Reading Between the Lines

A Benchmark for Editorial Correction Detection in Printed Text

Amin Ahmad
PAIR Systems, Inc.
amin@pairsys.ai

ABSTRACT

We introduce the Little Dorrit Editor Benchmark, a small but challenging multimodal evaluation task that tests whether language models can interpret handwritten editorial corrections in printed literary text. Based on annotated scans of six pages from Charles Dickens's *Little Dorrit*, the benchmark asks models to detect edits—such as insertions, deletions, and punctuation changes—and output them in a structured JSON format. Our evaluation framework includes a partial-credit scoring function and uses nonparametric bootstrapping to estimate confidence intervals around key metrics. Initial results show that even strong models achieve modest F_1 scores, with wide confidence intervals pointing to both model variability and dataset sparsity. While limited in size, the benchmark offers a focused, realistic testbed for document understanding and multimodal reasoning. We hope future work will expand the dataset and build on these early results.

1 INTRODUCTION

More than 150 years after her quiet rebellion in the Marshalsea, *Little Dorrit* returns—with red pen in hand—to evaluate the judgment of modern language models. We introduce the **Little Dorrit Editor Benchmark**, a multimodal evaluation task designed to test whether foundation models can understand handwritten editorial corrections in printed literary text.

The benchmark is constructed from annotated scans of Charles Dickens' *Little Dorrit*, featuring realistic editorial marks made by human annotators. These include insertions, deletions, replacements, punctuation changes, capitalization edits, and text reordering—each mimicking the conventions used by professional editors. Unlike standard OCR or layout analysis tasks, our benchmark requires fine-grained interpretation: the model must infer the editor's intent by integrating both visual and textual cues within the context of natural language.

Each sample in the benchmark comprises a single scanned page with markup overlaid on printed text. The task is to detect all editorial changes and output structured JSON entries describing each edit: its type, location (line number and page), and the original and corrected text. This setting demands not only visual understanding but linguistic precision and familiarity with editorial norms.

We evaluate a range of state-of-the-art models, including GPT-4o, GPT-4.5 Preview, and smaller variants such as GPT-4o Mini. Results show a wide performance spread, with even the strongest model achieving modest F1 scores—highlighting the benchmark's difficulty and the gap between human editorial fluency and current model capabilities.

The Little Dorrit Editor Benchmark aims to inspire progress in document understanding, multimodal reasoning, and human-in-the-loop AI systems that collaborate in the revision and refinement of text.

2 TASK FORMALIZATION AND EVALUATION FRAMEWORK

2.1 DATASET AND GROUND TRUTH ANNOTATIONS

Let $P = \{p_1, \ldots, p_N\}$ be a set of scanned pages, where each p_i is a single page image from a literary source. Associated with each page is a set of human-annotated editorial corrections, denoted

$E_i^{\text{gt}} = \{e_{i1}, \ldots, e_{im_i}\}$. Each edit e_{ij} represents a structured correction localized to a specific region of the page.

2.2 Edit Types and Structure

Each annotation is a structured edit denoted as a tuple:

$$e = (\text{type}, \text{original\_text}, \text{corrected\_text}, \text{line\_number}, \text{page\_id})$$

The task requires identifying all such edits present in a scanned document page and outputting them individually. Each edit belongs to one of six possible types:

- **insertion**: New text added to the original.
- **deletion**: Existing text removed from the original.
- **replacement**: Substitution of one segment of text with another.
- **punctuation**: Changes involving punctuation marks (e.g., addition, deletion, or substitution).
- **capitalization**: Case corrections (e.g., "hello" to "Hello").
- **reordering**: Re-sequencing of words or phrases (e.g., "fast ran" to "ran fast").

Each edit must be recorded as a separate entry, even if multiple edits occur within the same phrase or sentence. This granular modeling is critical for precise benchmarking.

The fields `original_text` and `corrected_text` should be as minimal as possible—only the directly affected portion of text is included. Line numbers begin at 1 for the main body of text; headings and titles are assigned line number 0.

2.3 Model Predictions

Let $f_\theta \colon P \to \mathcal{P}(E)$ be a multimodal language model that maps each page p_i to a predicted set of edits $\hat{E}_i = f_\theta(p_i)$. The function f_θ is stochastic: multiple invocations on the same input may yield different outputs due to the non-deterministic nature of decoding when using modern inference APIs.

To account for this, we define K samples per page:

$$\hat{E}_i^{(1)}, \hat{E}_i^{(2)}, \ldots, \hat{E}_i^{(K)} \sim f_\theta(p_i)$$

We evaluate each model using 2-shot inference, in which the prompt includes two worked examples of structured editorial corrections. The same two examples are used for all pages and all models.

2.4 Evaluation Metrics

Given a predicted edit set $\hat{E}_i$ and ground truth E_i^{gt}, we define the per-page precision, recall, and F_1 score as:

$$\text{Precision}_i = \frac{|\hat{E}_i \cap E_i^{\text{gt}}|}{|\hat{E}_i|}, \quad \text{Recall}_i = \frac{|\hat{E}_i \cap E_i^{\text{gt}}|}{|E_i^{\text{gt}}|}, \quad F_{1,i} = \frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

To compute dataset-level scores, we aggregate across all pages:

$$\text{TP} = \left| \bigcup_{i=1}^{N} \left(\hat{E}_i \cap E_i^{\text{gt}} \right) \right|, \quad \text{FP} = \left| \bigcup_{i=1}^{N} \hat{E}_i \right| - \text{TP}, \quad \text{FN} = \left| \bigcup_{i=1}^{N} E_i^{\text{gt}} \right| - \text{TP}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.5 BOOTSTRAPPED CONFIDENCE INTERVALS

To estimate the uncertainty of dataset-level evaluation metrics, we employ a nonparametric bootstrapping procedure using cached model outputs. This approach provides an efficient and statistically principled approximation of the sampling distribution of performance metrics, without requiring additional model inference.

For each page p_i, we independently generate M samples of predicted edits from the model, denoted $\{\hat{E}_i^{(1)}, \hat{E}_i^{(2)}, \ldots, \hat{E}_i^{(M)}\}$. These represent distinct outputs sampled from the model's inherent stochasticity.

Then, for each bootstrap replicate $b = 1, \ldots, B$, we:

1. Sample N pages with replacement from the evaluation set.
2. For each sampled page, randomly select one of its M cached model outputs.
3. Compute dataset-level metrics (e.g., precision, recall, and F_1) using the adjudication and scoring procedure described in Section 3.

This produces a set of metric values $\{\text{Metric}^{(1)}, \ldots, \text{Metric}^{(B)}\}$ for each metric of interest. We report 95% confidence intervals using the 2.5th and 97.5th percentiles of the resulting distributions.

Discussion. This procedure captures two primary sources of variability:

- *Dataset-level variability:* Pages are sampled with replacement, so each bootstrap replicate reflects how performance may vary with different test set compositions.
- *Model stochasticity:* For each sampled page, one of the M cached model outputs is randomly selected, simulating variation in predictions due to sampling noise, decoding temperature, or other nondeterminism in model generation.

Although this bootstrap does not fully explore the space of possible outputs $f_\theta(p_i)$—since it is limited to M samples per page—it offers a practical and conservative estimate of metric uncertainty. The resulting confidence intervals may slightly underestimate the true variance, especially when M is small or the model exhibits high variability. Nevertheless, this approach remains a sound and efficient approximation under realistic computational constraints.

3 SCORING FUNCTION AND MATCHING CRITERIA

To determine whether a predicted edit is correct, we define a two-stage evaluation process involving (1) a heuristic matcher for identifying candidate alignments between predicted and ground truth edits, and (2) an LLM-based judgment to confirm correctness.

3.1 EDIT MATCHING HEURISTICS

Each page p_i contains a set of ground truth edits E_i^{gt} and a set of predicted edits $\hat{E}_i$. We perform approximate matching using the following rules:

- **Edit type match**: The predicted edit must have the same type (e.g., `punctuation`, `replacement`) as the ground truth edit.
- **Textual overlap**: The predicted and gold edits must share significant overlap in either `original_text` or `corrected_text`.
- **Line number proximity**: The predicted and ground truth edits must occur within ± 3 lines of one another.

If these criteria are satisfied, the predicted edit is considered a candidate match and passed to an LLM for final judgment.

3.2 LLM ADJUDICATION

We use GPT-4.5 as a judging model to evaluate whether a predicted edit accurately captures the intent of a ground truth edit. The prompt (provided in Appendix **??**) presents both edits and asks the LLM to assess:

1. Whether the edit types are identical.
2. Whether the predicted edit captures the essential change, even if surrounding context differs.

The model returns a JSON response with `"is_correct": true/false` and justification.

3.3 LINE NUMBER PENALTY AND PARTIAL CREDIT

While line numbers are ignored during LLM adjudication, we impose a penalty post hoc to account for location mismatches. Given the line difference $\Delta\ell = |\ell_{\text{pred}} - \ell_{\text{gt}}|$, a quadratic penalty is applied:

$$\text{penalty} = \min(1.0,\ 0.1 \cdot \Delta\ell^2)$$

The final edit score is defined as:

$$\text{score} = \begin{cases} 1 - \text{penalty} & \text{if LLM deems the edit correct} \\ 0 & \text{otherwise} \end{cases}$$

An edit is considered a *true positive* only if its final score is ≥ 0.5. This effectively caps the tolerated line mismatch at 2 lines. Edits matched by the heuristic but falling below this threshold are treated as false positives.

3.4 DATASET-LEVEL METRICS

Precision, recall, and F_1 are computed using adjudicated scores assigned to each matched edit. Let S denote the total score across all adjudicated matches, where each score lies in $[0, 1]$ and may be reduced (e.g., due to line number penalties). Then:

$$\text{Precision} = \frac{S}{TP + FP}, \quad \text{Recall} = \frac{S}{TP + FN}, \quad F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

This formulation preserves theoretical consistency by treating S as a weighted "correctness mass" and comparing it to the total prediction or ground truth mass. It generalizes classical precision and recall while accommodating partial credit.

These metrics are computed both globally and separately by edit type.

4 RESULTS

Table 1 summarizes the performance of various models on the Little Dorrit Editor Benchmark. For each model, we report the F_1 score along with its 95% confidence interval (expressed as the upper and lower margins), as well as the precision and recall. Note that even the top-performing model achieves a modest F_1 score, and the wide confidence intervals suggest considerable variability in performance.

5 CONCLUSION

We presented the Little Dorrit Editor Benchmark, a challenging multimodal task requiring models to interpret handwritten editorial corrections in literary text. The leaderboard results show that even advanced large language models struggle to achieve high F_1 scores, highlighting the complexity and nuance of editorial understanding. Moreover, the relatively wide confidence intervals suggest that further data and more refined evaluation methods could improve reliability and reduce variance. Despite these challenges, the benchmark demonstrates promise as a realistic, high-value test of both document-level vision and linguistic reasoning. We hope this work will spur progress in human-in-the-loop AI tools, driving future innovations in editorial assistance and multimodal document processing.

Table 1: Leaderboard Results on the Little Dorrit Editor Benchmark.

Rank	Model	F_1	95% CI	Precision	Recall
1	GPT-4o (2024-11-20) (2-shot)	0.3215	+0.316 / -0.215	0.4146	0.2625
2	ChatGPT-4o (2-shot)	0.2297	+0.150 / -0.132	0.2972	0.1871
3	Mistral Small 3.1 24B (2-shot)	0.1497	+0.164 / -0.097	0.1513	0.1481
4	Llama 4 Maverick (2-shot)	0.1059	+0.090 / -0.106	0.1314	0.0887
5	GPT-4 Turbo (2-shot)	0.0903	+0.177 / -0.090	0.1443	0.0657
6	Gemma 3 27B (2-shot)	0.0846	+0.154 / -0.085	0.0907	0.0793
7	GPT-4o-mini (2-shot)	0.0498	+0.118 / -0.050	0.0562	0.0447
8	GPT-4.5 Preview (2-shot)	0.0488	+0.088 / -0.049	0.0483	0.0492
9	Llama 4 Scout (2-shot)	0.0123	+0.061 / -0.012	0.0139	0.0111
10	Grok 2 Vision 1212 (2-shot)	0.0034	+0.010 / -0.003	0.0022	0.0083
11	DeepSeek R1 (2-shot)	0.0000	+0.000 / -0.000	0.0000	0.0000
12	DeepSeek V3 0324 (2-shot)	0.0000	+0.000 / -0.000	0.0000	0.0000
13	Gemini 2.5 Pro Preview 3/25 (2-shot)	0.0000	+0.000 / -0.000	0.0000	0.0000
14	Google: Gemini 2.0 Flash (2-shot)	0.0000	+0.000 / -0.000	0.0000	0.0000
15	Llama 3.2 90B Vision Instruct (2-shot)	0.0000	+0.000 / -0.000	0.0000	0.0000
16	Phi 4 Multimodal Instruct (2-shot)	0.0000	+0.000 / -0.000	0.0000	0.0000

6 Resources

For reproducibility and further exploration, we make our code and evaluation framework publicly available. The following resources are provided:

- GitHub Repository: little-dorrit-editor — Contains the complete codebase for dataset management, evaluation, and bootstrapping.
- Leaderboard: Little Dorrit Editor Benchmark — A live leaderboard showcasing model performance on the benchmark.

References

A Prompt Template for Model Instructions

This appendix contains the prompt used in our system to elicit structured editorial corrections from language models. It defines the task, output format, edit taxonomy, and markup conventions.

A.1 Instruction Prompt

You are an expert editor tasked with identifying handwritten editorial corrections on printed text pages.

Your job is to identify all handwritten markups and corrections on the page and convert them to structured JSON output.

EXTREMELY IMPORTANT: Create a *SEPARATE* edit entry for *EACH* individual correction, even if multiple corrections occur in the same sentence or phrase. **DO NOT** combine multiple edits into a single edit entry.

A.2 Correction Fields

For each individual correction, identify:

1. The type of edit (insertion, deletion, replacement, punctuation, capitalization, reordering)
2. The original text being modified (minimal scope)

3. The corrected text after applying the edit
4. The line number where the edit occurs (line 0 is title; body text begins at line 1)
5. The page identifier

A.3 Output Format

Structured JSON format:

```
{
  "edits": [
    {
      "type": "insertion | deletion | replacement \
              | punctuation | capitalization | reordering",
      "original_text": "the text before the edit",
      "corrected_text": "the text after the edit",
      "line_number": <integer>,
      "page": "page_identifier"
    },
    ...
  ]
}
```

A.4 Edit Types

Each edit must be categorized into one of the following types:

- **insertion**: Adding new text
- **deletion**: Removing text
- **replacement**: Substituting text with alternatives
- **punctuation**: Modifying or adding punctuation marks
- **capitalization**: Changing case (upper/lower)
- **reordering**: Rearranging text sequence

A.5 Handwritten Markup Conventions

The following visual cues are commonly used in the annotated documents:

- Caret marks (ˆ) for insertions
- Strikethroughs for deletions
- Circled or underlined text for replacements
- Added or modified punctuation
- Capitalization notations
- Arrows or numbering for reordering

A.6 Examples of Separated Edits

Example 1:
Instead of one edit:
`"My dog ran fast and barked"` → `"My dog ran fast, and barked."`

Create two edits:

- `"fast and"` → `"fast, and"` (punctuation)
- `"barked"` → `"barked."` (punctuation)

Example 2:
Instead of one edit:
`"hello world"` → `"Hello, world!"`

Create three edits:

- `"hello"` → `"Hello"` (capitalization)
- `"world"` → `"world,"` (punctuation)
- `"world,"` → `"world!"` (punctuation)

Be precise about line numbers. Count full body lines starting from 1. Titles and headings are line 0.