet still show position and verbosity bias [25], so consistent and interpretable scores need a capable model and a detailed rubric that defines each error. PAIR [17] ranks two parses with an LLM judge against a shared baseline, so its win rates depend on the judge and the baseline.

Gaps addressed by ParseBench-2. None of these approaches states how much each error should cost. ParseBench-2 instead scores parsing errors under an explicit severity policy (Section 3.2).

5.3 Evaluating and designing metrics

Prior work. In machine translation, MQM-based evaluation has experts mark each error with a severity and weights the counts [4], while learned metrics such as BLEURT [21] and COMET [18] predict human ratings and leave their priorities implicit. Controlled tests check behavior directly: CheckList [19] organizes tests around capabilities and expected behavior, and Sai et al. [20] apply perturbations to evaluation metrics.

Gaps addressed by ParseBench-2. What these precedents leave open is how document-parsing metrics should respond to errors of different severity. We therefore define severity targets for parsing errors and test metrics against them (Sections 2.1 and 4.1).

6 Conclusion

We introduced a method for developing parsing metrics from explicit severity policies. We express how errors should affect scores through test scenarios, then use coding agents with human review to refine metrics against those expectations. ParseBench-2 applies this method across six parsing dimensions, combining broad document coverage, detailed annotations and targeted difficult cases with metrics designed to reflect error severity.

On the test scenarios used for metric development, our metrics follow the stated policy more closely than the compared alternatives while rarely penalizing benign variations (Section 4.1). Our evaluation reveals differences in parser capabilities and consequential errors that high content-recovery scores can obscure. By making evaluation priorities explicit and reporting the errors behind each score, ParseBench-2 helps users understand parser failures and adjust penalties to reflect what matters in their applications. Future work will test how well these scores reflect errors in downstream applications.

References

[1] Thomas Berkane and Maimuna S. Majumder. EpiCurveBench: Evaluating VLMs on epidemic curve digitization. arXiv preprint arXiv:2605.27195, 2026. URL https://arxiv.org/abs/2605.27195.

[2] William S. Cleveland and Robert McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387): 531–554, 1984.

[3] Ismail El Maarouf, Juyeon Kang, Abderrahim Ait Azzi, Sandra Bellato, Mei Gan, and Mahmoud El-Haj. The financial document structure extraction shared task (FinTOC2021). In *Proceedings of the 3rd Financial Narrative Processing Workshop*, pages 111–119. Association for Computational Linguistics, 2021. URL https://aclanthology.org/2021.fnp-1.21/.

[4] Markus Freitag, George Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. Experts, errors, and context: A large-scale study of human evaluation for machine translation. *Transactions of the Association for Computational Linguistics*, 9:1460–1474, 2021.

[5] Ling Fu, Zhebin Kuang, Jiajun Song, Mingxin Huang, Biao Yang, Yuzhe Li, Linghao Zhu, Qidi Luo, Xinyu Wang, Hao Lu, Zhang Li, Guozhi Tang, Bin Shan, Chunhui Lin, Qi Liu, Binghong Wu, Hao Feng, Hao Liu, Can Huang, Jingqun Tang, Wei Chen, Lianwen Jin, Yuliang Liu, and Xiang Bai. OCRBench v2: An improved benchmark for evaluating large multimodal models on visual text localization and reasoning. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025.

[6] Jeffrey Heer and Michael Bostock. Crowdsourcing graphical perception: Using Mechanical Turk to assess visualization design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 203–212, 2010.

[7] Pius Horn and Janis Keuper. Beyond string matching: Semantic evaluation of PDF table extraction. arXiv preprint arXiv:2603.18652, 2026.

[8] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955.

[9] Zichao Li, Aizier Abulaiti, Yaojie Lu, Xuanang Chen, Jia Zheng, Hongyu Lin, Xianpei Han, Shanshan Jiang, Bin Dong, and Le Sun. READoc: A unified benchmark for realistic document structured extraction. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 21889–21905, 2025.

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

[11] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2263–2279. Association for Computational Linguistics, 2022.

[12] Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. 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, 2025.

[13] Ritvik Pandey, Sid Manchkanti, Mohammed Wazir Adain, Mohammed Hadi, and Dushyanth Sekhar. PulseBench-Tab: A multilingual benchmark for table extraction with graph-based evaluation. arXiv preprint arXiv:2606.07534, 2026. URL https://arxiv.org/abs/2606.07534.

[14] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, 2002.

[15] Birgit Pfitzmann, Christoph Auer, Michele Dolfi, Ahmed S. Nassar, and Peter Staar. DocLayNet: A large human-annotated dataset for document-layout segmentation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3743–3751. ACM, August 2022.

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

[17] Reducto. Pairwise AI Ranking (PAIR): Open parse evaluation. https://reducto.ai/blog/pairwise-ai-ranking-parse-evaluation, 2026. Blog post, November 9, 2026. Accessed: 2026-11-10.

[18] Ricardo Rei, Craig Stewart, Ana C. Farinha, and Alon Lavie. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, 2020.

[19] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4902–4912, 2020.

[20] Ananya B. Sai, Tanay Dixit, Dev Yashpal Sheth, Sreyas Mohan, and Mitesh M. Khapra. Perturbation CheckLists for evaluating NLG evaluation metrics. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7219–7234, 2021.

[21] Thibault Sellam, Dipanjan Das, and Ankur Parikh. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7881–7892, 2020.

[22] Brandon Smock, Rohith Pesala, and Robin Abraham. GriTS: Grid table similarity metric for table structure recognition. In *Document Analysis and Recognition – ICDAR 2023*, volume 14188 of *Lecture Notes in Computer Science*, pages 535–549. Springer, 2023.

[23] Michael J. Wise. String similarity via greedy string tiling and running Karp-Rabin matching. Technical report, Department of Computer Science, University of Sydney, 1993.

[24] Boyang Zhang, Sebastián G. Acosta, Preston Carlson, Sacha Bron, Pierre-Loïc Doulcet, Daniel B. Ospina, and Simon Suo. ParseBench: A document parsing benchmark for AI agents. arXiv preprint arXiv:2604.08538, 2026.

[25] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

[26] Xinyi Zheng, Douglas Burdick, Lucian Popa, Xu Zhong, and Nancy Xin Ru Wang. Global table extractor (GTE): A framework for joint table identification and cell structure recognition using visual context. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 697–706, 2021.

[27] Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. PubLayNet: Largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1015–1022. IEEE, 2019.

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

Appendix Contents

A Benchmark Definition **15**
A.1 Dataset Composition . . . 15
A.2 The Severity Policy . . . 16
A.3 Metric Definitions by Dimension . . . 27
A.3.1 How detected errors lower the cap . . . 27
A.3.2 Text . . . 27
A.3.3 Formatting . . . 28
A.3.4 Tables . . . 30
A.3.5 Charts . . . 31
A.3.6 Visual grounding . . . 32
A.3.7 Forms . . . 34

B Metric Evaluation **36**
B.1 Severity Test Suites: Scenario Scores . . . 36

C Parser Evaluation **48**
C.1 Experimental Setup . . . 48
C.2 Where the Cap Decides, per System . . . 49
C.3 Full Results . . . 50
C.3.1 Opt-in chart and form features . . . 54

A Benchmark Definition

A.1 Dataset Composition

Table 3 lists the collections in the ParseBench-2 dataset, grouped by dimension, with their weight in the dimension score (Equation (5)) and their number of pages. Each page is one test case with its own annotations.

Collection	Role	Weight	Pages
■ TEXT			508 pages
Text	core	1	508
■ FORMATTING			861 pages, 508 shared with Text
Text pages[†]	core	$\frac{3}{4}$	508
Headings	targeted	$\frac{1}{8}$	100
Strikethrough[‡]	targeted	$\frac{1}{16}$	128
Underline[‡]	targeted	$\frac{1}{16}$	125
■ TABLES			1,753 pages
Tables	core	$\frac{1}{2}$	500
Complex table headers	targeted	$\frac{1}{8}$	149
SERFF rate tables	targeted	$\frac{1}{8}$	412
Financial tables	targeted	$\frac{1}{8}$	319
Dense tables	targeted	$\frac{1}{8}$	373
■ CHARTS			568 pages
Charts	core	1	568
■ VISUAL GROUNDING			535 pages
Layout	core	1	535
■ FORMS			600 pages
Forms	core	$\frac{7}{8}$	500
Blank public forms	targeted	$\frac{1}{8}$	100
Full dataset			**4,317**

Table 3 Collections in the ParseBench-2 dataset, grouped by dimension. Role: the core collection covers a broad range of documents, and each targeted collection tests one difficult case. Weight: share of the dimension score (Equation (5)). Pages: page-level test cases. [†]The Formatting core collection reuses the Text collection's pages, scored with the Formatting metric. These pages are counted once in the dataset total. [‡]Strikethrough and underline share one targeted group of weight $\frac{1}{8}$.

A.2 The Severity Policy

This appendix gives the full severity policy of ParseBench-2 (Section 3.2): the 330 perturbations of the six severity test suites, which are 240 error types and 90 benign variations. Each error type has one *severity level*, which sets how far a page score should drop when the error occurs once on a page (on the 0–100 scale):

- **Critical** (87 error types): the error can mislead a reader who acts on the output, such as a changed amount or a value under the wrong header. The score should drop by at least 10 points.
- **Major** (71): the error loses or misplaces content and leaves a gap a reader can see, such as a missing value or a moved block. The score should drop by at least 5 points.
- **Minor** (58): the meaning survives, as with a typo or a lost bold label. The score should drop by 1 to 25 points.
- **Debatable** (24): reasonable benchmarks could assign different levels. These error types are listed and are not graded.

A *benign variation* (90) is a different way of writing the same information, such as a table in Markdown where the annotations use HTML. It is not an error, and the score should not drop (by less than 0.5 points). The levels are our expert judgment (Section 3.2); another application could assign them differently.

Summary. Table 4 counts the error types of each severity level and the benign variations of each dimension, with one example of each.

	Error types by level, benign variations					
Test suite	**C**	**M**	**m**	**H**	**D**	**Example perturbation (C; M; m; H)**
Text	13	16	5	8	9	negation flipped; words dropped; spelling typos; paragraphs run together
Formatting	5	8	24	15	3	strikethrough lost; insertion underline lost on a redline; bold and italic stripped; `__bold__` for `**bold**`
Tables	27	18	9	5	3	column headers swapped; two rows merged; empty row inserted; extra HTML markup
Charts	22	9	3	14	5	order of magnitude off; series missing; series names abbreviated; units written in cells
Forms	13	13	8	34	3	digit misread in a value; value dropped, caption kept; typo in a text value; fields written as a two-column table
Visual grounding	7	7	9	14	1	biggest paragraph missed; 50 junk boxes; tiny icon missed; every class wrong

Table 4 The six severity test suites [P]. C, M, m: critical, major and minor error types; H: benign variations; D: debatable error types, not graded. Each error type is applied to the correct output of 100 real benchmark pages of its suite; every page that the perturbation changes gives one test scenario, 16,059 graded scenarios in all. Visual grounding: one synthetic page with variants, and fixed predictions instead of a corruption strength. Its targets are score drops in the same bands, although its page score has no cap.

Evaluation coverage. We exclude a scenario if the perturbation leaves the correct output unchanged, for example because the page has no strikethrough to remove. Across the five real-page suites evaluated in Table 1, 10 error types (3 critical, 3 major and 4 minor) and 12 benign variations change none of the sampled pages and remain untested. The evaluation includes 16,059 graded scenarios across these five suites.

Full policy. Tables 5 to 10 list the policy by dimension. Within a level, rows run from the most to the least severe. *Category* is the group of the error in its test suite, and *What it is* describes the perturbation that the test suite applies to a correct output. Appendix B.1 gives the score drop of every metric on each of them.

Table 5 Severity policy for Text: 43 error types (13 critical, 16 major, 5 minor, 9 debatable) and 8 benign variations.

Name	Category	What it is
CRITICAL ERRORS		
Negation flipped	Word errors	"not", "no" and "never" are dropped, or "not" is added after is, are, was, will or can. One or two words change and the meaning inverts.
Output cut off	Missing or extra content	The output stops early, as with a token limit.
Date with day and month swapped	Wrong facts	One date is read with its day and month exchanged (March 4 → April 3, 10 November → 11 October), as when a parser normalizes a date in the wrong locale. Every word still looks plausible.
Paragraphs dropped	Missing or extra content	Whole paragraphs disappear, as when a region is missed by detection.
Two values swapped between clauses	Wrong facts	Two figures in one paragraph trade places (about 62% ... the remaining 38% becomes about 38% ... the remaining 62%). Every number is still on the page, each in the wrong clause.
Wrong digits	Character errors	One digit is changed in some numbers, dates and phone numbers. Text is otherwise perfect, so this is the error that matters most and moves word metrics least.
Hallucinated summary sentence	Missing or extra content	A fluent closing sentence that is not on the page is added after the last paragraph, with a believable figure: "In summary, the report shows a 12% improvement over the previous year."
Invented sentences	Missing or extra content	Plausible sentences that are not on the page are added inside paragraphs.
OCR confusion in an identifier	Wrong facts	Inside one identifier, digits are read as look-alike letters (0 → O, 1 → l): CA3310021 → CA33lOO2l, 01632 → Ol632. The text reads normally; the identifier no longer works.
Read straight across columns	Reading order	The classic layout failure: each output line joins the left column's line with the right column's line at the same height, so sentences from different columns are spliced together.
Unit changed	Wrong facts	Units next to numbers are misread (mg/L → g/L, £3.20 → $3.20, 20 km → 20 m). Numbers are right, the quantities are wrong.
Email address corrupted	Wrong facts	One character changes in each email address (quality@riversidewater.org → quality@riversidewaters.org), so the address no longer works.
Names and places misspelled	Word errors	Proper nouns are changed by one or two letters (Riverside to Riverview, Norco to Norca). Small character change, large factual error.
MAJOR ERRORS		
Real-word substitutions	Word errors	Words are replaced by other valid words, the kind of error a spell checker cannot see: form/from, then/than, of/or, on/in, and content words swapped for other words on the page.
% and currency signs dropped	Wrong facts	Percent and currency signs disappear (62% → 62, £86.50 → 86.50), so readers cannot tell shares from counts or amounts from plain numbers.
OCR character confusions	Character errors	Typical OCR confusions such as rn→m, l→1, O→0, e→c and lost accents.
Words dropped	Character errors	Individual words go missing, as with low-contrast or faint text.
Paragraph repeated	Missing or extra content	A paragraph is emitted two or more times, a common VLM looping failure.
Phrase omitted	Word errors	A run of 3 to 8 consecutive words disappears from some paragraphs, as when a line inside a paragraph is skipped.
Random words inserted	Word errors	Words that are not on the page appear inside sentences, as small hallucinations.
Sentence cut at a column break	Missing or extra content	The last paragraph of the first column stops in the middle of its final sentence, as when a parser loses the lines that continue at the top of the next column. The rest of the page is intact.
Footnote spliced mid-sentence	Reading order	A short closing note (the last paragraph of the page) is moved into the middle of the last sentence of the paragraph before it, on its own line, as when a parser reads a footnote at a page or column break. All words are present.
Paragraphs shuffled	Reading order	Some blocks are moved to random positions, like a layout model that orders regions wrongly.
Adjacent words swapped	Word errors	Pairs of neighbouring words trade places ("water delivered" for "delivered water"). Every word is still present.
Headings dropped	Missing or extra content	Section headings (Testing and treatment, Results at a glance) are left out; the paragraphs under them are intact.
List items' contents swapped	Reading order	In a bulleted list the first two items trade places while the markers stay put: each item is complete, but the list now reads in the wrong order. Only pages with a list change.
One sentence repeated	Missing or extra content	A single sentence is emitted twice in a row inside its paragraph, a small looping failure.

Severity policy for Text, continued

Name	Category	What it is
Table of contents duplicated	Missing or extra content	A block of short entries (here the results table, read like a table of contents) is emitted twice in a row, a looping failure on lists of short lines. Only pages with such a block change.
Sentences swapped in a paragraph	Reading order	In paragraphs with several sentences, the first two sentences trade places. All words are present.
MINOR ERRORS		
Spelling typos	Character errors	Letters inside words are inserted, deleted, doubled, swapped with a neighbour or replaced by a nearby keyboard key ("treatmnet", "chloramnie", "deliverred").
Look-alike characters	Character errors	Some Latin letters are replaced by identical-looking Cyrillic ones (a, e and o). The page looks right but search and copy-paste break.
Date format changed	Formatting only	Dates are written in another order (March 4 → 4 March, 14 October → October 14). Same dates, different notation than the page.
Thousands separators removed	Formatting only	Large numbers lose their separators (312,480 → 312480, 18,906 → 18906). Same values, different notation than the page.
Parentheses dropped	Character errors	Round brackets disappear, including in phone numbers ((951) 351-6170 → 951 351-6170) and asides.
DEBATABLE ERROR TYPES (LISTED, NOT GRADED)		
Accents stripped	Character errors	Diacritics are removed: información becomes informacion, μg becomes ug. Common in ASCII-only OCR engines.
Columns in reverse order	Reading order	Every block is transcribed perfectly, but the right-hand column comes first. Text is intact; only the order of whole columns is wrong.
Header and footer dropped	Page furniture	The running header and footer are left out. ParseBench v2.1 counts them as content; olmOCR-bench requires them to be absent; OmniDocBench ignores them.
Headings in capitals	Formatting only	The title and headings are transcribed in capitals, as when a parser copies a small-caps style.
Line-end hyphenation kept	Formatting only	Line breaks are kept and words split at the end of a line keep their hyphen ("distri-\nbution"). The words themselves are broken in the output.
List items run together	Formatting only	Bulleted items lose their markers and line breaks and are joined into one paragraph. Words are unchanged; the list structure is gone.
Punctuation dropped	Character errors	Commas, periods, semicolons and parentheses go missing. Words and numbers are intact, though decimal points inside numbers are kept.
Table flattened to lines	Formatting only	Each table cell becomes its own line. Every value survives, but the row and column structure is lost.
Words split or joined	Character errors	Word boundaries are wrong: a space disappears between two words or appears inside one ("thedistribution", "distri bution"). Letters are all present.
BENIGN VARIATIONS		
Different Markdown style	Formatting only	Headings become bold lines and list bullets switch from - to *. Same text, different markup.
Extra blank lines	Formatting only	Extra empty lines appear between paragraphs and after headings. Text is identical.
Hard line breaks	Formatting only	Paragraphs keep the line breaks of the printed column. In Markdown this is still one paragraph and the text is identical.
Hyphenated line breaks kept	Formatting only	Up to six long words keep the hyphen of a printed line break, with the break ("distri-\nbution") or as a space ("informa- tion"). Lines are not otherwise rewrapped and the words are the page's words.
Paragraphs run together	Formatting only	Paragraph breaks inside each column are lost, so consecutive paragraphs come out as one block. Words and order are unchanged.
Random emphasis added	Formatting only	Some words are wrapped in bold or italic markers that the page does not have. Text is identical.
Table as HTML	Formatting only	The table is emitted as an HTML table instead of a pipe table. Cell text is identical.
Typographic characters	Formatting only	Straight quotes become curly, hyphens become en dashes, fi becomes a single ligature character and spaces before units become non-breaking.

Table 6 Severity policy for Formatting: 40 error types (5 critical, 8 major, 24 minor, 3 debatable) and 15 benign variations.

Name	Category	What it is
CRITICAL ERRORS		
Strikethrough lost	Strikethrough and scripts	Deleted text loses its strikethrough and reads as live text: a voided clause or an old fee reads as current.
Exponent lost (10^9 → 109)	Strikethrough and scripts	A superscript exponent becomes ordinary digits glued to its base: 10⁵ reads as 105, 10⁻³ as 10-3.
Live text struck through	Strikethrough and scripts	A phrase of live text is struck through: the reader takes it as deleted.
Digit turned into exponent	Strikethrough and scripts	The last digit of a plain number is set as a superscript: 2026 reads as 202^6.

Severity policy for Formatting, continued

Name	Category	What it is
Exponent written as subscript	Strikethrough and scripts	Exponents are emitted as subscripts: 10₅.
MAJOR ERRORS		
All headings turned into body text	Headings	Every heading is plain text: the page has no structure left.
Insertion underline lost (redline)	Strikethrough and scripts	On a redline, underline marks inserted text. Without it the reader cannot tell new terms from old ones.
One heading turned into body text	Headings	A section heading is emitted as an ordinary paragraph.
List items run together	Lists	The items of one list are joined into a single line of text.
Parent and child levels swapped	Headings	A section and its first subsection swap levels: the subsection now contains its parent.
All headings at one level	Headings	Every heading is ##: title, sections and subsections are siblings.
Body paragraph turned into heading	Headings	A full sentence of body text is emitted as a heading.
List nesting flattened	Lists	Sub-items lose their indentation and become siblings of their parent item.
MINOR ERRORS		
All headings shown as bold lines	Headings	Every heading is a bold line, a common habit of OCR-first parsers.
All bold and italic stripped	Emphasis	The parser emits no emphasis at all: every bold and italic span is plain text.
Heading shown as a bold line	Headings	A heading is emitted as a standalone bold line: it still looks like a heading but is no longer one.
List markers dropped	Lists	Bullets and numbers are dropped; each item stays on its own line.
Code block emitted as text	Lists	A fenced code block loses its fences and reads as a paragraph.
Footnote and unit superscripts lost	Strikethrough and scripts	Superscripts after a word or symbol become plain digits: billion1, site.12, m2.
Numbered list turned into bullets	Lists	A numbered list comes out with bullets (or bullets with numbers).
Bold lost	Emphasis	Bold phrases and bold label lines lose their bold.
One bold phrase lost	Emphasis	A single inline bold phrase loses its bold.
One heading one level too deep	Headings	A heading is one level deeper than it should be, so it hangs under the wrong parent.
Short line promoted to heading	Headings	A short line of body text becomes a heading.
Subscripts lost (CO_2 → CO2)	Strikethrough and scripts	Chemical subscripts become plain digits. Readable, but the formula is no longer marked up.
Underline lost	Emphasis	Underlined text (a cross-reference, a defined term) is plain. Only on pages without strikethrough.
Italic lost	Emphasis	Italic phrases (species, defined terms, variables) become plain.
Item nested under its sibling	Lists	A top-level item is indented under the item before it.
List item split in two	Lists	A wrapped list item is read as two items.
Paragraph turned into a bullet	Lists	A body paragraph is emitted as a list item.
Whole paragraph in bold	Emphasis	A body paragraph is emitted entirely in bold.
Bold added to plain words	Emphasis	Two plain words in running text are set in bold.
Bold rendered as italic	Emphasis	An inline bold phrase comes out in italic.
Highlight lost	Emphasis	A highlighted warning loses its highlight.
Bold span extended by a word	Emphasis	The bold run swallows the next word: **Lakeside plant** instead of **Lakeside** plant.
Inline code lost	Emphasis	Inline code (a command, a function name) is plain text.
Italic added to plain words	Emphasis	Two plain words in running text are set in italic.
DEBATABLE ERROR TYPES (LISTED, NOT GRADED)		
Bold label line promoted to heading	Headings	A standalone bold line ("Procedure") becomes a ### heading. The reference treats bold lines as the lowest title level.
Second half of the page missing	Equivalent syntax	The output stops half way. Formatting of what is there is perfect.
Struck text omitted	Strikethrough and scripts	The struck text is left out entirely. The live text is right, but the record of what was deleted is gone.
BENIGN VARIATIONS		
All heading levels one deeper	Equivalent syntax	# becomes ##, ## becomes ###: the hierarchy is the same.
Bold split word by word	Equivalent syntax	**heating** **demand** instead of **heating demand**.
Closing #s and bold inside headings	Equivalent syntax	## **Abstract** ## instead of ## Abstract.
Every item numbered 1.	Equivalent syntax	Markdown renumbers 1. 1. 1. as 1, 2, 3.
Everything written as HTML	Equivalent syntax	<h2>, <b>, <i>, <del>, and <p> instead of Markdown.
HTML entities	Equivalent syntax	&, ' and instead of the characters.
LaTeX exponents and indices	Equivalent syntax	10^{5} and CO_{2} instead of HTML tags.
Ordinal superscript lost (1st)	Strikethrough and scripts	1st written as 1st. Nothing a reader needs is lost.
Output wrapped in “‘markdown	Equivalent syntax	The whole page arrives inside a “‘markdown fence, as some VLMs return it.
Punctuation outside the bold	Equivalent syntax	**Electric**: instead of **Electric:**.
Setext headings	Equivalent syntax	Title underlined with ===, sections with —.
Spacing and indentation	Equivalent syntax	Extra blank lines, trailing spaces and 4-space nesting.
Unicode superscripts and subscripts	Equivalent syntax	10^5, CO_2 and m^2 written with Unicode characters.
__bold__ and _italic_	Equivalent syntax	Underscore emphasis instead of asterisks.
• bullets and 1) numbers	Equivalent syntax	Bullet glyphs and parenthesis numbering instead of - and 1.

Table 7 Severity policy for Tables: 57 error types (27 critical, 18 major, 9 minor, 3 debatable) and 5 benign variations.

Name	Category	What it is
CRITICAL ERRORS		
Table missing entirely	Missing or extra	The parser returns no table at all, only a placeholder line, as when it fails on a complex layout.
Column headers swapped	Values in the wrong place	Two column headers trade places, so both columns are labelled with the other's period or measure.
Decimal point inserted	Numbers	A speck or a line break is read as a decimal point inside a whole number: 97 → 9.7.
Decimal point misplaced	Numbers	The decimal point is lost or moves one place: 13.8 → 138, 0.41 → 4.1.
Headers shifted one column	Values in the wrong place	The lowest header row slides one column right, so every column is labelled with its left neighbour's header.
Minus signs added	Numbers	Some positive values come out negative, as when a dash or a parenthesis next to the number is misread.
Sign or comparator lost	Numbers	Accounting negatives lose their parentheses, minus signs vanish, and < or > disappear from ranges: (21,330) → 21,330, <200 → 200.
Wrong digits	Cell content	One digit is changed in some numeric cells.
Wrong year in a column header	Values in the wrong place	One column header reads another year (2023 → 2022), so a whole column of figures is attributed to the wrong period.
Empty cell skipped, values shift left	Structure	In some rows an empty cell is dropped, so every value after it slides one column left.
Footnote mark glued to a number	Numbers	A superscript footnote mark is read as one more digit of the number: $1{,}284^{1}$ → 1,2841.
Last row (total) dropped	Missing or extra	The last body row, often the total, is left out, as when a parser stops at a rule above the total line.
Plausible row invented	Missing or extra	A row that is not on the page is added before the last row, with a believable label ("Other") and believable numbers.
Row labels swapped	Values in the wrong place	Two row labels trade places, so both rows carry the other row's numbers.
Rows missing	Missing or extra	Some data rows are left out.
Two values swapped within a row	Values in the wrong place	In one row the values of the first two value columns trade places (2023 and 2024 swapped), so both periods carry the other's figure.
Values shifted one row	Values in the wrong place	Row labels stay in place but every row's values come from the row below (the last row takes the first row's values), as when a parser loses one row boundary. Every number is now under the wrong label.
Values swapped between rows	Values in the wrong place	A value trades places with the value in another row of the same column, as when a parser misreads row boundaries.
Values swapped within a row	Values in the wrong place	In some rows two values trade columns, so each sits under the other's header. Every value is still present.
Column missing	Missing or extra	One data column is left out entirely.
Header rows missing	Row and column structure	Column headers are left out; the data rows are intact but no longer say what each column is.
OCR character confusions	Cell content	Character confusions such as 0→O, 1→l, 5→S, rn→m inside cells.
Adjacent cells merged	Structure	Two neighbouring values end up in one cell ("48,215 41,906") and the row loses a column.
Empty cells filled in	Text in cells	Cells that are empty on the page get invented content (a flag, a zero or a dash with a number), a typical hallucination on sparse tables.
European number format	Numbers	Thousands and decimal separators are swapped (48,215 → 48.215, 0.92 → 0,92). Each number now reads as a different value to anyone using the other convention.
Invented extra table	Missing or extra	A second small table that is not on the page is added after the real one.
Table transposed	Structure	Rows become columns. Every value keeps its row and column labels, but the layout differs from the page.
MAJOR ERRORS		
Two rows merged	Row and column structure	Two neighbouring rows collapse into one, each cell holding both values ("9,412 7,208").
Values missing	Text in cells	Some filled cells come out empty, as with faint or overprinted text.
Merged-cell spans lost	Structure	The colspan and rowspan attributes of merged cells are lost: a header that should cover two columns now covers one, so every cell after it slides one column left and rows become ragged. Text is unchanged; positions are wrong.
Two tables merged into one	Structure	Two separate tables on the page come out as one table: the second table's header becomes an ordinary row and its rows are padded to the first table's width. Only applies to pages with two tables (the Two tables on one page sample).
Units dropped	Numbers	Units next to numbers disappear and a Units column is left blank: 450 W → 450, mg/dL → (empty).
Values rounded	Numbers	Decimal values lose a digit of precision (0.92 → 0.9, 13.8 → 14, 55.8% → 56%).
Group labels missing	Row and column structure	Labels that span several rows (Lipid panel, Metabolic) are left blank, so those rows lose their grouping.
One cell split in two	Row and column structure	In one row a long label is split across two cells, pushing every value in that row one column to the right.
Table split into several tables	Structure	The table is emitted as several separate tables, as when a parser breaks it at page or section boundaries; only the first part keeps the header.

Severity policy for Tables, continued

Name	Category	What it is
Long cells truncated	Text in cells	Longer cell texts are cut after their first word or two, as when text overflows a narrow column.
Row label wrapped into a new row	Row and column structure	A long row label wraps onto a second line that becomes its own row with empty values ("Research and" / "development").
Body rows reordered	Structure	Some data rows are moved. Each row is intact, but the table no longer reads in the page's order.
Title row dropped	Row and column structure	The table's title row (Consolidated statement of operations, USD thousands) is lost, along with the unit it states. Applies to tables with a title row.
Two columns swapped	Structure	Two data columns trade places, headers included. Every value stays under its own header.
Two header rows joined into one	Structure	Multi-row headers are collapsed into one row by joining the texts of each column ("Three months ended Sep 30, 2024").
Column duplicated	Missing or extra	One data column appears twice side by side.
Row labels abbreviated	Text in cells	Row labels are shortened to abbreviations the page does not use (Research and development → R&D, North America → NA, Total cholesterol → Total chol.).
Row repeated	Missing or extra	One data row appears twice in a row.
MINOR ERRORS		
Column headers reworded	Values in the wrong place	Column headers are paraphrased (Three months ended → 3 months ended, Reference range → Ref. range, Region → Area). The columns still mean the same thing to a careful reader, but the labels no longer match the page.
Header repeated mid-table	Row and column structure	The header rows appear a second time in the middle of the body, as when a table continues across a page break and the repeated header is kept as data.
Empty column inserted	Row and column structure	A blank column appears at the right edge, as when a ruling line or margin is read as a column.
Empty row inserted	Row and column structure	A blank spacer row appears between the header and the data, as when vertical whitespace on the page is read as a row.
Number format changed	Cell content	Thousands separators are removed and accounting negatives become minus signs: 48,215 → 48215, (21,330) → -21330. Same values.
Trailing zeros dropped	Numbers	Decimals lose their trailing zeros (0.60 → 0.6, 12.0 → 12). The value is equal but its stated precision is gone.
Letter case changed	Cell content	Row labels are written in capitals.
Currency symbol moved	Numbers	Currency symbols are written after the number ($0.41 → 0.41 $). Same amount, different notation. Applies to tables with currency values.
Punctuation dropped inside cells	Text in cells	Commas, periods and hyphens inside text cells are lost ("Net income per share, diluted" → "Net income per share diluted", "Asia-Pacific" → "Asia Pacific"). Numbers are untouched.
DEBATABLE ERROR TYPES (LISTED, NOT GRADED)		
Header cells not marked as headers	Format	Header cells are written as ordinary data cells (td instead of th). Text and layout are identical; only the information that these rows are headers is lost.
Markdown pipe table	Format	The table as a Markdown pipe table. Cell text is identical, but spans cannot be expressed: a spanning cell keeps its text in the first position and leaves the rest empty, and only the first row counts as a header.
Merged cells repeated in every position	Format	Every merged cell (a cell with colspan or rowspan, such as "Three months ended" over two columns or "Lipid panel" over three rows) is split back into ordinary cells, each holding a copy of its text. The grid reads the same; the merge itself is gone.
BENIGN VARIATIONS		
Bold row labels	Cell content	Row labels are wrapped in Markdown bold markers.
Empty cells as dashes	Cell content	Blank cells are written as an em dash.
Extra HTML markup (thead, styles, bold)	Format	Same table with thead and tbody sections, inline style attributes and bold header text.
Title above the table	Format	A spanning title row is emitted as a heading just above the table instead of inside it.
Typographic variants	Cell content	En dashes become hyphens, × becomes x, μ becomes u, and a non-breaking space appears before units.

Table 8 Severity policy for Charts: 39 error types (22 critical, 9 major, 3 minor, 5 debatable) and 14 benign variations.

Name	Category	What it is
CRITICAL ERRORS		
Negative signs lost	Wrong numbers	Bars below the axis are transcribed as positive values. Only applies to charts with negative values.
Order of magnitude off	Wrong numbers	A decimal point is misread, so a value is ten times too large or too small (412.5 → 41.25).

Severity policy for Charts, continued

Name	Category	What it is
Scale lost	Scale and units	Values are written in the base unit without saying so, 412500000 instead of 412.5 million, or as fractions without a % sign. Nothing in the output tells the reader the scale.
Cumulative values read	Wrong numbers	Each series is read at the top of its stack instead of its own height, so later series include the earlier ones (the classic stacked-bar misreading).
Digit typo	Values	One digit is wrong in some values, as from a misread label.
Pie slice dropped, rest renormalized	Missing or invented data	The last slice of a pie is left out and the remaining shares are rescaled to add up to 100%, so every share left is wrong too. Only applies to pie charts.
Secondary axis read on the primary scale	Wrong numbers	The last series belongs to a right-hand axis but is read against the left one, so its values come out scaled by the ratio of the two axes (×4 here). Only applies to charts with several series.
Series names swapped	Structure	The first two series get each other's names, as from misreading the legend colors. Every value sits under the wrong label.
Shares instead of values	Wrong numbers	Each value is replaced by its share of the category total (North America 46.6% instead of 412.5). The labels are right; the quantity is a different one.
Two series summed into one	Missing or invented data	The first two series are added together into one column named after both, as when stacked segments are read as a single bar. Only applies to charts with several series.
Values copied from the neighbouring chart	Values attached to the wrong label	The page has a second chart with the same layout just below (same categories and series, other values). The table under this chart's title carries the neighbour's values; the neighbour's own table follows.
Values shifted one category	Structure	Every series is shifted by one position along the axis, so each value sits next to the wrong category label. The first category is left empty.
Wrong unit stated	Labels	The numbers are right but the stated unit is wrong ("Values in USD thousands" for a chart in millions, "Values in percentage points of GDP" for shares).
Category labels swapped	Values attached to the wrong label	Two axis labels are read in the wrong order, so every series is wrong for both categories.
Value copied from a neighbour	Wrong numbers	One point repeats the value of the point next to it instead of its own, as when a data label is attached to the wrong bar.
Values 8–15% off	Values	Some values are clearly wrong, 8 to 15% away from the truth.
Values swapped between categories	Values attached to the wrong label	Two neighbouring categories trade values in the first series (Q2 gets Q3's bar and the reverse).
Values swapped between series	Values attached to the wrong label	For some categories the values of two series trade places, as when bar colors are confused for one group.
All values one gridline too high	Wrong numbers	Every value is read one gridline above its true height, a systematic misreading of the axis.
Invented series	Missing or extra data	An extra series that is not in the chart is added with plausible values.
Legend lost	Structure	Series are named Series 1, Series 2 instead of their legend names. Values are right but cannot be attributed. Critical when the chart has several series (the values cannot be told apart); major on a single-series chart (the values stay readable).
Invented category	Missing or invented data	A category that is not on the axis is added with plausible values (an extra quarter, year or vendor).
MAJOR ERRORS		
Series missing	Missing or invented data	The last data series in the legend is left out entirely (a trendline is not a data series: see Trendline left out). Only applies to charts with several series.
Category missing	Missing or invented data	One whole category is left out of the extraction.
Points missing	Missing or extra data	Some cells of the data series are left empty (a trendline is not a data series).
Category labels truncated	Labels	Long axis labels are cut short (Q1 2023 → Q1, Transsion → Trans, 2016 → 16), so some categories become ambiguous.
Series repeated	Missing or invented data	The first series appears twice as two identical columns.
Title dropped	Labels	The chart title and unit note are left out. The numbers are intact, but nothing says what they measure.
Wrong title	Labels	The chart title is replaced by a different one (Quarterly revenue by region → Quarterly profit by region), so the table claims to show another measure.
Row repeated	Missing or extra data	One category row appears twice, a common looping failure.
Total row added	Missing or invented data	A Total row that the chart does not show is appended, with values computed by summing each series.
MINOR ERRORS		
Ranges instead of values	Wrong numbers	Each value is written as a small range around the true number (412.5 → 410–415), as a cautious reader of an unlabeled chart would.
Category axis mislabeled	Labels	The category column is headed with the wrong axis name (Quarter → Month, Year → Age group, Vendor → Region).
Series names abbreviated	Labels	Series names are shortened (North America → N. America, Asia-Pacific → APAC, Women → F). Recognizable, but not what the legend says.
DEBATABLE ERROR TYPES (LISTED, NOT GRADED)		

Severity policy for Charts, continued

Name	Category	What it is
Category labels reworded	Labels	Axis labels are written differently: Q1 2023 becomes 1Q23, 2016 becomes '16, Others becomes Other.
Labels misspelled	Labels	Series and category names carry a small typo (Europe → Eurpoe, Samsung → Samsnug). The data is right and recognisable.
Rounded to whole numbers	Values	Values are rounded to integers. Close on large numbers, several percent off on small ones.
Small reading noise	Values	Every value is re-read with a small error, as when estimating from the axis. On a chart without data labels this is normal; on a labeled chart it is a transcription error.
Values snapped to gridlines	Wrong numbers	Every value is rounded to the nearest gridline, as a lazy reader of an unlabeled chart would (412.5 → 400). Wrong on a labeled chart, borderline on an unlabeled one.
BENIGN VARIATIONS		
Bullet list instead of a table	Output format	One bullet per category listing each series value. Complete and correct, but not a table.
Categories reversed	Structure	Rows are listed from last to first. Labels stay attached.
Decimal commas	Output format	Values use a decimal comma (412,5) as in many European locales. Same numbers.
HTML table	Output format	The same table emitted as HTML instead of Markdown.
Hedged value	Output format	One value per series carries a hedge, as a parser writes a reading it is unsure of: approximately 438.2, ~305.1, 224.8 (est.). Same numbers.
JSON instead of a table	Output format	The data is emitted as a JSON code block. Complete and correct, but not a table.
Negatives in parentheses	Output format	Negative values are written in accounting style, (3.2) instead of -3.2. Only applies to charts with negative values.
One table per series	Output format	Each series gets its own two-column table under a heading with the series name.
Rescaled, unit in header	Scale and units	The parser converts to a different unit and says so in the column headers, for example 0.4125 with "North America (USD bn)". The data is correct; only the unit changed.
Series in another order	Structure	Columns are listed in a different order than the legend. Labels stay attached.
Table transposed	Structure	Series become rows and categories become columns. Every value keeps both labels.
Trendline left out	Missing or invented data	The chart draws an Excel trendline over its bars (legend entry Linear (LDCs)) and the parser leaves it out. The trendline is computed from the data, so nothing a reader needs is lost. Only applies to charts with a trendline.
Units written in cells	Scale and units	Each cell carries its unit, like $412.5M or 5.1%. Same numbers, more verbose.
Value and share in one cell	Output format	Each cell carries the value and, in parentheses, its share of the category total: 412.5 (46.6%). Every value of the chart is there; the share is extra arithmetic. Only applies to charts with several series that are not in percent.

Table 9 Severity policy for Forms: 37 error types (13 critical, 13 major, 8 minor, 3 debatable) and 34 benign variations.

Name	Category	What it is
CRITICAL ERRORS		
Decimal point lost in an amount	Values	52,287.47 written 5228747: same digits, a hundred times the amount.
Digit misread in a value	Values	One digit changes in some numeric values (amount, date, id): the form now states a different number.
Two digits of an ID transposed	Values	An identifier with two neighbouring digits swapped (123-45-6789 → 123-54-6789): every digit is there, the number is wrong.
Values of two fields swapped	Values	Two filled fields exchange their values: each value is attached to the wrong caption.
Repeated block's values swapped	Values	Two blocks with the same captions (Applicant / Co-applicant, State line 1 / 2) exchange their values: each value is written under the wrong person or line.
Values shifted to the next field	Values	A run of values slides down by one field (a column read out of step): every value in the run sits under the wrong caption.
Selected option changed	Checkboxes and signatures	In a single-choice group the tick moves to another option: Married filing jointly instead of Single.
Text value misread as another word	Values	A text value is replaced by a different word: Dayton becomes Daytona Beach County.
Checkbox state flipped	Checkboxes and signatures	Standalone or multi-choice boxes come back with the opposite state: a No reads as a Yes.
Every option of a choice ticked	Checkboxes and signatures	A single-choice group comes back with all its options ticked (the over-selection CheckboxQA reports): the answer is no longer readable.
Table column slipped a row	Values	In a form table one column's values sit one row below their own: each amount belongs to the wrong line.
Value invented in a blank field	Values	Blank fields come back filled with plausible values that are not on the page.

Severity policy for Forms, continued

Name	Category	What it is
Signature claimed on an unsigned line	Checkboxes and signatures	An empty signature line comes back as signed.
MAJOR ERRORS		
All values dropped (blank template)	Values	The parser returns the blank form: every caption, no filled value.
Chosen option unticked (no choice shown)	Checkboxes and signatures	A single-choice group comes back with every box empty: the answer is lost, though no other option is claimed.
Filled field missing	Values	Filled fields are left out entirely, caption and value.
Ticked option missing	Checkboxes and signatures	A ticked option is left out of the output.
Value dropped (caption kept)	Values	Some filled fields come back with their caption but no value: the reader takes them as blank.
Caption repeated as the value	Values	A filled field comes back with its own caption in the value slot (Employer's name: Employer's name): the value is lost.
Signature lost	Checkboxes and signatures	A signed line comes back blank.
Ticks dropped (option text kept)	Checkboxes and signatures	Checked options lose their box glyph: the option is listed but nothing says it was ticked.
Value truncated	Values	A multi-word value is cut after its first word: an address loses its city.
Values without their captions	Values	Values are written on their own lines, captions dropped: the value is there but no longer says what it is.
Amount rounded to whole units	Values	52,287.47 written 52,287: the cents are gone, the amount is nearly right.
Choice group written without boxes	Checkboxes and signatures	A select group's options are written as plain text, no glyphs: which one was chosen is lost.
Value marked [illegible]	Values	A filled value replaced by an [illegible] marker: the parser abstains instead of guessing (KIE-HVQA rewards this over a hallucinated value).
MINOR ERRORS		
Typo in a text value	Values	One letter of a text value is misread (Sorensen → Soransen); no digit changes.
Form table flattened to lines	Captions and structure	A form table is written as one line per row (Row 1: Description . . . Amount . . .): values stay in their row but the grid is gone.
Blank fields omitted	Values	Empty fields are left out of the output: nothing false, but the form loses fields.
Extra text after a value	Values	A value runs into neighbouring printed text: 60082 (see attached).
Unticked boxes omitted	Checkboxes and signatures	Standalone and multi-choice boxes that are not ticked are left out.
Caption garbled	Captions and structure	Two words of a caption are misread; the value is right.
Empty boxes dropped (option text kept)	Checkboxes and signatures	Unticked options lose their empty box glyph. A reader assumes they are not ticked; the box is gone.
Form table rows in another order	Captions and structure	A form table's rows come back in reverse order, each row intact (DocILE's line-item matching ignores order; ParseBench's table metric does not).
DEBATABLE ERROR TYPES (LISTED, NOT GRADED)		
Captions, then values, in two runs	Debatable	A two-column block read row by row: two captions, then their two values. The pairing is only implied by order.
Printed hint read into a blank field	Debatable	The instruction printed in an empty box (Select from the drop-down list) is transcribed as its value.
Second half of the page missing	Debatable	The output stops halfway down the page.
BENIGN VARIATIONS		
Amounts with currency and cents	Format	52,287.47 written $52,287.47 and 300,000 written 300,000.00 (VRDU's numeric price match).
Ballot-box glyphs (☒ ☐)	Format	Unicode boxes instead of [x] / [ ].
Blank fields show their format hint	Format	An empty date box written MM/DD/YYYY, other empty boxes as a dotted line.
Blank fields written N/A	Format	Empty fields as N/A.
Blank fields written as _ _ _ _	Format	Empty fields as an underscore line.
Bold key: value paragraphs	Format	**Caption**: value lines, options inline after their caption.
Box states as Yes / No values	Format	Each option written as a field with a Yes or No value, the way extraction outputs write booleans.
Box states as words (Selected / Not selected)	Format	Each option followed by its state in words: Single: Selected, Married: Not selected.
Box-per-character field spaced out	Format	A comb field read one box at a time: 919293710 written 9 1 9 2 9 3 7 1 0.
Boxes after their option	Format	Yes [x] No [ ] instead of [x] Yes [ ] No.
Captions in capitals	Format	Every caption upper-cased.
Check marks and Azure markers	Format	✓ for ticked, :unselected: for empty boxes.
Choice written as the caption's value	Format	Filing status: Single.
Dates normalised to ISO	Format	08/23/2026 and 14/01/2022 written 2026-08-23 and 2022-01-14 (Kleister's canonical form).
Dates reformatted	Format	1997-09-18 written 18/09/1997.
Dates with month names	Format	08/23/2026 written August 23, 2026; 1997-09-18 written 18 September 1997 (VRDU's type-aware date match).

Severity policy for Forms, continued

Name	Category	What it is
HTML table with checkbox inputs	Format	<table> rows, boxes as <input type=checkbox checked>.
IDs and phone numbers re-punctuated	Format	920-24-6595 written 920 24 6595, (666) 431-5871 written 666.431.5871.
JSON nested by section	Format	One JSON object per section, blanks as null (the dictionary output ANLS* scores).
JSON-like key/value lines	Format	"Caption": "value", boxes as true / false.
Long value wrapped onto the next line	Format	A multi-word value (an address) continues on the line below its caption line.
No colon after captions	Format	**Caption** value.
Plain Caption: value lines	Format	No markup at all, [X] / [ ] boxes inline.
Printed ids dropped	Format	Box and line numbers (1a, 12) left out of the captions.
Radio buttons as (X) / ( )	Format	Boxes written as parenthesised radio marks: (X) Single ( ) Married.
Round radio glyphs (● ○)	Format	Filled and empty circles instead of boxes.
Section headings left out	Format	Every field kept, the section headings (Part I, Employee) dropped.
Sections in another order	Format	Whole sections emitted in a different order (a two-column page read column by column).
Single choice: only the chosen option	Format	A single-choice group lists only its ticked option.
Thousands separators dropped	Format	28,898.28 written 28898.28.
Two fields on one line	Format	A two-column block written row by row with both fields on one line: Name: Ana Ruiz Date: 2026-04-27.
Two-column table	Format	Each section as a Field | Value table.
Values in capitals	Format	Every filled value upper-cased, as some OCR engines return block capitals.
Values underlined	Format	Filled values wrapped in <u>...</u>, as the enriched forms output writes them.

Table 10 Severity policy for Visual grounding: 24 error types (7 critical, 7 major, 9 minor, 1 debatable) and 14 benign variations.

Name	Category	What it is
CRITICAL ERRORS		
One page-wide box		A single Text box covering the page with all the text.
Biggest paragraph missed		Perfect except paragraph 5.
Five paragraphs in one box		All five paragraphs as one Text box. The box swallows the picture and the table.
Right box, wrong text		Paragraph 3's box carries paragraph 4's text.
Coarse blocks		Two Text blocks: title through paragraph 3, and icon through paragraph 5.
Japanese paragraph garbled		Paragraph 4's box is right but about half of its characters are wrong or missing.
Loose boxes (IoU about 0.45)		Every box grown 1.49× per side.
MAJOR ERRORS		
Every box six times		Every GT box predicted once per Basic class. Correct class first.
50 junk boxes		Perfect, plus 50 random boxes with random classes and no text.
Table box cuts the header row		Perfect, but the table box starts below the first row.
Table box cuts the last column		Perfect, but the table box stops before the last column.
Header merged into a paragraph		Section header and paragraph 1 as one Text box.
Two columns, one box over both		Same page. Paragraph 5 predicted as one box around both column parts, with the whole paragraph text. The box covers the gutter and most of paragraph 6.
Loose boxes (IoU about 0.64)		Every box grown 1.25× per side.
MINOR ERRORS		
Empty boxes between table rows		Perfect, plus 6 empty Text boxes between the table rows.
Five boxes on blank margins		Perfect, plus five empty boxes in the page margins.
Second box over a matched line		Perfect, plus a Section box over the first line of paragraph 2. Those words are locked to paragraph 2's box.
Box nicks the next paragraph		Perfect, but paragraph 2's box extends over the first line of paragraph 3. Text is right.
Header and footer as page-wide bands		Page header and page number as full-width bands with the right text.
Text boxes inside a picture		Perfect, plus 4 Text boxes inside the big picture.
Hyphen dropped, word left split		The header is output as "Quarterly infor mation report": the hyphen is lost and the word stays in two pieces.
Japanese paragraph, one character wrong		Paragraph 4 is right except one character (the character for three read as the character for two).
Tiny icon missed		Perfect except the 18×18 pt icon.
DEBATABLE ERROR TYPES (LISTED, NOT GRADED)		
Split at the wrong place		Paragraphs 1–2 as two boxes cut mid-paragraph: lines 1–3 of P1, then line 4 of P1 with all of P2. All text is present and in order; the cut point is off. Benchmarks disagree on whether this matters.
BENIGN VARIATIONS		

Severity policy for Visual grounding, continued

Name	Category	What it is
Boxes 1.5 pt off		Every box shifted 1.5 pt to the right (IoU about 0.95). Tight enough to cite.
Boxes on optional sub-images		Perfect, plus a Picture box on each optional sub-image inside the big picture.
Every class wrong		Right boxes and text, but every element carries the wrong class. Classes are reported separately.
Hyphenated word joined		The header is output as "Quarterly information report", the word rejoined across the line break.
Hyphenated word kept as printed		The two-line header is output as "Quarterly infor- mation report". The benchmark joins "infor- mation" on both sides.
Japanese paragraph, read right		Paragraph 4 is Japanese. The output box and text are right; every CJK character is its own token.
Paragraph split into lines		Paragraph 5 (7 lines) as 7 line boxes, each with its line text.
Perfect		Every element boxed tightly, right class, right text.
Annotations by line, output by paragraph		Paragraph 5 annotated as 7 line boxes; prediction is one tight paragraph box.
Table labelled Picture		Perfect, but the table box has class Picture.
Table plus tight cell boxes		Perfect, plus one Text box per table word, tight to the word.
Two columns, one box per part		Paragraph 5 continues from the left column into the right one. One box per part, each with its own text.
Two columns, one element in two segments		Same page. Paragraph 5 predicted as one element with two segments (one per column) carrying the whole paragraph text.
Two paragraphs in one box		Paragraphs 1 and 2 as one Text box holding both texts.

A.3 Metric Definitions by Dimension

This appendix defines the six metrics of Section 3.2 one by one. For each of Text, Formatting, Tables, Charts and Forms, it gives what the metric checks, how it compares an output with the annotations, the parsing score m and the errors that enter the severity cap c. Visual grounding has a parsing score and no cap. The worked examples use short hand-built pages.

A.3.1 How detected errors lower the cap

A metric detects errors by aligning the normalized output with the annotations and reading *detected errors*, each with a type and a severity level, off the mismatches, such as an unmatched passage in Text or a value under another header in Tables. Repeated errors from one cause, such as margin line numbers written into the text, are charged as one group.

The first detected error of a severity level costs its full penalty, since one wrong number misleads a reader however long the page is. Each later one counts $\min(1, K/N)$, where N is the size of the annotations in the metric's units (tokens for Text, cells for Tables and Charts, scored items for Formatting, fields for Forms) and K is a size constant of the metric. This keeps a long page with many independent slips from falling to zero, whereas a per-unit rate would make one wrong number on a long page almost free. N counts units in the annotations and not the output, since padding the output would otherwise make errors cheaper. The *effective error count* of level ℓ is

$$n_\ell^{\text{eff}} \;=\; \min(n_\ell, 1) \;+\; \max(0, n_\ell - 1) \cdot \min\Big(1, \frac{K}{N}\Big), \tag{7}$$

where n_ℓ is the number of detected errors of level $\ell \in \{\mathsf{C}, \mathsf{M}, \mathsf{m}\}$ (critical, major, minor) on the page, and the cap is

$$c = \max\Big(0,\; 1 - w_{\mathsf{C}}\, n_{\mathsf{C}}^{\text{eff}} - w_{\mathsf{M}}\, n_{\mathsf{M}}^{\text{eff}} - \min\left(0.15,\; w_{\mathsf{m}}\, n_{\mathsf{m}}^{\text{eff}}\right)\Big), \tag{8}$$

with $w_{\mathsf{C}} = 0.10$, $w_{\mathsf{M}} = 0.05$ and $w_{\mathsf{m}} = 0.01$. The total of the minor errors is limited to 0.15, one critical plus one major penalty, which keeps cosmetic errors alone from costing more than a misread fact. With the Text constant $K = 400$, five wrong numbers cap a 1,200-token page at 76.7 ($1 + 4 \cdot \frac{400}{1200} = 2.33$ critical errors) and a 300-token page at 50. These weights and constants are implementation choices that we adjusted to meet the policy's targets (Section 2.3); they are not the targets themselves.

A.3.2 Text

The Text metric checks that the output contains all words, numbers and identifiers in the annotations, in whole sentences and a sensible order. It also tells moved text from missing text and a wrong number from a typo. As the ParseBench-1 Content Faithfulness metric averages equal-weight rules, a misspelled word and a wrong revenue share on a 10-K page (a US annual report) both cost half a point. The $1 - \text{NED}$ score ranks errors almost in reverse, scoring a moved sidebar (63.8) below a deleted one (81.9).

Both sides are reduced to plain text, so supported Markdown, HTML and emphasis syntax cost nothing. Numbers, identifiers, units, currency signs and negators are tracked as *key tokens*, whose change can alter a stated fact. Greedy string tiling [23] finds maximal common token runs wherever they occur, including in a moved block, and an in-order longest common subsequence aligns the remaining tokens between tiles (Figure 6). A move then costs only order and one *moved block* error.

Four components measure the match, with *words* the F1 of matched tokens and *key* the F1 of key tokens. The *integrity* component is the share of sentence words whose sentence is not interrupted by text from elsewhere on the page. The *order* component is the share of matched tokens in the heaviest in-order chain, the set of tiles that keep their order in the annotations with the largest total length in tokens. The parsing score subtracts the weighted losses:

$$m_{\text{text}} = \max\Big(0,\; 1 - \big[\,(1 - \text{words}) + 1.5\,\rho\,(1 - \text{key}) + 0.25\,(1 - \text{integrity}) + 0.1\,(1 - \text{order})\big]\Big), \tag{9}$$

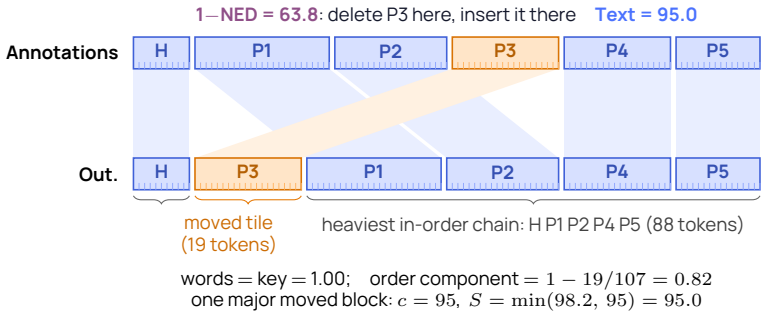


Figure 6 Move-tolerant alignment in the Text metric. The 107-token annual-report page (running head H, paragraphs P1–P5), with the sidebar P3 printed first. Greedy string tiling matches the blocks wherever they moved; blue bands are blocks in the order given by the annotations and the orange band is the moved tile. The words and key components, the F1 of matched tokens and of key tokens, stay at 1. The order component, the share of matched tokens in the heaviest in-order chain, falls to 0.82, which lowers the parsing score by 0.1×0.18, or 1.8 points, to 98.2. The move is one major error, and the cap of 95 sets the Text score, whereas 1−NED charges the same move 36 points.

where $\rho = \min(1, n_c/10)$, with n_c the larger of the key-token counts in the annotations and the output, keeps two numbers from dominating a page, and an undefined component counts as 1. A key-token F1 of 0.9 costs 15 points when $\rho = 1$, while order has the smallest weight, since a reader can still find text that moved.

Unmatched runs become detected errors, of which a wrong number, a changed negator (detected lexically) or an omitted passage of 12 or more tokens is critical. Shorter omissions, wrong words and a moved block are major, while spelling or accent changes are minor. The cap is Equation (8) with $K = 400$ tokens, but moved blocks and cut sentences together cost at most 30 points. This limit keeps a page read in the wrong column order with all its words intact from being capped at zero. On a 56-token insurance notice, benign variations of format score 100, and swapping two paragraphs costs one major error (95.0). Changing "30 days" to "60 days" is one critical error (cap 90), and the parsing score falls further to 83.2, which sets the score.

A.3.3 Formatting

The Formatting metric checks strikethrough, exponents, and the role and level of headings and list items, which is the formatting that changes what a reader understands. The ParseBench-1 Semantic Formatting metric checks the annotated formatting spans with one rule per span, never penalizes invented formatting, and accepts one markup syntax per rule. Writing `__bold__` where a rule expects `**bold**` costs up to 67 points on a page of bold labels (Figure 7). We score the 480 core text pages that carry at least one formatting item.

All supported syntaxes map to one representation, in which a line has a role (a paragraph line, a heading with its level, a list item with its depth). Characters carry style bits for bold, italic, underline, strikethrough, scripts, highlight and inline code. In this representation `**`, `__` and `<b>` all read as bold, and ATX, setext and `<hN>` headings all read as headings. A *scored item* is a style span, scored script token, heading, list item or code block in the annotations, or one that the output adds to aligned text. Words are aligned as in Text, even across moved blocks, and heading levels count as relations between consecutive headings, which makes a uniform shift of all levels cost nothing.

A lost, added or changed item is one error with a loss of 1.0 (critical), 0.5 (major) or 0.2 (minor). The parsing score subtracts a damped mean of these losses over the n_{items} items in the annotations and the output:

$$m_{\text{fmt}} = 1 - \frac{\sum_a \text{loss}_a}{n_{\text{items}} + 3}. \tag{10}$$

Here, a indexes the scored items, and loss_a is the loss assigned to item a; correct items contribute zero. The sum $\sum_a \text{loss}_a$ adds these losses across all scored items. The 3 acts as three items without loss and keeps one lost item on a page with little formatting from costing most of the parsing score, leaving such pages to the cap. To keep many lost minor items from disappearing into an average, the item losses rank severity less

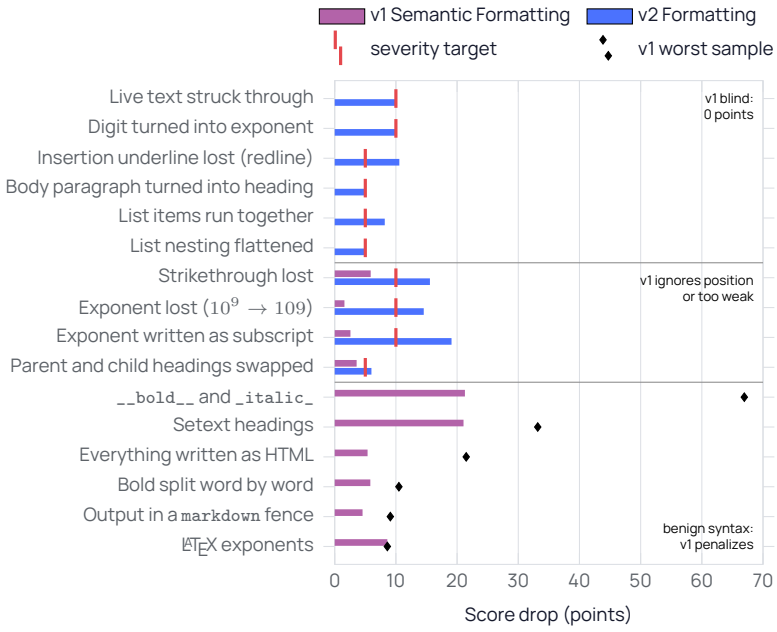


Figure 7 What the ParseBench-1 Semantic Formatting metric and the ParseBench-2 Formatting metric deduct for one change at a time (mean score drop on the pages of the Formatting test suite at 50% corruption strength, Section 4.1). Top: errors the ParseBench-1 metric cannot see. Middle: errors it sees only weakly, as its rules accept the formatted text anywhere on the page. Bottom: syntax variants, which the ParseBench-1 metric penalizes by up to 67 points on one sample (◆). A missing bar is a drop of 0. A red tick is the smallest drop that the row's class should cause, 10 points for a critical error and 5 for a major one, and a longer bar also meets it. Rows without a tick are benign variations and should cost nothing. A diamond is the largest v1 drop on a single sample page. The v2 Formatting metric meets every target and charges none of the syntax variants.

steeply than the cap (5:1 from critical to minor, against 10:1). The 3 is a pseudo-count, and its role matters more than its exact value.

Lost or added strikethrough is critical, since it makes deleted text read as live or the reverse, and a changed exponent (10^9 read as 109) is critical too. Underline is major on redline pages, where it marks inserted text, and other emphasis is minor. The cap is Equation (8) with $K = 10$ items, with minor errors not scaled. On a short contract amendment, other markup syntax and a uniform heading shift score 100, a lost bold label costs 1.7 points, and a struck fee written as plain text caps the page at 90.

Item losses alone would let a parser that ignores one kind of decoration score high on the strikethrough and underline datasets. A parser that never writes underline would score 93 on the underline dataset, and one that never writes strikethrough would score 52. These two targeted datasets (Section 3.1) therefore have full-page *decoration inventories* that list all decorated runs. On them, the parsing score becomes the F1 of decorated words over the aligned text, which counts decoration added anywhere on the page against the score, while the cap is unchanged. With simulated errors on the 253 inventory pages, an output that drops all decorations then scores 14 on strikethrough and 20 on underline.

A.3.4 Tables

The Tables metric checks that cell values are correct and sit under the right row labels and column headers, in the right table and row order. ParseBench-1 scored tables with GTRM, the mean of GriTS [22] and TableRecordMatch. Under GTRM a transposed table scores 13.2, below a table with half its rows missing, and ten wrong numbers in a 459-cell table still score 98.7.

A data cell becomes a triple of its row-label path, its column-header path and its value, $(r_1/\cdots/r_a,\ c_1/\cdots/c_b,\ v)$, keyed by *all* its row labels and column-header levels, and a table is the multiset of its triples. In Figure 8, 901 is Europe's revenue for 2024, and a 901 under 2023 states a different fact. HTML and Markdown pipe tables parse into identical grids, with spans expanded, and `<th>` or `<td>` carries no weight. A blank cell drawn as a dash is a benign variation wherever it sits: as a value, in a row label, in the corner cell or in a section row. A dash written over a value is still a wrong value.

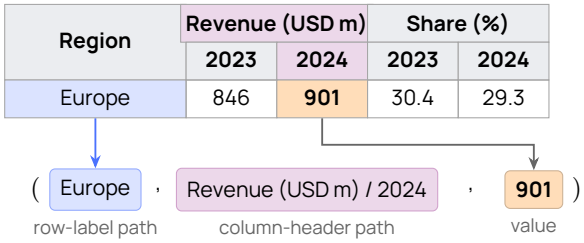


Figure 8 Each table value stays attached to its headers. The Tables metric reads a table as cell triples keyed by their row and column headers: the value 901 is keyed by its row-label path (Europe, blue) and its column-header path (Revenue (USD m) / 2024, pink), written with / as in the text. Markdown or HTML, `th` or `td`, and a rowspan or a repeated label all yield identical triples, whereas a value under the wrong year changes them.

This differs from the common table metrics in what counts as a correct value. TEDS [28] edits the HTML tree and GriTS aligns cells on a 2D grid, and both average over cells. One wrong value is diluted in this average, and a value moved under the wrong header costs little while the grid keeps its shape. TableRecordMatch matches rows as records but ignores row order and cannot represent hierarchical row labels. In the Tables metric a value is correct only under the headers that give it meaning, and the cap makes one wrong digit cost 10 points on any table. Table 1 compares it with these metrics on the Tables test suite.

Each annotated table fixes the structure: its title rows, header rows and label columns are inferred once and transferred to the output table it pairs with. The output is also read with its own inferred structure, transposed, and with its dash-only cells as blanks. The scorer keeps the reading that fits the annotations best, which preserves the parsing score of a transposed table and leaves the cap to charge the transposition. These readings do not cover all serializations. Tables are paired by shared keys, and their triples are matched one

to one with a Hungarian assignment [8] among triples with compatible labels. An output table that shares nothing with the annotations is ignored, since table annotations are not exhaustive. When an annotated table has fewer than three values, its empty markers (a dash, "n/a") count as values, so that a table of markers is still scored.

A matched pair of triples weighs its label similarity times its value similarity. The value similarity is 1 for equal values, 0.95 when only letter case differs, 0.9 for the same number written differently, 0.5 when a unit is lost and 0 when any digit differs. The parsing score is the F1 of these weights, less a penalty for rows out of order:

$$\begin{aligned} \text{content} &= \mathrm{F1}\Big(\tfrac{\sum w}{|G|},\ \tfrac{\sum w}{|O|}\Big), \\ m_{\text{table}} &= \max\big(0,\ \text{content} - 0.3\,(1 - \text{row_order})\big), \end{aligned} \tag{11}$$

where G and O are the annotated and output triples of the paired tables, $\mathrm{F1}(x, y) = 2xy/(x + y)$, and row_order is the share of annotated rows inside the longest run that the output keeps in the order given by the annotations. A missing cell and an invented one cost alike, and an output with no table scores 0.

A changed digit or a value under the wrong header is critical. A header that reads as another one is critical too, since every value below it then states a different fact: its number changed (2025 read as 2024 or as 202.5), or two headers are swapped or shifted. A reworded label is charged once per table, as a minor error when it is one typo away from the annotated label and as a major one otherwise. A transposition that keeps all header–value associations is one major error, for consumers that read a table row by row. A transposition that also loses, changes or misplaces a cell stays critical. The cap is Equation (8) with $K = 80$ annotated cells. Errors with one cause are charged once: three or more cell errors of one severity level in a column count as one column error, and so do an inserted column, a whole table in the other decimal convention (minor) and a table moved in the page order. On a small revenue table, one changed digit costs 10 points through the cap although the content F1 is 91.7.

A.3.5 Charts

The Charts metric checks that the output contains all data points of the page's charts, with the right category, series and value. A chart is any figure that plots data values, and diagrams are not yet scored (Section 6). ParseBench-1's ChartDataPointMatch checked at most ten spot values per chart, and its 4,864 rules covered 13% of the plotted values. Some rules expected a neighboring bar's value, and sparse rules cannot see an invented series, a wrong unit or a missing title.

The annotations include a complete data table per chart, with title, chart type, axis labels, unit, categories and series (Section 3.1). Since the table is complete, all cells are scored and invented data counts against precision. The scorer reads tables in any orientation, JSON blocks and bullet lists, normalizes labels, and converts values through unit suffixes, header units and scale notes. A stated unit that contradicts the annotations makes the value wrong. Rows and columns are mapped to categories and series with a Hungarian assignment, and when a parser writes several charts as one table, a row or column labeled with a chart's title belongs to that chart's panel alone.

The allowance a differs between labeled and unlabeled charts, since printed data labels are transcribed while values read off an axis are estimates. On labeled charts $a = 1\% \cdot \max(|t|, |p|)$ for annotated value t and output value p, or the chart's own relative tolerance when the annotations set one. Using $\max(|t|, |p|)$, as Python's `math.isclose` does, makes the test independent of which value comes from the annotations and gives an overshoot at most 1% more allowance than an undershoot (1.01 against 1.00 on a value of 100). On unlabeled charts $a = \max\big(1\% \cdot |t|,\ 2\% \cdot \text{range}\big)$, where the range is that of the chart's values. The 2% is our choice, of the order of human reading error on bar and line charts [2, 6], and we did not measure reading error on the charts of this benchmark. The closeness of p to t is

$$\nu(t, p) = \begin{cases} 1 & |p - t| \le a \\ \max\big(0,\ 1 - \frac{|p-t|-a}{2a}\big) & \text{otherwise,} \end{cases} \tag{12}$$

and $\nu = 0$ when the signs differ beyond the allowance, with the credit falling linearly from a to $3a$.

The parsing score is an F1 over annotated and output points, in which a matched point counts its closeness ν times its label binding β:

$$R = \frac{\sum_i \beta_i \nu_i}{n_{\text{ref}}}, \quad Pr = \frac{\sum_i \beta_i \nu_i}{n_{\text{out}}}, \quad m_{\text{chart}} = \text{F1}(R, Pr), \tag{13}$$

where β_i is 1 when the labels of annotated point i match the output's after normalization, their string similarity for a typo and lower for a rewording. Here n_{ref} and n_{out} count the annotated points and the output points read from the grids assigned to the charts.

An error between a and $3a$ is a major *value off*, and a larger one a critical *wrong value* (Figure 9). The cap is Equation (8) with $K = 10$ cells for per-cell errors such as a wrong value or a missing point. Structural errors such as a missing series, a lost legend or an invented category have a fixed cost on any page, and an omission never costs a chart more than a missing chart. On a labeled two-series bar chart, a misread bar (412.5→512.5) is a critical *wrong value* whether or not a ParseBench-1 rule checks that cell (page score 87.5). A 1.8% error (412.5→420) is a major *value off* (95.0), and a rescaled unit or abbreviated labels cost nothing, though they fail the ParseBench-1 rules.

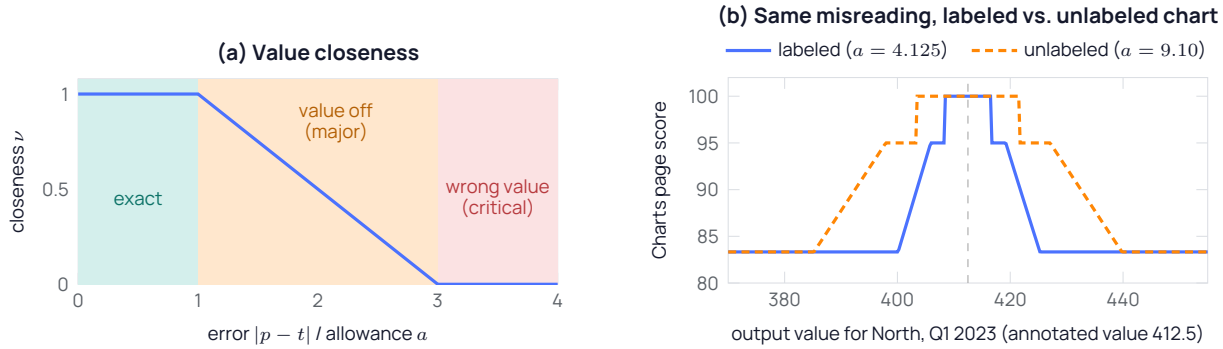


Figure 9 How the Charts metric tolerates reading error. (a) Within the allowance a, a value is exact. The allowance is $1\% \cdot \max(|t|, |p|)$ on a labeled chart and $\max(1\% \cdot |t|,\ 2\% \cdot \text{range})$ on an unlabeled one. Closeness then falls linearly to 0 at $3a$. Errors up to $3a$ are a major *value off*, and larger ones a critical *wrong value*, shaded in the colors of the severity levels. (b) Sweeping one value of a six-value, two-series bar chart. The dashed line marks the annotated value 412.5, and the legend gives a at that value. With printed labels, a value just outside a is a value off that caps the page at 95 (the flat shoulders). Further out the page score follows the parsing score m, which falls with closeness until the value counts as wrong and P reaches 5/6, about 83.3, below the critical cap of 90. An output of 420 scores 93.5 here. The same misreading scores 95.0 on an eight-value chart, where one value weighs less in m. On an unlabeled chart, where the value is read from the axis, 420 is exact.

The leaderboard scores chart pages on all their annotated cells, in *full-table mode*. The ParseBench-1 rules remain only in a sparse-rule mode, which scores recall, for pages annotated with individual data points.

A.3.6 Visual grounding

The Visual grounding metric checks that parsed content can be traced back to the page. A block the output emits, such as a paragraph, a table, a picture or a page header, must be boxed where its content is printed and must carry that content. Layout-detection evaluations, such as the COCO mean average precision used by PubLayNet and DocLayNet [10, 15, 27], ask whether the predicted boxes reproduce the annotated segmentation of the page, class by class. The ParseBench-1 Element Pass Rate (EPR) relaxes localization with overlap ratios, which lets two paragraphs written as one block pass. Because it still checks annotated elements on their own text, a paragraph written line by line scores 91.7, and paragraph-level output against line-level annotations scores 61.1 (Figure 10).

A parser that writes two paragraphs as one block, or a paragraph line by line, has chosen a granularity, and we do not count that choice as an error. The Visual grounding metric therefore lets the output's own blocks

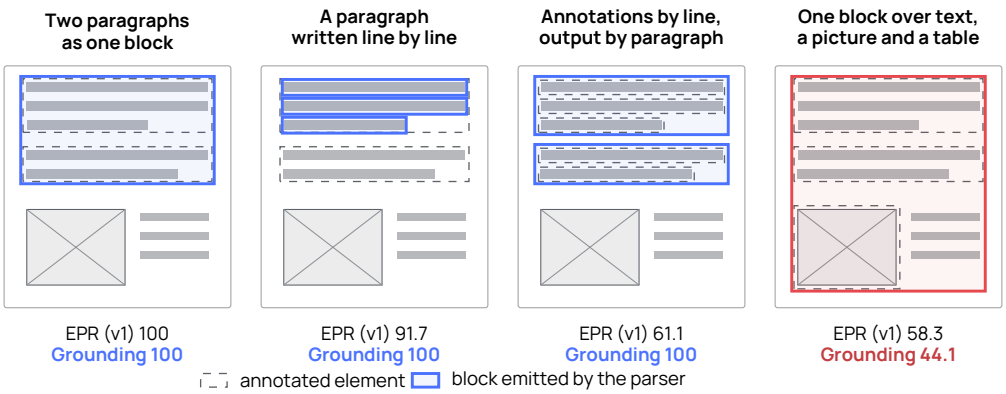


Figure 10 Visual grounding follows the output's own granularity. A synthetic page with two paragraphs, a picture and its labels, scored by the ParseBench-1 Element Pass Rate (EPR) and the Visual grounding metric. Writing two paragraphs as one block, or one paragraph line by line, is a choice of granularity: the content is boxed where it is printed and carries the right text, and the Visual grounding metric scores both 100. EPR checks each annotated element on its own text, so it penalizes the split, and penalizes a paragraph-level output against line-level annotations most. A block that lumps five elements across a picture and a table no longer locates them and is charged by both metrics (the scores are for the whole 12-element page; the drawing shows part of it).

set the granularity, measures how well they are grounded and scores all three cases 100. The annotated element and word boxes serve only to check where content is and what it says.

EPR passes an annotated element when a prediction localizes it, has its basic class and carries its text. The Visual grounding metric also gives partial credit for loose boxes, weights elements by size and charges boxes that ground nothing. On a synthetic 12-element page, 50 junk boxes keep EPR at 100%, and a box passes EPR until it is about 2.2× the element in width and height. A missed icon also costs EPR as much as a missed paragraph (Figure 11).

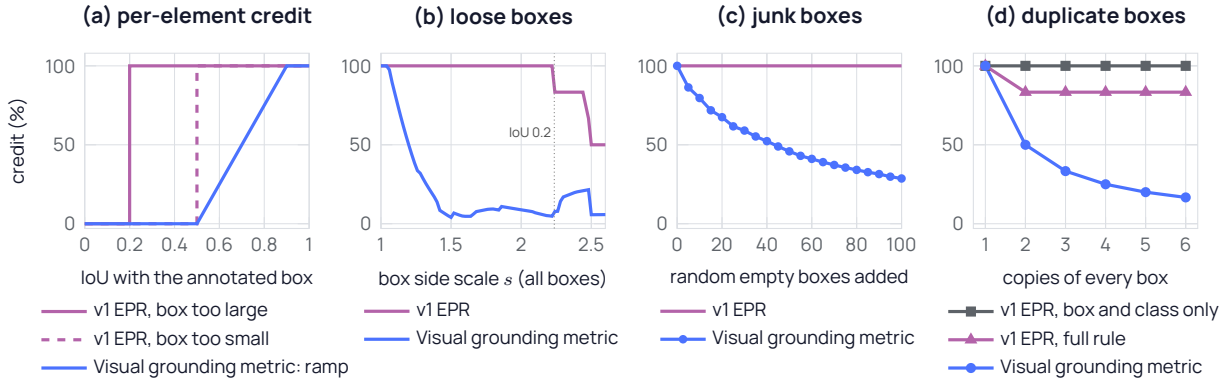


Figure 11 The ParseBench-1 Element Pass Rate (EPR) and the ParseBench-2 Visual grounding metric. The ParseBench-1 EPR stays at 100% under loose, junk and duplicate boxes, and the Visual grounding metric falls in all three cases. (a) Credit for one element against its IoU. The ParseBench-1 EPR is a step: a box that is too large passes down to IoU 0.2 and one that is too small down to 0.5 (dashed). The Visual grounding metric ramps linearly from IoU 0.5 to 0.9. (b)–(d) Page score in percent, on the y axis of (a), for a synthetic 12-element page when every box is scaled by s per side (b), empty boxes are added (c) or every box is repeated with another class (d). A box scaled by s has IoU $1/s^2$ with its element, and the dotted line in (b) marks IoU 0.2, at s of about 2.2, where the v1 EPR stops localizing elements. Past its minimum, the Visual grounding score rises a little as a grown box covers its neighbors and is compared with their hull. In (d), the full v1 rule adds the text test to the box and class test.

An annotated element earns credit s_g once a prediction *localizes* it, covering at least half of the element and lying at least 20% on it as in ParseBench-1, and the prediction's text matches the element's at a token F1 of

at least 0.8. The predicted class is not checked. The credit rises linearly from 0 at an intersection over union (IoU) of 0.5, where a box too small for its element already fails EPR, to 1 at 0.9. Since annotations are boxed tight to the ink, an IoU of 0.9 allows a box about 5% larger than the element in each dimension.

When one output block covers several consecutive annotated text elements, or several output boxes cover one, they are scored as one group. The scorer compares the group's pooled annotated text with the block's text, and the hull of the elements with the hull of the boxes, column by column. Page headers and footers are grouped in this way even when their pieces are not consecutive. Tables and pictures are matched one to one, and a table's box must cover 90% of its words. Unmatched predictions that duplicate an element, lie on blank areas or hold words already assigned to another element are *extras*.

Elements weigh $w = \sqrt{\text{area}}$, between counting and pure area, which makes a missed paragraph cost more than a missed icon while one large picture does not outweigh the text of the page. Extras enter the denominator with the same weight:

$$\text{Grounding} = \frac{\sum_{g \in \text{Ref}} w_g \, s_g}{\sum_{g \in \text{Ref}} w_g + \sum_{p \in \text{extra}} w_p}, \tag{14}$$

where Ref is the set of annotated elements and extra the set of extra predictions. Junk boxes lower the score in proportion to their size (Figure 11c).

The dimension is the mean of page scores, with no severity levels and no cap. A misplaced box changes where content is found and leaves what it says unchanged, which means there is no wrong fact to cap, although the test suite of this dimension still sets score-drop targets by severity (Appendix A.2).

We chose the IoU bounds and the $\sqrt{\text{area}}$ weights ourselves, with citation and highlighting in mind. We have not validated them against these uses or against independent annotation, and our sensitivity analysis does not vary them.

A.3.7 Forms

A *form* is a page whose printed captions label text fields, checkboxes, choices and signature lines that a person fills in. The Forms metric checks that a parsed form has the right value in all its fields and the right state in all its checkboxes, attached to the right caption, whatever the output's syntax (Figure 12). It reads the page Markdown, and a provider's separate key–value output counts only if it is rendered into that Markdown (Appendix C.1).

Value correctness. We check whether each field's value is read correctly. For a textbox, this means reading the entered text correctly. For a checkbox, it means recording whether the box is checked or unchecked.

Semantic binding. Textboxes and checkboxes have labels, or captions, that indicate what their values or states mean. Semantic binding checks whether each recovered value or state is associated with the correct label. For example, the value *Ingrid Sorensen* belongs to the label *Full Name* in Figure 12. A model may read *Ingrid Sorensen* correctly but incorrectly associate it with *Date of birth*. Conversely, it may preserve the correct label but misread the value. Value correctness and semantic binding can therefore fail independently. Printed identifiers, section headings and table context can further distinguish fields with similar labels.

We develop a scoring system that measures content correctness while remaining robust to common variations in output formatting. Figure 12b illustrates the same form transcribed as key–value lines or as a two-column table. Both outputs preserve the same values, checkbox states and label associations, so both receive the same score by design. The scorer also treats common checkbox symbols and equivalent date and amount formats alike. It checks each value together with its label, so correctly transcribed text under the wrong label is still an error.

Error severity. Forms uses the same per-item loss weights and severity-cap framework as Formatting (Appendices A.3.1 and A.3.3). For Forms, errors are assigned a severity level by their effect on the information recovered from a field:

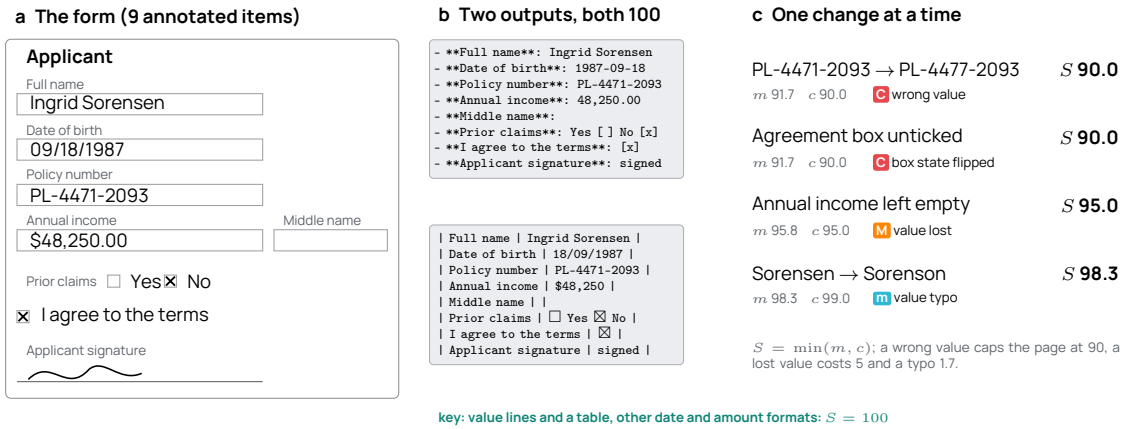


Figure 12 The Forms metric reads values by their captions, whatever the syntax, and charges a wrong value more than a lost one. (a) A synthetic form section with nine scored items: five text fields, one blank, a Yes/No choice answered No, a ticked box and a signed line. (b) Written as `key: value` lines or as a table with another date and amount format, the output scores 100. (c) Single changes and their Forms scores. m is the parsing score across fields, c is the severity cap and S is the final score.

- **Critical:** a wrong value, a value attached to the wrong label, a value invented in a blank field, or a checkbox whose state is flipped.
- **Major:** a missing value, an omitted tick or a lost signature.
- **Minor:** a typo or an omitted blank field.
- **Benign variation:** an equivalent representation of the same content, such as key–value lines instead of a two-column table.

These levels distinguish an incorrect fact from missing content and a small transcription error, and set both apart from a formatting difference.

Figure 12c shows how changing one entry at a time affects the score while the rest of the form remains correct. A wrong policy number or a flipped checkbox reduces the score from 100 to 90.0, an omitted annual income reduces it to 95.0, and a typo in the surname reduces it to 98.3.

Table 1 compares the metric with the ParseBench-1 form metrics on the Forms test suite.

B Metric Evaluation

B.1 Severity Test Suites: Scenario Scores

Tables 11 to 16 give the score drop of every metric on every scenario of the six severity test suites. A cell compares the metric's score drop with the scenario's target, the drop being the score of the correct output minus that of the perturbed one, averaged over the sample pages that the scenario changes: ✓ the target is met; ∼ the drop is at least 1 point but below the target, or above 25 points for a minor error; ✗ the drop is below 1 point, or at least 0.5 points on a benign variation. Debatable scenarios are not graded, and their cells show a grey dash (–). The targets are at least 10 points for a critical error, at least 5 for a major error, 1 to 25 for a minor error and under 0.5 for a benign variation; debatable scenarios are listed but not graded. Columns are ordered by policy-compliance summary (PCS), given in the last row, and · marks a metric with no value for the scenario.

We adapt FinTOC [3] to page headings and use our generic exact-match field F1 baseline for Forms.

Table 11 Text test suite: verdict of each of the 31 metrics on each of the 51 scenarios, on 2 sample pages. Sev.: severity rating of the level (critical 90, major 50, minor 25, benign 0). ✓ target met, ∼ partly met, ✗ missed.

Perturbation	Sev.	1. **ParseBench-2 Text**	2. Order rule pass rate	3. Text 1 − NED	4. Ordered critical tokens F1	5. Move-tolerant fidelity (component average)	6. Unit-test pass rate	7. Content Faithfulness	8. Text + reading order	9. CCT accuracy	10. Bag-of-words recall	11. Flexible character accuracy	12. ISRI word accuracy	13. NID (Indel similarity)	14. Text 1 − NED (MGAM matching)	15. Next: reading-order component	16. Word order (Kendall)	17. Reading order 1 − ED	18. BLEU-4	19. Heuristic score	20. Word accuracy (1 − WER)	21. Word edit distance	22. METEOR	23. chrF (character n-gram F)	24. Unique-word F1	25. Tokens found	26. Word F1 (multiset)	27. Full-page OCR score	28. 1 − tokens added	29. 1 − edit distance	30. Character accuracy (1 − CER)	31. Document parsing STEDS
CRITICAL																																
Output cut off	90	✓	✓	✓	✓	✓	∼	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
Negation flipped	90	✓	∼	✗	✗	∼	∼	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	∼	✗	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Paragraphs dropped	90	✓	✓	✓	∼	∼	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
Wrong digits	90	✓	∼	✗	✓	✓	✗	∼	✗	✗	∼	✗	✗	✗	✗	∼	✗	✗	✗	✗	∼	∼	∼	∼	∼	∼	∼	∼	∼	✗	✗	✗
Date with day and month swapped	90	✓	∼	✗	∼	∼	∼	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Two values swapped between clauses	90	✓	∼	✗	∼	∼	∼	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Read straight across columns	90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	∼	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
Invented sentences	90	✓	✗	✓	∼	∼	✗	∼	∼	∼	✗	✓	✗	∼	✓	✗	✗	✗	✓	✗	✓	✓	∼	∼	∼	∼	✗	∼	✓	✓	✓	✗
Unit changed	90	✓	✗	✗	✗	∼	✗	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	∼	✗	∼	∼	∼	∼	∼	∼	∼	∼	∼	✗	✗	✗
OCR confusion in an identifier	90	✓	∼	✗	∼	∼	✗	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Hallucinated summary sentence	90	✓	✗	∼	∼	∼	✗	✗	∼	∼	✗	∼	✗	∼	∼	∼	✗	✗	∼	✗	∼	∼	∼	✗	∼	∼	✗	∼	∼	∼	∼	∼
Names and places misspelled	90	✓	✗	✗	✗	✗	∼	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Email address corrupted	90	✓	✗	✗	∼	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
MAJOR																																
Real-word substitutions	50	✓	✓	✓	✗	∼	✓	✓	∼	∼	∼	∼	∼	∼	∼	✗	✗	✗	✓	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	✗
OCR character confusions	50	✓	✓	∼	∼	✓	✓	✓	∼	∼	✓	∼	✓	∼	∼	✗	✗	✗	✓	∼	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼	∼	✗
Words dropped	50	✓	✓	✓	∼	∼	✓	✓	∼	✓	✓	✓	✓	∼	✓	✗	✗	✗	✓	∼	✓	✓	✓	✓	∼	✓	✓	✓	✓	✓	✓	✗
% and currency signs dropped	50	✓	✗	✗	✗	∼	✗	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	∼	∼	✗	✗	✗	✗	∼	✗	✗	✗	✗	✗	✗	✗
Paragraph repeated	50	✓	✗	✓	✓	✓	✗	✓	✓	✓	✗	✓	✗	✓	✓	✗	✗	✗	✓	✗	✓	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓
Random words inserted	50	✓	✓	∼	✗	✗	✓	✓	∼	∼	✗	∼	✗	∼	∼	∼	✗	✗	✗	✗	∼	∼	∼	✗	∼	✗	✗	∼	∼	∼	∼	✗
Phrase omitted	50	✓	✓	∼	✗	∼	✓	✓	∼	∼	∼	∼	∼	∼	∼	∼	✗	✗	✗	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	✗
Sentence cut at a column break	50	✓	✓	∼	∼	∼	∼	✓	∼	∼	∼	∼	∼	∼	∼	✗	✗	✗	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	✗	∼	✗
Paragraphs shuffled	50	✓	✓	✗	✓	✗	✓	∼	✓	✓	✗	✓	✓	✗	✓	∼	✓	∼	✓	∼	∼	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✓
Footnote spliced mid-sentence	50	✓	✓	✓	∼	∼	✓	∼	∼	✓	✗	✗	∼	∼	✓	∼	∼	✗	∼	∼	∼	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	∼
Adjacent words swapped	50	✓	✓	∼	✗	✓	✓	✓	∼	∼	✗	∼	∼	∼	∼	✗	✗	✗	✓	∼	∼	∼	∼	✗	∼	✗	✗	✗	✗	✗	∼	✗
Headings dropped	50	✓	✓	∼	✗	∼	✓	✓	∼	∼	∼	∼	∼	∼	∼	✗	✗	✗	✓	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	✗	∼	✓
One sentence repeated	50	✓	✗	✓	∼	∼	✗	∼	∼	✓	✗	✓	✗	∼	✓	✗	✗	✗	✓	✗	✓	✓	✓	✗	∼	✗	✗	∼	✓	✓	✓	✗
List items' contents swapped	50	✓	✓	✓	∼	✗	✗	∼	∼	∼	✗	✗	∼	∼	∼	∼	∼	✗	✓	✗	∼	∼	✗	✗	✗	✗	✗	✗	✗	✗	∼	✗
Table of contents duplicated	50	✓	✗	✗	✓	✓	✗	∼	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓	✗	∼	∼	✗	✗	✓	✓	✗	✓	✓
Sentences swapped in a paragraph	50	✓	✓	✓	✓	∼	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	∼	✗	✓	✓	✓	✓	✗	∼	✗	✗	✗	✓	✗	✓	✓	✗
MINOR																																
Spelling typos	25	✓	✗	✗	✗	✓	✓	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
Look-alike characters	25	✓	✗	✓	✗	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗
Date format changed	25	✓	✓	✓	✗	✓	✓	✓	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗
Thousands separators removed	25	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✗	✗
Parentheses dropped	25	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
BENIGN																																
Extra blank lines	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
Random emphasis added	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Paragraphs run together	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Hard line breaks	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
Different Markdown style	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Table as HTML	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗
Typographic characters	0	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

Text test suite, continued

Perturbation	Sev.	1. **ParseBench-2 Text**	2. Order rule pass rate	3. Text 1 − NED	4. Ordered critical tokens F1	5. Move-tolerant fidelity (component average)	6. Unit-test pass rate	7. Content Faithfulness	8. Text + reading order	9. CCT accuracy	10. Bag-of-words recall	11. Flexible character accuracy	12. ISRI word accuracy	13. NID (Indel similarity)	14. Text 1 − NED (MGAM matching)	15. Next: reading-order component	16. Word order (Kendall)	17. Reading order 1 − ED	18. BLEU-4	19. Heuristic score	20. Word accuracy (1 − WER)	21. Word edit distance	22. METEOR	23. chrF (character n-gram F)	24. Unique-word F1	25. Tokens found	26. Word F1 (multiset)	27. Full-page OCR score	28. 1 − tokens added	29. 1 − edit distance	30. Character accuracy (1 − CER)	31. Document parsing STEDS
Hyphenated line breaks kept	0	✓	✗	✓	✓	✓	✗	✗	✓	✗	✗	✗	✗	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
DEBATABLE																																
Columns in reverse order	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Words split or joined	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Punctuation dropped	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Accents stripped	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
List items run together	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Line-end hyphenation kept	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Table flattened to lines	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Headings in capitals	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Header and footer dropped	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PCS		95	36	32	28	25	25	24	18	17	15	13	12	10	9	8	3	2	1	−5	−8	−14	−15	−20	−20	−22	−24	−31	−33	−49	−51	−55

Table 12 Formatting test suite: verdict of each of the 17 metrics on each of the 55 scenarios, on 4 sample pages. Sev.: severity rating of the level (critical 90, major 50, minor 25, benign 0). ✓ target met, ∼ partly met, ✗ missed.

Perturbation	Sev.	1. **ParseBench-2 Formatting**	2. formatting_v2 cap only	3. formatting_v2 proportional only	4. Style-token F1	5. Heading F1, exact level	6. Normalized title accuracy	7. FormatTest pass rate	8. STEDS	9. Normalized text styling	10. Title F1 / level accuracy harmonic mean	11. All formatting rules pass rate	12. Heading TEDS	13. Semantic Formatting	14. MHS heading tree	15. Text 1 − NED (cleaned)	16. Tag-sequence 1 − NED	17. Raw Markdown 1 − NED
CRITICAL																		
Strikethrough lost	90	✓	✓	∼	∼	✗	✗	✗	✗	✓	✗	✓	✗	∼	✗	✗	✗	∼
Exponent lost ($10^9 \rightarrow 109$)	90	✓	✓	∼	∼	✗	✗	✗	✗	∼	✗	∼	✗	∼	∼	∼	∼	∼
Live text struck through	90	✓	✓	∼	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Digit turned into exponent	90	✓	✓	∼	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	∼	∼
Exponent written as subscript	90	✓	✓	✓	∼	✗	✗	✗	✗	∼	✗	✓	✗	∼	✗	∼	∼	✗
MAJOR																		
All headings turned into body text	50	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	∼
Insertion underline lost (redline)	50	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	∼	✓	∼
One heading turned into body text	50	✓	✓	∼	∼	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	∼	✗
Parent and child levels swapped	50	✓	✓	∼	✓	✓	✓	✗	✗	✗	✓	✗	✓	∼	✗	✗	∼	✗
List items run together	50	✓	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	∼
All headings at one level	50	✓	✓	∼	✓	✓	✓	✗	✗	✗	✓	∼	✓	✓	✗	✗	∼	✗
Body paragraph turned into heading	50	✓	✓	∼	✓	✓	✗	✗	✓	✗	✓	✗	✓	✗	✓	✗	∼	✗
List nesting flattened	50	✓	✓	∼	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	∼	✗
MINOR																		
All headings shown as bold lines	25	✓	✓	✓	✓	∼	✓	∼	∼	✗	∼	✓	∼	✓	∼	✗	✓	✓
All bold and italic stripped	25	✓	✓	✓	✓	✗	✓	∼	✗	∼	✗	∼	✗	✓	✗	✗	∼	✓
Heading shown as a bold line	25	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✗	∼	✓	✓	✗	✓	✗
List markers dropped	25	✓	✓	✓	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
Footnote and unit superscripts lost	25	✓	✓	✓	✓	✗	✗	✗	✗	✓	✗	✓	✗	✗	✓	✓	✓	✓
Numbered list turned into bullets	25	✓	✓	✓	∼	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓
Code block emitted as text	25	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✓	✓
Bold lost	25	✓	✓	✓	✓	✗	✓	✓	✗	✓	✗	✓	✗	✓	✗	✗	✓	✗
One bold phrase lost	25	✓	✓	✗	✓	✗	✗	✓	✗	✓	✗	✓	✗	✓	✗	✗	✓	✗
One heading one level too deep	25	✓	✓	✓	✓	∼	✓	✗	✗	✓	✓	✗	∼	✓	✗	✗	✗	✗
Short line promoted to heading	25	✓	✓	✗	✓	✓	✗	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗
Subscripts lost ($CO_2 \rightarrow$ CO2)	25	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓
Underline lost	25	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗
Italic lost	25	✓	✓	✓	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗
Whole paragraph in bold	25	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
List item split in two	25	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗
Paragraph turned into a bullet	25	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
Item nested under its sibling	25	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
Bold added to plain words	25	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
Bold rendered as italic	25	✓	✓	✗	✓	✗	✗	✓	✗	✓	✗	✓	✗	✓	✗	✗	✓	✗
Highlight lost	25	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✓	✓
Bold span extended by a word	25	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Italic added to plain words	25	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
Inline code lost	25	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
BENIGN																		
Ordinal superscript lost (1st)	0	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✗	✓	✗	✓	✗	✗	✗
Everything written as HTML	0	✓	✓	✓	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
__bold__ and _italic_	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗
• bullets and 1) numbers	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗

Formatting test suite, continued

Perturbation	Sev.	1. **ParseBench-2 Formatting**	2. formatting_v2 cap only	3. formatting_v2 proportional only	4. Style-token F1	5. Heading F1, exact level	6. Normalized title accuracy	7. FormatTest pass rate	8. STEDS	9. Normalized text styling	10. Title F1 / level accuracy harmonic mean	11. All formatting rules pass rate	12. Heading TEDS	13. Semantic Formatting	14. MHS heading tree	15. Text 1 − NED (cleaned)	16. Tag-sequence 1 − NED	17. Raw Markdown 1 − NED
Unicode superscripts and subscripts	0	✓	✓	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
LaTeX exponents and indices	0	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
Setext headings	0	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗
All heading levels one deeper	0	✓	✓	✓	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓
Closing #s and bold inside headings	0	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✗	✓	✗	✓	✗	✗
Spacing and indentation	0	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗	✗
Punctuation outside the bold	0	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓	✗	✓	✗	✓	✓	✓	✓
Bold split word by word	0	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓	✗	✓	✗	✓	✓	✗	✗
Output wrapped in “‘markdown	0	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
HTML entities	0	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗
Every item numbered 1.	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DEBATABLE																		
Struck text omitted	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Bold label line promoted to heading	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Second half of the page missing	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PCS		100	100	23	21	13	5	−14	−14	−18	−19	−19	−20	−26	−40	−44	−53	−80

Table 13 Tables test suite: verdict of each of the 28 metrics on each of the 62 scenarios, on 5 sample pages. Sev.: severity rating of the level (critical 90, major 50, minor 25, benign 0). ✓ target met, ∼ partly met, ✗ missed.

Perturbation	Sev.	1. **ParseBench-2 Tables**	2. Table test pass rate	3. TableRecordMatch	4. TEDS-S	5. GriTS-Top	6. GriTS-Con	7. Table size match	8. T-LAG (cell-neighbour graph F1)	9. GTRM composite	10. Position accuracy (text adaptation)	11. Structure BLEU-4	12. Merged-cell F1 (MCD)	13. Composite structure acc	14. AccCon (exact table)	15. Adjacency F1	16. JSON accuracy	17. Cell accuracy (TCE)	18. Row and column F1 (RCE)	19. Relation F1, exact	20. TEDS (v1.6 normalization)	21. TEDS (normalized)	22. Shifted content accuracy	23. GriTS-Con	24. TEDS	25. Relation F1, chrF	26. 1 − table edit distance	27. Alignment score	28. Bigram Jaccard
CRITICAL																													
Table missing entirely	90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Column headers swapped	90	✓	✓	✓	✗	✗	∼	✗	∼	✓	∼	✗	✗	∼	✓	∼	✓	∼	∼	✓	✓	∼	∼	∼	✗	✓	✗	∼	✗
Headers shifted one column	90	✓	✓	✓	✗	✗	∼	✗	∼	✓	∼	∼	✗	∼	✓	∼	✓	∼	∼	✓	∼	∼	∼	∼	✗	✓	∼	∼	∼
Sign or comparator lost	90	✓	✓	∼	✗	✗	∼	✗	✓	∼	∼	✗	✗	✗	✓	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	✗	∼	∼
Decimal point misplaced	90	✓	∼	∼	✗	✗	✗	✗	∼	∼	∼	✗	✗	✗	✓	∼	∼	∼	∼	∼	✗	✗	✗	✗	✗	∼	∼	✗	∼
Minus signs added	90	✓	✓	✓	✗	✗	∼	✗	✓	∼	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	∼	∼	∼	∼	∼	∼	✗	✗	∼
Wrong digits	90	✓	✓	✓	✗	✗	∼	✗	✓	∼	∼	✗	✗	✗	✓	✓	∼	∼	∼	✓	∼	∼	∼	∼	∼	∼	✗	∼	∼
Decimal point inserted	90	✓	∼	∼	✗	✗	✗	✗	∼	∼	∼	✗	✗	✗	✓	∼	∼	∼	∼	∼	✗	✗	✗	✗	✗	✗	✗	∼	∼
Wrong year in a column header	90	✓	✓	✓	✗	✗	✗	✗	∼	✓	∼	✗	✗	✗	✓	∼	✓	∼	∼	∼	✓	✗	✗	✗	✗	✗	✓	✗	∼
Empty cell skipped, values shift left	90	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Rows missing	90	✓	✓	✓	✓	∼	∼	✓	∼	✓	✓	✓	∼	∼	✓	✓	✓	✓	✓	✓	∼	✓	✓	✓	✓	✓	✓	✓	✓
Values swapped within a row	90	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✗	✗	∼	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼	✗
Values swapped between rows	90	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✗	✗	∼	✓	✓	✓	✓	✓	✓	✓	∼	∼	∼	∼	∼	∼	∼	✗
Row labels swapped	90	✓	∼	∼	✗	✗	∼	✗	✓	∼	∼	✗	✗	∼	✓	✓	✓	∼	∼	∼	✓	∼	∼	∼	∼	✓	∼	∼	✗
Values shifted one row	90	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Two values swapped within a row	90	✓	✓	∼	✗	✗	∼	✗	✓	∼	∼	✗	✗	∼	✓	✓	∼	∼	∼	✓	∼	∼	∼	∼	∼	∼	∼	∼	✗
Footnote mark glued to a number	90	✓	✓	∼	✗	✗	✗	✗	∼	∼	∼	✗	✗	∼	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	∼
Plausible row invented	90	✓	∼	✓	✓	∼	∼	✓	✓	∼	∼	∼	✗	∼	✓	✓	✓	∼	∼	∼	✓	✓	✓	∼	✓	∼	∼	∼	∼
Last row (total) dropped	90	✓	✓	✓	✓	∼	∼	✓	∼	✓	✓	✓	∼	✗	✓	✓	✓	✓	∼	∼	∼	✓	✓	✓	✓	✓	∼	✗	✓
Column missing	90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Header rows missing	90	✓	✓	✓	✓	∼	∼	✓	∼	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
OCR character confusions	90	✓	✓	✓	✗	✗	∼	✗	✓	∼	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	∼	✓	✓	✓	✓	∼	∼
Adjacent cells merged	90	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	∼	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Empty cells filled in	90	∼	∼	✓	✗	✗	∼	✗	✓	✓	✗	∼	✗	✗	✓	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	✗	∼	∼
European number format	90	✓	✓	✓	✗	✗	∼	✗	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	∼	✓	✓	✓	∼	∼	∼
Table transposed	90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Invented extra table	90	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
MAJOR																													
Two rows merged	50	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Values missing	50	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Merged-cell spans lost	50	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Two tables merged into one	50	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Units dropped	50	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	∼	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Values rounded	50	✓	✓	✓	✗	✗	∼	✗	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	∼	✓	∼	✓	∼	∼	∼
Table split into several tables	50	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
Group labels missing	50	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼	✓	✓
One cell split in two	50	✓	✓	✓	∼	✓	✓	✓	✓	✓	✓	∼	✗	∼	✓	✓	✓	✓	✓	✓	✓	∼	∼	∼	✓	∼	∼	∼	✗
Row label wrapped into a new row	50	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼
Long cells truncated	50	✓	✓	✓	✗	✗	∼	✗	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	∼	∼	✓	∼	∼	∼	∼	✓
Two header rows joined into one	50	✓	✓	✗	✓	✓	✓	✓	✓	∼	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼	✓	✓
Body rows reordered	50	✓	✓	✗	∼	∼	✓	✗	✓	✓	∼	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	∼
Two columns swapped	50	✓	✓	✗	✗	✗	✓	✗	✓	✓	✓	✗	✗	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✗
Title row dropped	50	✓	✗	✗	∼	∼	✗	✓	∼	✗	✓	∼	✓	✓	✓	∼	✓	✓	✓	✓	✓	∼	∼	✓	∼	∼	✓	✓	✓
Row repeated	50	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓
Row labels abbreviated	50	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✗	✗	∼	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Tables test suite, continued

Perturbation	Sev.	1. **ParseBench-2 Tables**	2. Table test pass rate	3. TableRecordMatch	4. TEDS-S	5. GriTS-Top	6. GriTS-Con	7. Table size match	8. T-LAG (cell-neighbour graph F1)	9. GTRM composite	10. Position accuracy (text adaptation)	11. Structure BLEU-4	12. Merged-cell F1 (MCD)	13. Composite structure acc	14. AccCon (exact table)	15. Adjacency F1	16. JSON accuracy	17. Cell accuracy (TCE)	18. Row and column F1 (RCE)	19. Relation F1, exact	20. TEDS (v1.6 normalization)	21. TEDS (normalized)	22. Shifted content accuracy	23. GriTS-Con	24. TEDS	25. Relation F1, chrF	26. 1 − table edit distance	27. Alignment score	28. Bigram Jaccard
Column duplicated	50	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MINOR																													
Column headers reworded	25	✓	✓	≈	✗	✗	✓	✗	✓	≈	✓	✗	✗	✗	≈	✓	≈	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Header repeated mid-table	25	✓	✓	✓	✓	✓	✓	≈	✓	✓	≈	✓	✓	✓	≈	✓	✓	≈	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Empty row inserted	25	✓	✓	✗	✓	✓	✓	≈	✓	✓	≈	✓	✓	≈	≈	✓	≈	≈	≈	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Empty column inserted	25	✓	✗	✓	✓	✓	✓	≈	✓	✓	✗	✓	✗	✗	≈	✗	≈	✗	✓	✗	✓	✓	✗	✓	✓	✗	✓	✗	✓
Number format changed	25	✓	≈	✓	✗	✗	✓	✗	≈	✓	≈	✗	✗	✓	≈	≈	≈	≈	≈	≈	✓	✓	✓	✓	✓	≈	✓	✓	✓
Trailing zeros dropped	25	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	≈	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗	✓	✗	✓	✓
Letter case changed	25	✓	✓	✗	✗	✗	✓	✗	≈	✓	✓	✗	✗	✗	≈	✗	≈	✗	✓	✓	≈	✓	✓	✓	✓	✓	≈	✓	≈
Currency symbol moved	25	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✗	✗	✗	≈	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Punctuation dropped inside cells	25	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	≈	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓
BENIGN																													
Extra HTML markup (thead, styles, bold)	0	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
Title above the table	0	✓	✓	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Empty cells as dashes	0	✓	✗	✓	✓	✓	✗	✓	✓	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Typographic variants	0	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Bold row labels	0	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
DEBATABLE																													
Markdown pipe table	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Header cells not marked as headers	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Merged cells repeated in every position	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PCS		97	20	17	12	9	7	6	3	−4	−8	−15	−16	−20	−22	−26	−35	−36	−37	−39	−39	−39	−40	−44	−44	−47	−50	−50	−56

Table 14 Charts test suite: verdict of each of the 20 metrics on each of the 53 scenarios, on 5 sample pages. Sev.: severity rating of the level (critical 90, major 50, minor 25, benign 0). ✓ target met, ∼ partly met, ✗ missed.

Perturbation	Sev.	1. **ParseBench-2 Charts**	2. ChartDataPoint pass rate	3. EpiCurveSimilarity	4. 1 − A-MAPE	5. SCRM slight	6. SCRM strict	7. SCRM high	8. ChartOCR score	9. RMS-TBE-F1-Sig	10. RMS-TBE-F1	11. Metric 6b	12. Point F1 (2% box)	13. Normalized mapping similarity	14. RNSS	15. (1 − nMAE) × coverage	16. Relaxed accuracy (QA by lookup)	17. RMS F1	18. Row IoU (medium tolerance)	19. 0.5 RMS + 0.5 TEDS′	20. Chart parsing TEDS
CRITICAL																					
Scale lost	90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Order of magnitude off	90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼	✓	✓	✓	✓	✓	✓	✓	✓	∼
Negative signs lost	90	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼
Digit typo	90	✓	✓	✓	∼	✓	✓	✓	✓	✓	✓	∼	✓	✓	∼	✓	✓	✓	✓	∼	✗
Series names swapped	90	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	∼
Values shifted one category	90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	∼
Shares instead of values	90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼
Cumulative values read	90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼
Pie slice dropped, rest renormalized	90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓
Wrong unit stated	90	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Two series summed into one	90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Secondary axis read on the primary scale	90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼
Values copied from the neighbouring chart	90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼
Values 8–15% off	90	✓	✓	✓	∼	✓	✓	✓	∼	✓	✓	∼	✓	∼	✓	∼	✓	✓	✓	✓	∼
Values swapped between series	90	✓	∼	✓	∼	✓	✓	✓	✗	✓	✓	∼	✗	∼	✗	∼	✓	✓	✓	∼	∼
Values swapped between categories	90	✓	✓	✓	∼	✓	✓	✓	∼	✓	✓	✗	✓	∼	✗	∼	✓	✓	✓	∼	∼
Category labels swapped	90	✓	✓	✓	∼	✗	✓	✗	✓	✓	✓	∼	✓	∼	✗	∼	✓	✓	✓	✓	∼
Value copied from a neighbour	90	✓	✓	✓	∼	✓	✓	∼	∼	∼	✓	∼	✓	∼	∼	∼	∼	∼	✓	∼	✗
Legend lost	90	∼	✓	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✓	✗	✗	✓	✓	✓	✓	∼
Invented series	90	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓
All values one gridline too high	90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼
Invented category	90	∼	✗	✓	✓	✗	✓	✗	✓	∼	∼	∼	∼	∼	✗	∼	✗	∼	✓	∼	✓
MAJOR																					
Series missing	50	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
Category missing	50	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
Points missing	50	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	∼
Title dropped	50	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	∼	✗
Wrong title	50	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	∼	✗	✗	✗
Category labels truncated	50	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Series repeated	50	✓	✗	✗	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓
Row repeated	50	✓	✗	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	∼	✗
Total row added	50	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓
MINOR																					
Ranges instead of values	25	✓	∼	✓	✗	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	✗	∼	∼	∼	∼	✓
Category axis mislabeled	25	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	∼	✗	✓
Series names abbreviated	25	✓	∼	∼	✗	∼	∼	∼	✗	∼	∼	✓	✗	∼	✗	✗	∼	∼	∼	∼	✓
BENIGN																					
Rescaled, unit in header	0	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Units written in cells	0	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
Table transposed	0	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
Series in another order	0	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
Categories reversed	0	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
Trendline left out	0	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
Decimal commas	0	✓	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Charts test suite, continued

Perturbation	Sev.	1. **ParseBench-2 Charts**	2. ChartDataPoint pass rate	3. EpiCurveSimilarity	4. 1 − A-MAPE	5. SCRM slight	6. SCRM strict	7. SCRM high	8. ChartOCR score	9. RMS-TBE-F1-Sig	10. RMS-TBE-F1	11. Metric 6b	12. Point F1 (2% box)	13. Normalized mapping similarity	14. RNSS	15. (1 − nMAE) × coverage	16. Relaxed accuracy (QA by lookup)	17. RMS F1	18. Row IoU (medium tolerance)	19. 0.5 RMS + 0.5 TEDS′	20. Chart parsing TEDS
Negatives in parentheses	0	✓	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
Hedged value	0	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
Value and share in one cell	0	✓	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
HTML table	0	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
One table per series	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✗	✗
JSON instead of a table	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Bullet list instead of a table	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
DEBATABLE																					
Small reading noise	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Rounded to whole numbers	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Category labels reworded	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Values snapped to gridlines	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Labels misspelled	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PCS		91	27	23	17	6	−1	−1	−3	−9	−10	−12	−14	−14	−17	−17	−18	−20	−25	−37	−45

Table 15 Forms test suite: verdict of each of the 14 metrics on each of the 71 scenarios, on 6 sample pages. Sev.: severity rating of the level (critical 90, major 50, minor 25, benign 0). ✓ target met, ∼ partly met, ✗ missed.

Perturbation	Sev.	1. **ParseBench-2 Forms**	2. forms_v2 cap only	3. forms_v2 proportional only	4. Value presence	5. Field F1 (exact)	6. ANLS per field	7. 1 - NED per field	8. Entity-linking F1	9. Checkbox accuracy	10. Page text 1 - NED	11. rule_form_field_pass_rate	12. form_kv_md_core_kv_f1	13. form_bench_text_value_f1	14. form_bench_text_attribution_f1
CRITICAL															
Digit misread in a value	90	✓	✓	∼	✓	∼	✗	✗	∼	✗	✗	∼	∼	∼	∼
Decimal point lost in an amount	90	✓	✓	∼	∼	∼	✗	✗	∼	✗	∼	✗	∼	∼	∼
Two digits of an ID transposed	90	✓	✓	∼	∼	∼	✗	✗	∼	✗	✗	∼	∼	∼	∼
Values of two fields swapped	90	✓	✓	∼	✗	✓	∼	∼	∼	✗	∼	∼	∼	∼	∼
Values shifted to the next field	90	✓	✓	∼	∼	✓	✓	✓	✓	✗	∼	✓	✓	✓	✓
Repeated block's values swapped	90	✓	✓	✓	✗	✓	✗	✗	✗	✗	∼	✗	✓	✓	✓
Text value misread as another word	90	✓	✓	∼	∼	∼	∼	∼	∼	✗	∼	∼	∼	∼	∼
Selected option changed	90	✓	✓	∼	✗	✗	✗	✗	✗	✓	✗	∼	∼	✗	✗
Checkbox state flipped	90	✓	✓	∼	✗	✗	✗	✗	✗	✓	✗	∼	∼	✗	✗
Every option of a choice ticked	90	✓	✓	∼	✗	✗	✗	✗	✗	✓	✗	∼	∼	✗	✗
Value invented in a blank field	90	✓	✓	∼	✗	✗	∼	∼	✓	✗	∼	∼	∼	∼	∼
Table column slipped a row	90	✓	✓	∼	∼	✗	✗	✗	✗	✗	✗	✗	✗	·	·
Signature claimed on an unsigned line	90	✓	✓	∼	✗	✗	✗	✗	✓	✗	∼	∼	∼	·	·
MAJOR															
All values dropped (blank template)	50	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
Value dropped (caption kept)	50	✓	✓	∼	✓	✓	✓	✓	∼	✗	∼	∼	✓	✓	✓
Filled field missing	50	✓	✓	∼	✓	✓	✓	✓	∼	✗	✓	∼	∼	∼	∼
Ticked option missing	50	✓	✓	∼	✗	✗	✗	✗	✗	✓	∼	∼	∼	✗	✗
Chosen option unticked (no choice shown)	50	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	∼	∼	✗	✗
Ticks dropped (option text kept)	50	✓	✓	∼	✗	✗	✗	✗	✗	✓	✗	∼	∼	✗	✗
Signature lost	50	·	·	·	·	·	·	·	·	·	·	·	·	·	·
Caption repeated as the value	50	✓	✓	∼	✓	✓	✓	✓	✓	✗	∼	∼	✓	✓	✓
Values without their captions	50	✓	✓	∼	✗	✓	✓	✓	∼	✗	∼	✓	✓	✓	✓
Value truncated	50	✓	✓	∼	✓	✓	∼	∼	∼	✗	∼	∼	∼	∼	∼
Choice group written without boxes	50	✓	✓	∼	✗	✗	✗	✗	✗	✓	✗	∼	∼	∼	∼
Amount rounded to whole units	50	✓	✓	✗	∼	∼	✗	✗	∼	✗	✗	∼	∼	∼	∼
Value marked [illegible]	50	✓	✓	∼	✓	✓	✓	✓	✓	✗	∼	✓	✓	✓	∼
MINOR															
Typo in a text value	25	✓	✓	✗	✓	✓	✗	✗	✓	✗	✗	✓	✓	✓	✓
Form table flattened to lines	25	✓	✓	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	·	·
Extra text after a value	25	✓	✓	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
Blank fields omitted	25	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✗
Unticked boxes omitted	25	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓	✗	✗
Empty boxes dropped (option text kept)	25	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✓	✗	✗
Caption garbled	25	✓	✓	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✓	✓
Form table rows in another order	25	✓	✓	✓	✗	✓	✗	✗	✗	✗	∼	✗	✗	·	·
BENIGN															
Bold key: value paragraphs	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
Plain Caption: value lines	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
Two-column table	0	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
HTML table with checkbox inputs	0	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓	✗	✗	✗	✗
JSON-like key/value lines	0	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
Boxes after their option	0	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗
Ballot-box glyphs (☒ ☐)	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗

Forms test suite, continued

Perturbation	Sev.	1. **ParseBench-2 Forms**	2. forms_v2 cap only	3. forms_v2 proportional only	4. Value presence	5. Field F1 (exact)	6. ANLS per field	7. 1 - NED per field	8. Entity-linking F1	9. Checkbox accuracy	10. Page text 1 - NED	11. rule_form_field_pass_rate	12. form_kv_md_core_kv_f1	13. form_bench_text_value_f1	14. form_bench_text_attribution_f1
Check marks and Azure markers	0	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗
Blank fields written N/A	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗
Blank fields written as _____	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
Captions in capitals	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
No colon after captions	0	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗
Printed ids dropped	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓
Values underlined	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
Thousands separators dropped	0	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✓	✓	✓
Dates reformatted	0	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
Sections in another order	0	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✗	✓
Single choice: only the chosen option	0	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓	✓
Choice written as the caption's value	0	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗
Radio buttons as (X) / ( )	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
Round radio glyphs (● ○)	0	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗
Box states as words (Selected / Not selected)	0	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓
Box states as Yes / No values	0	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓
Dates normalised to ISO	0	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
Dates with month names	0	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
Amounts with currency and cents	0	✓	✓	✓	✓	✗	✗	✗	✗	✓	✗	✓	✓	✓	✓
IDs and phone numbers re-punctuated	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
Box-per-character field spaced out	0	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✓	✓
Values in capitals	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Long value wrapped onto the next line	0	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✓	✓
Two fields on one line	0	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗	✓	✓	✓
JSON nested by section	0	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Section headings left out	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗
Blank fields show their format hint	0	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	✓	✗	✗	✗
DEBATABLE															
Captions, then values, in two runs	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Printed hint read into a blank field	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Second half of the page missing	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PCS		99	99	17	12	9	−8	−8	−11	−14	−18	−20	−34	−37	−39

Table 16 Visual grounding test suite: verdict of each of the 4 metrics on each of the 38 scenarios, on one synthetic page. Sev.: severity rating of the level (critical 90, major 50, minor 25, benign 0). ✓ target met, ∼ partly met, ✗ missed.

Perturbation	Sev.	1. **ParseBench-2 Visual grounding**	2. EPR v3 draft	3. EPR v1, no text gate	4. EPR v1
CRITICAL					
One page-wide box	90	✓	✓	✓	✓
Biggest paragraph missed	90	✓	✓	∼	∼
Five paragraphs in one box	90	✓	✓	✓	✓
Right box, wrong text	90	✓	✓	✗	∼
Coarse blocks	90	✓	✓	✓	✓
Loose boxes (IoU about 0.45)	90	✓	✓	✗	✗
Japanese paragraph garbled	90	∼	∼	✗	∼
MAJOR					
Every box six times	50	✓	✓	✗	✓
50 junk boxes	50	✓	✓	✗	✗
Table box cuts the header row	50	✓	✓	✗	✗
Table box cuts the last column	50	✓	✓	✗	✗
Header merged into a paragraph	50	∼	✓	✓	✓
Two columns, one box over both	50	✗	✓	✗	✗
Loose boxes (IoU about 0.64)	50	✓	✓	✗	✗
MINOR					
Five boxes on blank margins	25	✓	✓	✗	✗
Second box over a matched line	25	✓	✓	✗	✗
Empty boxes between table rows	25	✓	✓	✗	✗
Header and footer as page-wide bands	25	✓	✓	✓	✓
Box nicks the next paragraph	25	✓	✓	✗	✗
Text boxes inside a picture	25	✓	✗	✗	✗
Tiny icon missed	25	✓	✓	✓	✓
Hyphen dropped, word left split	25	✓	✓	✗	✓
Japanese paragraph, one character wrong	25	✗	✗	✗	✗
BENIGN					
Perfect	0	✓	✓	✓	✓
Boxes 1.5 pt off	0	✓	✓	✓	✓
Every class wrong	0	✓	✓	✗	✗
Table labelled Picture	0	✓	✓	✗	✗
Two paragraphs in one box	0	✓	✓	✓	✓
Paragraph split into lines	0	✓	✓	✗	✗
Annotations by line, output by paragraph	0	✓	✓	✗	✗
Two columns, one box per part	0	✓	✓	✓	✓
Two columns, one element in two segments	0	✓	✓	✓	✗
Table plus tight cell boxes	0	✓	✓	✓	✓
Boxes on optional sub-images	0	✓	✓	✓	✓
Japanese paragraph, read right	0	✓	✓	✓	✓
Hyphenated word kept as printed	0	✓	✓	✓	✓
Hyphenated word joined	0	✓	✗	✓	✗
DEBATABLE					
Split at the wrong place	–	–	–	–	–
PCS		82	81	2	−6

C Parser Evaluation

C.1 Experimental Setup

We use the same evaluation pipeline and scoring code for all systems.

Configurations. We compute overall scores using the default configuration of each parser tier or mode. We evaluate optional features, such as chart and form parsing, separately and report their dimension scores alongside the defaults. We ask frontier VLMs to transcribe each page using the same prompt. We run open-weight models with their own prompts or pipelines.

Scoring protocol. Providers return outputs in different formats, so we convert them into a common representation before scoring. We score saved outputs against the page annotations using the ParseBench-2 metrics, then combine page scores into dimension and overall scores (Section 3.3). We retry operational failures and flag runs where these failures remain on at least 1% of the pages.

Cost. Cost is the average cost per page over the run, at list prices or usage-based billing as the provider's API reports it. Document parsers use per-page prices, while frontier VLM costs depend on billed token usage. The reported VLM costs therefore depend on the benchmark documents and transcription prompt. We do not report per-page serving costs for self-hosted open-weight models.

Uncertainty. We report 95% confidence intervals using 2,000 bootstrap resamples. We group test cases from the same source document and resample these groups together, using the same draws for every system. This accounts for related pages when comparing scores. The intervals reflect variation across sampled documents; because we run each configuration once, they do not capture variation between repeated runs.

C.2 Where the Cap Decides, per System

Table 17 gives, for every ranked system, the share of pages on which the severity cap decides the page score and the number of charged errors per page (Section 4.2).

Pipeline	Cap-decided pages (%)						Errors per page		
	Text	Fmt	Tab.	Chart	Form	All	Crit.	Maj.	Min.
DOCUMENT PARSERS									
LlamaParse (Agentic Plus)	47	37	21	34	75	42	0.63	1.85	3.84
LlamaParse (Agentic)	54	33	27	59	79	51	1.92	4.25	4.23
LlamaParse (Cost Effective)	56	35	31	56	79	52	2.65	4.77	4.24
Reducto	59	33	39	46	78	51	1.36	2.75	5.28
Extend (Parse 2)	61	32	41	43	76	51	1.50	3.32	4.95
Pulse (Ultra 2)	51	33	33	49	82	50	1.64	3.06	4.83
Extend (Parse Light)	60	26	42	43	77	50	1.58	3.57	5.09
Datalab (Accurate)	52	36	32	42	83	49	1.84	3.66	4.32
Datalab (Balanced)	51	36	34	45	84	50	1.99	3.85	4.44
Datalab (Fast)	56	38	40	48	84	53	2.94	4.50	4.57
Databricks AI Parse	58	27	44	57	83	54	2.83	5.66	5.58
Reducto (r-1)	50	39	35	9	77	41	1.22	1.58	4.75
Mistral OCR 4.1 (annotation)	54	24	40	60	83	53	7.13	5.13	5.01
Reducto (r-1-mini)	54	40	35	0	78	40	1.33	1.61	4.93
LandingAI	53	33	40	17	82	44	1.55	2.30	4.67
Extend (Parse)	60	26	35	1	77	39	1.17	2.04	4.89
Mistral OCR 4.1	54	24	40	0	82	39	2.02	2.61	4.97
Reducto (non-agentic)	63	27	49	0	78	42	2.12	2.27	5.28
Azure Document Intelligence (layout)	61	20	43	0	84	41	1.64	2.77	5.28
DIRECT VISION MODELS									
Claude Opus 5.5	42	30	17	15	78	36	0.75	1.45	4.02
Gemini 4 argon (low)	45	31	23	37	79	43	1.26	2.38	4.12
GPT-5.6 terra (low)	64	36	55	46	83	57	2.28	3.84	4.59
Gemini 3 Flash (minimal)	42	29	33	50	78	46	2.90	3.96	4.50
Gemini 3.7 Flash (low)	47	30	31	10	82	39	1.09	1.74	4.45
Gemini 3.8 Flash (low)	45	30	32	10	82	39	1.02	1.75	4.43
Gemini 3.5 Flash	41	28	28	11	79	37	1.28	1.56	4.40
Gemini 3.6 Flash	42	29	31	9	79	37	1.30	1.52	4.41
Muse Ember 1.0 (low)	48	30	34	29	80	44	1.20	2.59	4.37
GPT-6.1 terra (low)	48	29	22	13	76	37	0.78	1.50	4.29
Claude Sonnet 5.5	47	29	26	7	80	37	0.90	1.29	4.44
GPT-6 luna (low)	57	33	40	17	80	45	1.38	2.09	4.44
Claude Haiku 5.5	47	30	39	2	82	39	1.42	1.52	4.61
GPT-5.6 luna (low)	59	33	53	14	80	47	2.10	2.39	4.72
Gemini 3.1 Flash-Lite	48	23	40	3	80	38	1.54	2.09	5.11
Gemini 3.5 Flash-Lite	56	26	42	3	84	41	1.47	2.93	5.18

Table 17 Cap-decided pages: percentage of scored core pages with an output where $S = c < m(y, r)$; All pools the five dimensions, and Fmt is Formatting on the text pages. Errors per page: mean charged critical, major and minor errors per scored page, pooled over the five core datasets. Visual grounding has no cap.

C.3 Full Results

Tables 18 and 19 are the headline leaderboards of the ranked configurations. Tables 20 and 21 then give the ParseBench-2 scores of every ranked pipeline and of the opt-in feature pipelines that the leaderboards leave out. Table 20 gives the overall score, the six dimension scores, cost and latency; Table 21 gives the page mean of each dataset that enters the overall score. Option pipelines are a base configuration with one opt-in chart or form feature (Appendix C.1). A – marks a dataset that was not run, or a dimension or overall score that cannot be composed because one of its datasets was not run. Fmt core is the Formatting score of the core text pages.

#	Pipeline	ParseBench-2 score	Text	Format.	Tables	Charts	Visual grounding	Forms	$/page
■ DOCUMENT PARSERS						ranked by ParseBench-2 score; best score per column in bold			
1	LlamaParse (Agentic Plus)	**90.1**	90.3	**93.6**	**94.6**	**92.6**	87.5	**81.8** (82.9)	0.028
2	LlamaParse (Agentic)	86.0	87.0	92.7	89.9	77.4 (87.1)	88.3	80.9 (82.7)	0.013
3	LlamaParse (Cost Effective)	84.6	85.5	91.9	87.2	73.1 (87.2)	**88.6**	81.2 (82.5)	0.004
4	Reducto	83.9	81.9	82.8	84.9	89.1	84.1	80.9	0.030
5	Extend (Parse 2)	83.2	80.2	83.3	84.2	85.1	84.8	81.5	0.025
6	Pulse (Ultra 2)	82.7	86.1	88.7	84.7	86.8	73.5	76.4	0.015
7	Extend (Parse Light)	81.6	78.3	78.7	81.4	85.4	84.7	81.2	0.006
8	Datalab (Accurate)	81.3	**91.1**	92.5	90.3	67.8 (61.8)	70.1	75.8	0.010
9	Datalab (Balanced)	81.1	90.3	92.6	89.5	67.6	70.3	76.0	0.004
10	Datalab (Fast)	78.3	87.2	92.0	84.1	61.5	69.4	75.6	0.004
11	Databricks AI Parse	76.9	79.3	79.4	81.3	64.2	83.8	73.5	0.004
12	Reducto (r-1)	76.8	87.8	92.3	89.3	25.2 (73.6*)	87.7	78.2	0.010
13	Mistral OCR 4.1 (annotation)	72.5	84.5	82.0	81.8	29.8	81.8	75.3	0.005
14	Reducto (r-1-mini)	71.6	85.1	91.7	86.1	1.4	87.9	77.7	0.001
15	LandingAI	70.8	86.2	79.9	84.7	38.8	58.4	76.9	0.030
16	Extend (Parse)	69.4	80.0	78.8	88.6	2.6	84.7	81.7	0.025
17	Mistral OCR 4.1	67.8	85.1	82.0	81.8	1.2	81.8	75.2	0.004
18	Reducto (non-agentic)	66.3	79.1	79.8	73.4	1.5	83.8	80.0	0.022
19	Azure Document Intelligence (layout)	66.1	79.8	77.9	77.6	1.5	83.8	76.3 (75.4)	0.010

Table 18 ParseBench-2 headline leaderboard of the document parsers (0–100). LlamaParse (Agentic Plus) leads, and Charts spreads the parsers most. Default configurations. Score: the overall benchmark score, the unweighted mean of the six dimension scores, each a weighted mean of the mean page scores of its datasets (Equation (5)), where one layout test case is one page. Ranks follow the unrounded score, and for 7 of the 18 adjacent pairs the 95% interval of the difference contains zero. Bold: best score in each score column. A score in brackets is the dimension score with the opt-in feature of that dimension turned on, a chart feature under Charts and a form feature under Forms. It can be lower than the default and does not enter the overall score. A Charts score near 0 means that the default configuration writes no chart data. $/page: average cost per page at list prices. –: not evaluated, or no cost per page. *Score includes pages that failed for operational reasons (provider credit, rate limit or time limit) and were scored 0. Notes in Section 4.2.

#	Pipeline	ParseBench-2 score	Text	Format.	Tables	Charts	Visual grounding	Forms	$/page
■	DIRECT VISION MODELS						the raw model called with our prompt		
1	Claude Opus 5.5	85.7	92.7	89.4	94.1	74.1	86.2	77.8	0.065
2	Gemini 4 argon (low)	84.2	89.7	89.2	92.8	68.2	88.1	77.2	0.027
3	GPT-5.6 terra (low)	77.1	81.0	87.9	79.6	76.3	62.5	75.1	0.031
4	Gemini 3 Flash (minimal)	76.6	86.7	86.1	88.9	58.0	66.3	73.8	0.009
5	Gemini 3.7 Flash (low)	75.6	86.7	86.6	89.4	33.9	80.9	75.9	0.008
6	Gemini 3.8 Flash (low)	75.5	87.1	86.7	88.8	33.3	81.3	76.0	0.008
7	Gemini 3.5 Flash	74.7	88.5	86.2	90.0	31.1	75.3	77.4	0.050
8	Gemini 3.6 Flash	74.7	88.2	85.8	89.8	30.4	76.0	77.8	0.042
9	Muse Ember 1.0 (low)	74.5	90.2	86.0	89.0	60.5	43.8	77.5	0.002
10	GPT-6.1 terra (low)	74.3	87.4	88.4	92.2	46.7	51.9	79.0	0.012
11	Claude Sonnet 5.5	73.8	91.4	86.3	90.9	38.6	57.9	77.6	0.031
12	GPT-6 luna (low)	73.5	84.7	89.2	81.5	47.9	61.7	75.8	0.001
13	Claude Haiku 5.5	70.1	89.2	87.4	83.4	12.1	73.0	75.4	0.002
14	GPT-5.6 luna (low)	69.1	80.7	88.6	76.7	40.7	54.0	74.0	0.003
15	Gemini 3.1 Flash-Lite	62.7	84.0	79.3	79.7	12.2	48.1	73.0	0.004
16	Gemini 3.5 Flash-Lite	61.4	79.2	78.7	72.2	5.1	62.2	71.3	0.007
■	OPEN-WEIGHT MODELS						self-hosted, each with its own prompt or pipeline		
1	Infinity-Parser2 Pro	77.1	85.6	78.8	86.2	55.6	82.4	74.0	–
2	LightOnOCR-3 4B	77.0	84.1	81.6	78.2	62.9	80.5	75.0	–
3	Chandra 2	76.2*	77.2	88.8	85.5	62.5	69.6	73.7*	–
4	LightOnOCR-3 0.8B	75.9	81.0	80.9	78.6	61.2	79.2	74.4	–
5	MinerU2.5-Pro	75.6*	76.6	79.0	80.6*	64.0	82.2	71.1	–
6	Infinity-Parser2 Flash	74.4	80.8	78.3	75.4	56.6	81.4	73.8	–
7	Nemotron Parse 3.5	73.5	81.2	82.1	69.5	60.1	76.4	71.4	–
8	PaddleOCR-VL 1.6	70.3	73.5	79.4	73.0	57.5	70.1	68.1	–
9	Qwen3.8 27B	61.6	81.6	77.3	36.7	41.3	63.5	69.0	–

Table 19 ParseBench-2 leaderboard of the direct vision models and the open-weight models (0–100), each group ranked by the unrounded ParseBench-2 score. The best vision model, Claude Opus 5.5 at 85.7, scores below LlamaParse (Agentic Plus) at 90.1 (Table 18). Columns and notes are as in Table 18, without bold. (low) and (minimal) give the reasoning setting of the model. We call the vision models with our transcription prompt and run the open-weight models with their own prompt or pipeline, which limits comparison across the two groups. The open-weight models run self-hosted and have no cost per page. olmOCR 2 and Xiaomi-OCR-0, which return no element boxes, have no overall score and are not listed.

Table 20 Overall benchmark score, dimensions, cost and latency of every pipeline (0–100). Fmt core: the Formatting metric on the core text pages. $/page: average cost per page at list prices. p50, p90: latency per page in seconds on Text v2.0 (508 pages), or on the option's core dataset for option pipelines. *Score includes pages that failed for operational reasons (provider credit, rate limit or time limit) and were scored 0.

Pipeline	ParseBench-2 score	Text	Format.	Tables	Charts	Visual grounding	Forms	Fmt core	$/page	p50	p90
Document parsers, ranked											
LlamaParse (Agentic Plus)	90.1	90.3	93.6	94.6	92.6	87.5	81.8	93.1	0.0281	15.9	51.0
LlamaParse (Agentic)	86.0	87.0	92.7	89.9	77.4	88.3	80.9	92.7	0.0125	6.4	10.5
LlamaParse (Cost Effective)	84.6	85.5	91.9	87.2	73.1	88.6	81.2	92.2	0.0038	6.5	6.8
Reducto	83.9	81.9	82.8	84.9	89.1	84.1	80.9	87.7	0.0300	5.6	9.3
Extend (Parse 2)	83.2	80.2	83.3	84.2	85.1	84.8	81.5	85.2	0.0250	10.6	15.7
Pulse (Ultra 2)	82.7	86.1	88.7	84.7	86.8	73.5	76.4	90.4	0.0150	22.2	34.9
Extend (Parse Light)	81.6	78.3	78.7	81.4	85.4	84.7	81.2	86.1	0.0063	5.5	8.6
Datalab (Accurate)	81.3	91.1	92.5	90.3	67.8	70.1	75.8	92.1	0.0100	6.1	20.1
Datalab (Balanced)	81.1	90.3	92.6	89.5	67.6	70.3	76.0	92.0	0.0040	5.7	15.3
Datalab (Fast)	78.3	87.2	92.0	84.1	61.5	69.4	75.6	91.6	0.0040	5.4	11.0
Databricks AI Parse	76.9	79.3	79.4	81.3	64.2	83.8	73.5	85.4	0.0044	4.3	7.8
Reducto (r-1)	76.8	87.8	92.3	89.3	25.2	87.7	78.2	91.2	0.0100	3.1	5.4
Mistral OCR 4.1 (annotation)	72.5	84.5	82.0	81.8	29.8	81.8	75.3	89.4	0.0050	2.9	5.5
Reducto (r-1-mini)	71.6	85.1	91.7	86.1	1.4	87.9	77.7	90.4	0.0010	2.6	4.2
LandingAI	70.8	86.2	79.9	84.7	38.8	58.4	76.9	87.0	0.0300	11.8	19.0
Extend (Parse)	69.4	80.0	78.8	88.6	2.6	84.7	81.7	86.3	0.0250	6.5	10.6
Mistral OCR 4.1	67.8	85.1	82.0	81.8	1.2	81.8	75.2	89.4	0.0040	1.0	1.7
Reducto (non-agentic)	66.3	79.1	79.8	73.4	1.5	83.8	80.0	87.4	0.0225	2.9	5.3
Azure Document Intelligence (layout)	66.1	79.8	77.9	77.6	1.5	83.8	76.3	85.0	0.0100	5.0	7.9
Direct vision models											
Claude Opus 5.5	85.7	92.7	89.4	94.1	74.1	86.2	77.8	92.5	0.0649	12.4	24.4
Gemini 4 argon (low)	84.2	89.7	89.2	92.8	68.2	88.1	77.2	91.3	0.0268	13.4	25.6
GPT-5.6 terra (low)	77.1	81.0	87.9	79.6	76.3	62.5	75.1	90.3	0.0312	8.8	17.8
Gemini 3 Flash (minimal)	76.6	86.7	86.1	88.9	58.0	66.3	73.8	89.7	0.0089	6.2	11.9
Gemini 3.7 Flash (low)	75.6	86.7	86.6	89.4	33.9	80.9	75.9	89.2	0.0084	9.0	17.0
Gemini 3.8 Flash (low)	75.5	87.1	86.7	88.8	33.3	81.3	76.0	88.8	0.0085	8.9	17.0
Gemini 3.5 Flash	74.7	88.5	86.2	90.0	31.1	75.3	77.4	89.5	0.0497	16.6	33.9
Gemini 3.6 Flash	74.7	88.2	85.8	89.8	30.4	76.0	77.8	89.4	0.0416	13.8	27.0
Muse Ember 1.0 (low)	74.5	90.2	86.0	89.0	60.5	43.8	77.5	91.4	0.0016	5.2	7.6
GPT-6.1 terra (low)	74.3	87.4	88.4	92.2	46.7	51.9	79.0	91.0	0.0115	12.9	23.5
Claude Sonnet 5.5	73.8	91.4	86.3	90.9	38.6	57.9	77.6	91.5	0.0313	8.5	16.2
GPT-6 luna (low)	73.5	84.7	89.2	81.5	47.9	61.7	75.8	90.9	0.0013	10.5	17.3
Claude Haiku 5.5	70.1	89.2	87.4	83.4	12.1	73.0	75.4	91.1	0.0015	5.2	9.9
GPT-5.6 luna (low)	69.1	80.7	88.6	76.7	40.7	54.0	74.0	90.3	0.0030	10.0	17.3
Gemini 3.1 Flash-Lite	62.7	84.0	79.3	79.7	12.2	48.1	73.0	86.8	0.0039	3.7	6.4
Gemini 3.5 Flash-Lite	61.4	79.2	78.7	72.2	5.1	62.2	71.3	85.2	0.0072	3.4	6.7
Open-weight models											
Infinity-Parser2 Pro	77.1	85.6	78.8	86.2	55.6	82.4	74.0	86.5	–	51.9	111.0
LightOnOCR-3 4B	77.0	84.1	81.6	78.2	62.9	80.5	75.0	89.7	–	32.3	74.5
Chandra 2	76.2*	77.2	88.8	85.5	62.5	69.6	73.7*	91.2	–	10.3	22.9
LightOnOCR-3 0.8B	75.9	81.0	80.9	78.6	61.2	79.2	74.4	88.9	–	19.4	42.3
MinerU2.5-Pro	75.6*	76.6	79.0	80.6*	64.0	82.2	71.1	86.8	–	4.6	10.2
Infinity-Parser2 Flash	74.4	80.8	78.3	75.4	56.6	81.4	73.8	85.7	–	17.2	36.6
Nemotron Parse 3.5	73.5	81.2	82.1	69.5	60.1	76.4	71.4	89.9	–	4.3	14.3
PaddleOCR-VL 1.6	70.3	73.5	79.4	73.0	57.5	70.1	68.1	86.2	–	7.5	34.5
Qwen3.8 27B	61.6	81.6	77.3	36.7	41.3	63.5	69.0	84.8	–	25.8	54.7
Option pipelines (base configuration plus one option)											
LlamaParse (Agentic) + specialized charts	–	–	–	–	87.1	–	–	–	0.0125	21.5	31.1
LlamaParse (Cost Effective) + specialized charts	–	–	–	–	87.2	–	–	–	0.0037	20.6	35.8
Reducto (r-1) + chart agent	–	–	–	–	73.6*	–	–	–	0.0638	238.0	3848.8
Datalab (Accurate) + chart understanding	–	–	–	–	61.8	–	–	–	0.0130	13.7	21.9
LlamaParse (Agentic Plus) + enriched forms	–	–	–	–	–	–	82.9	–	0.0281	35.8	66.0

Pipeline	ParseBench-2 score	Text	Format.	Tables	Charts	Visual grounding	Forms	Fmt core	$/page	p50	p90
LlamaParse (Agentic) + enriched forms	–	–	–	–	–	–	82.7	–	0.0125	25.7	46.2
LlamaParse (Cost Effective) + enriched forms	–	–	–	–	–	–	82.5	–	0.0037	25.7	46.8
Azure Document Intelligence (layout) + key–value pairs	–	–	–	–	–	–	75.4	–	0.0100	5.7	12.7

Table 21 Page mean (0–100) of each dataset of the overall score for every pipeline. Core: Text v2.0, Tables v2.0, Charts v2.0, Layout v2.0, Forms v2.0. Slots: complex table headers v2.0, SERFF rate tables v2.0, financial tables v2.0, dense tables v2.0, headings v2.0, strikethrough v2.0, underline v2.0, blank public forms v2.0. n: datasets run out of 13. Strikethrough and underline share one slot. *Score includes pages that failed for operational reasons (provider credit, rate limit or time limit) and were scored 0.

Pipeline	Text	Tab.	Chart	Lay.	Form	Hdr.	SERFF	Fin.	Dense	Head.	Strike	Under.	Blank	n
Document parsers, ranked														
LlamaParse (Agentic Plus)	90.3	96.1	92.6	87.5	79.8	90.3	93.5	93.7	94.8	97.1	94.5	92.4	95.7	13
LlamaParse (Agentic)	87.0	94.9	77.4	88.3	78.9	75.2	85.3	92.2	86.5	97.2	82.7	94.1	95.5	13
LlamaParse (Cost Effective)	85.5	91.2	73.1	88.6	79.2	73.3	84.1	90.8	84.5	97.6	78.1	91.7	95.0	13
Reducto	81.9	88.6	89.1	84.1	79.0	70.2	86.4	88.5	80.2	95.3	32.6	49.5	94.0	13
Extend (Parse 2)	80.2	87.3	85.1	84.8	79.6	74.3	84.8	88.8	76.5	95.8	85.5	34.5	94.8	13
Pulse (Ultra 2)	86.1	90.7	86.8	73.5	74.3	65.7	85.3	92.4	71.5	96.4	67.9	74.5	91.8	13
Extend (Parse Light)	78.3	85.5	85.4	84.7	79.2	70.3	82.9	86.1	69.9	96.0	13.9	20.0	95.2	13
Datalab (Accurate)	91.1	93.3	67.8	70.1	73.2	82.9	86.6	93.2	86.2	96.6	89.4	93.0	93.9	13
Datalab (Balanced)	90.3	92.7	67.6	70.3	73.5	82.2	86.0	92.7	84.8	96.6	91.0	92.8	93.9	13
Datalab (Fast)	87.2	86.9	61.5	69.4	73.0	79.1	84.2	91.3	70.5	96.5	87.6	91.3	93.8	13
Databricks AI Parse	79.3	84.2	64.2	83.8	70.9	66.1	81.2	90.0	76.4	94.1	22.4	34.2	91.6	13
Reducto (r-1)	87.8	92.9	25.2	87.7	76.1	75.3	88.8	92.3	86.8	95.9	95.5	94.6	93.0	13
Mistral OCR 4.1 (annotation)	84.5	86.4	29.8	81.8	72.9	77.2	79.0	88.4	64.2	96.4	25.4	20.0	92.2	13
Reducto (r-1-mini)	85.1	90.2	1.4	87.9	75.6	68.6	85.6	90.8	82.9	95.6	94.1	96.8	92.5	13
LandingAI	86.2	89.3	38.8	58.4	74.8	69.3	80.7	90.6	80.0	90.9	32.1	21.4	92.2	13
Extend (Parse)	80.0	91.4	2.6	84.7	79.8	77.2	87.0	92.9	85.7	96.0	13.9	20.0	95.1	13
Mistral OCR 4.1	85.1	86.3	1.2	81.8	72.8	77.1	79.4	88.4	64.0	96.5	25.3	20.0	92.2	13
Reducto (non-agentic)	79.1	80.5	1.5	83.8	78.1	38.1	78.6	81.9	67.1	95.3	15.7	22.8	93.7	13
Azure Document Intelligence (layout)	79.8	86.2	1.5	83.8	73.7	34.5	77.7	86.9	76.7	96.1	13.9	20.0	94.0	13
Direct vision models														
Claude Opus 5.5	92.7	96.2	74.1	86.2	75.7	84.5	93.9	96.3	93.4	96.0	93.7	36.0	93.2	13
Gemini 4 argon (low)	89.7	94.3	68.2	88.1	74.9	87.4	90.9	94.2	92.7	95.3	89.2	52.0	93.3	13
GPT-5.6 terra (low)	81.0	83.7	76.3	62.5	72.7	57.6	83.1	89.1	71.8	95.4	70.9	61.1	92.1	13
Gemini 3 Flash (minimal)	86.7	91.8	58.0	66.3	72.1	78.3	87.7	92.2	86.1	95.8	67.1	43.3	85.1	13
Gemini 3.7 Flash (low)	86.7	92.6	33.9	80.9	73.5	75.2	89.4	92.2	88.2	96.1	74.1	49.2	93.1	13
Gemini 3.8 Flash (low)	87.1	91.9	33.3	81.3	73.5	75.6	88.2	91.6	87.4	95.9	72.9	55.9	93.1	13
Gemini 3.5 Flash	88.5	92.2	31.1	75.3	75.2	80.9	89.9	93.3	86.7	95.8	71.0	42.2	92.5	13
Gemini 3.6 Flash	88.2	92.2	30.4	76.0	75.7	81.8	89.1	92.7	86.3	95.9	69.4	39.7	93.1	13
Muse Ember 1.0 (low)	90.2	92.6	60.5	43.8	75.2	74.2	87.4	92.7	87.1	95.9	43.0	43.5	93.3	13
GPT-6.1 terra (low)	87.4	96.5	46.7	51.9	77.7	72.0	93.8	95.4	90.8	96.3	87.1	42.7	88.7	13
Claude Sonnet 5.5	91.4	93.6	38.6	57.9	75.5	82.4	89.6	94.0	86.7	95.8	61.0	30.6	92.2	13
GPT-6 luna (low)	84.7	87.1	47.9	61.7	73.6	63.6	80.4	90.4	69.5	95.2	83.4	62.2	91.4	13
Claude Haiku 5.5	89.2	88.0	12.1	73.0	73.2	67.0	85.4	89.1	73.8	95.3	69.6	45.5	90.7	13
GPT-5.6 luna (low)	80.7	83.2	40.7	54.0	71.8	52.4	79.6	87.8	61.3	96.1	77.4	64.2	89.7	13
Gemini 3.1 Flash-Lite	84.0	84.2	12.2	48.1	70.7	61.2	81.2	84.5	74.0	95.3	16.5	20.0	89.1	13
Gemini 3.5 Flash-Lite	79.2	78.3	5.1	62.2	68.8	58.1	69.5	68.8	67.9	93.1	27.3	22.2	88.8	13
Open-weight models														
Infinity-Parser2 Pro	85.6	88.9	55.6	82.4	71.4	74.1	86.7	89.7	83.4	94.4	13.0	20.7	91.8	13
LightOnOCR-3 4B	84.1	83.9	62.9	80.5	72.8	65.5	81.4	80.7	62.2	96.7	14.4	20.8	90.2	13
Chandra 2	77.2	89.1	62.5	69.6	72.2	72.4	85.4	91.7	78.0	95.9	57.1	78.2	83.7*	13
LightOnOCR-3 0.8B	81.0	84.3	61.2	79.2	72.3	61.4	80.5	82.4	67.0	96.7	13.8	20.8	89.4	13
MinerU2.5-Pro	76.6	86.0	64.0	82.2	68.3	52.0*	82.6	90.0	76.1	94.6	13.6	20.0	90.7	13
Infinity-Parser2 Flash	80.8	79.7	56.6	81.4	71.1	57.7	74.4	82.6	70.2	94.9	13.9	20.6	92.3	13
Nemotron Parse 3.5	81.2	75.8	60.1	76.4	69.3	55.3	67.4	77.2	52.8	95.8	16.9	26.7	86.4	13
PaddleOCR-VL 1.6	73.5	76.2	57.5	70.1	65.7	69.7	71.7	80.0	58.3	93.3	13.8	36.1	85.5	13
Qwen3.8 27B	81.6	38.6	41.3	63.5	66.7	38.1	35.7	37.1	28.3	93.2	13.9	18.4	85.2	13
Option pipelines (base configuration plus one option)														
LlamaParse (Agentic) + specialized charts	–	–	87.1	–	–	–	–	–	–	–	–	–	–	1
LlamaParse (Cost Effective) + specialized charts	–	–	87.2	–	–	–	–	–	–	–	–	–	–	1
Reducto (r-1) + chart agent	–	–	73.6*	–	–	–	–	–	–	–	–	–	–	1
Datalab (Accurate) + chart understanding	–	–	61.8	–	–	–	–	–	–	–	–	–	–	1
LlamaParse (Agentic Plus) + enriched forms	–	–	–	–	81.2	–	–	–	–	–	–	–	95.0	2

Pipeline	Text	Tab.	Chart	Lay.	Form	Hdr.	SERFF	Fin.	Dense	Head.	Strike	Under.	Blank	n
LlamaParse (Agentic) + enriched forms	–	–	–	–	81.0	–	–	–	–	–	–	–	94.8	2
LlamaParse (Cost Effective) + enriched forms	–	–	–	–	80.7	–	–	–	–	–	–	–	94.7	2
Azure Document Intelligence (layout) + key–value pairs	–	–	–	–	72.9	–	–	–	–	–	–	–	93.0	2

C.3.1 Opt-in chart and form features

The overall score uses each parser's default configuration, but several parsers offer an opt-in feature for charts or forms. Table 22 compares each base configuration with the same configuration and one feature turned on, on the dimension the feature targets.

Chart features matter most, in both directions. A default configuration that writes no chart data can gain the most: with its chart agent, Reducto r-1 rises from 25.2 to 73.6* on Charts. LlamaParse's specialized chart parsing raises Charts from 77.4 to 87.1 on the Agentic tier and from 73.1 to 87.2 on Cost Effective. A feature can also hurt: Datalab Accurate's chart understanding lowers Charts from 67.8 to 61.8.

The gain has a price, in money or in time. Reducto's chart agent raises the cost from $0.010 to $0.064 per page and the median latency from 3.0 s to 238.0 s per chart page. LlamaParse's chart parsing keeps the same price per page and takes about three to six times longer.

Form features change little. The ranked parsers already sit within a narrow band on Forms, 73.5 to 81.8. LlamaParse's enriched forms add one to three points (80.9 to 82.7 on Agentic), and Azure's key–value pairs lower Forms from 76.3 to 75.4.

These are not overall scores. Counted in the overall score, specialized charts would move LlamaParse Cost Effective from 84.6 to 87.0 and the chart agent would move Reducto r-1 from 76.8 to 84.8*. We ran each feature only on its target datasets, so these figures are estimates: turning a feature on everywhere could change the other dimensions and the cost.

			Dimension				$/page		ParseBench-2 score		
Base configuration	Feature	Dim.	Base	Feature	Δ	Core forms	Base	Feat.	Base	w/ feat.	p50 (s)
LlamaParse (Agentic)	specialized charts	Charts	77.4	87.1	+9.7	–	0.013	0.013	86.0	87.7	6.5 → 21.5
LlamaParse (Cost Effective)	specialized charts	Charts	73.1	87.2	+14.2	–	0.004	0.004	84.6	87.0	3.6 → 20.6
Reducto (r-1)	chart agent	Charts	25.2	73.6*	+48.4*	–	0.010	0.064	76.8	84.8*	3.0 → 238.0
Datalab (Accurate)	chart understanding	Charts	67.8	61.8	−6.1	–	0.010	0.013	81.3	80.3	7.1 → 13.7
LlamaParse (Agentic Plus)	enriched forms	Forms	81.8	82.9	+1.1	79.8 → 81.2	0.028	0.028	90.1	90.2	25.7 → 35.8
LlamaParse (Agentic)	enriched forms	Forms	80.9	82.7	+1.8	78.9 → 81.0	0.013	0.013	86.0	86.3	6.5 → 25.7
LlamaParse (Cost Effective)	enriched forms	Forms	81.2	82.5	+1.2	79.2 → 80.7	0.004	0.004	84.6	84.8	6.5 → 25.7
Azure Document Intelligence (layout)	key–value pairs	Forms	76.3	75.4	−0.8	73.7 → 72.9	0.010	0.010	66.1	66.0	4.6 → 5.7

Table 22 Each base configuration against the same configuration with one opt-in feature. Dimension: the Charts or Forms score without and with the feature, and the difference. Core forms: the core form pages alone. ParseBench-2 score with the feature: an estimate that replaces only the target dimension. p50: median latency per page in seconds, without and with the feature. *Score includes pages that failed for operational reasons (provider credit, rate limit or time limit) and were scored 0.