OPERATION-GUIDED PROGRESSIVE HUMAN-TO-AI TEXT TRANSFORMATION BENCHMARK FOR MULTI-GRANULARITY AI-TEXT DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

As AI writing assistants become increasingly integrated into real-world drafting and revision workflows, many documents are no longer purely human-written or AI-generated, but instead result from progressive human–AI co-editing. However, existing AI-text detection benchmarks largely focus on final outputs and provide limited understanding of how AI authorship signals emerge, accumulate, or disappear throughout the revision process. We introduce **OpAI-Bench**, an operation-guided benchmark for studying progressive human-to-AI text transformation across document, sentence, token, and span granularities. Starting from human-written documents, OpAI-Bench constructs nine sequentially revised versions for each sample under predefined AI coverage levels and five representative AI edit operations, covering four domains while preserving complete authorship provenance at multiple granularities. The benchmark supports comprehensive evaluation with 8 document-level detectors, 7 sentence-level detectors, and 2 fine-grained token/span-level detectors. Experiments reveal that AI-text detectability is governed not only by the proportion of AI-edited content, but also by edit operation, domain, and cumulative revision history. Interestingly, we notice that mixed-authorship intermediate versions are often harder to detect than both fully human and heavily AI-edited endpoints, exposing non-monotonic detection patterns missed by existing benchmarks. OpAI-Bench provides a controlled testbed for analyzing *whether*, *when*, and *how* AI-assisted writing becomes detectable under controlled progressive human–AI editing scenarios.

1 INTRODUCTION

AI-assisted writing and editing workflows (Saha & Feizi, 2025; Thai et al., 2025; Uchendu et al., 2021; Guo et al., 2023; He et al., 2024; Macko et al., 2023) are increasingly embedded in practical writing workflows, where they are used not only to generate complete passages, but also to revise, polish, expand, compress, and restructure human-written drafts. As a result, many documents are better characterized as products of progressive human–AI co-editing rather than as purely human-written or fully AI-generated text. This emerging writing paradigm challenges the conventional binary framing of AI-text detection, where a document is assumed to belong to one of two endpoint categories. In practice, AI involvement may appear locally, accumulate gradually, and interact with prior human content across multiple rounds of revision.

Existing AI-text detection benchmarks Dugan et al. (2024); Wang et al. (2024b;a) provide valuable resources for evaluating whether a completed text is human- or AI-written, but they offer limited support for analyzing how detectability evolves during the transformation from human writing to AI-edited text. Most benchmarks are constructed from static final outputs and do not preserve the intermediate revision states that lead to those outputs. Consequently, they cannot answer several important questions: at what stage does AI involvement become reliably detectable, which types of edits introduce the strongest detection signals, and whether mixed-authorship texts are easier or harder to detect than endpoint cases.

Evaluation should capture not only how much text is AI-edited, but also how it is edited and where the edits occur. Different operations, such as polishing, paraphrasing, expansion, and compres-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

sion, may leave different detection signals. Collapsing these operations into a single AI-written label obscures their different effects on detection. Moreover, document-level labels support global screening but cannot localize AI involvement, while sentence-level labels may miss mixed human–AI fragments within a sentence. Token- and span-level provenance is therefore needed to evaluate fine-grained localization.

To address these limitations, we introduce **OpAI-Bench**, an operation-guided benchmark for progressive human-to-AI text transformation and multi-granularity AI-text detection. Starting from human-written source documents, OpAI-Bench constructs cumulative revision trajectories under predefined AI coverage levels and representative edit operations. Each trajectory records how human text is transformed by AI edits over time while preserving authorship provenance at the document, sentence, token, and span levels. This design enables controlled evaluation of detection performance as a function of AI coverage, edit operation, domain, and cumulative revision history.

Using OpAI-Bench, we conduct a comprehensive evaluation of document-level, sentence-level, and fine-grained AI-text detectors. Our results show that AI-text detectability is not determined solely by the proportion of AI-edited content, but varies substantially across edit operations, domains, and revision stages. In particular, intermediate mixed-authorship versions can be more challenging than both purely human and heavily AI-edited endpoints, revealing non-monotonic detection behavior that is overlooked by existing static benchmarks. These findings motivate moving beyond binary endpoint classification toward trajectory-aware and operation-aware evaluation.

Our contributions are summarized as follows:

- We introduce **OpAI-Bench**, an operation-guided benchmark for progressive human-to-AI text transformation that preserves intermediate revision states rather than only final outputs.
- We construct cumulative revision trajectories with predefined AI coverage levels and five edit operations, and provide provenance across document, sentence, token, and span levels.
- We benchmark diverse detector families across granularities and use OpAI-Bench to assess detector stability across coverage, edit operations, domains, generators, and revision history. Our results reveal non-monotonic detectability, with a critical mixed-authorship region around v_4 where intermediate AI coverage and compression coincide.

2 Related Work

AI-text detection benchmarks. Prior AI-text detection benchmarks cover increasingly diverse human-written, machine-generated, and AI-edited text. TuringBench (Uchendu et al., 2021), HC3 (Guo et al., 2023), MGTBench (He et al., 2024), MULTITuDE (Macko et al., 2023), RAID (Dugan et al., 2024), M4 (Wang et al., 2024b), and M4GT-Bench (Wang et al., 2024a) broaden evaluation across generators, domains, languages, and robustness conditions. Other benchmarks extend evaluation to prompt attacks, paraphrasing, perturbations, data mixing, generator attribution, and mixed human–machine change-point detection (Wu et al., 2024; Wang et al., 2024a). However, they mainly evaluate final generated or edited texts, or a single authorship transition, rather than a full revision process with explicit intermediate stages.

Mixed-authorship and fine-grained attribution. Another line of work studies hybrid and coauthored text. MixSet (Zhang et al., 2024a) shows the difficulty of detecting mixed human–AI writing, including AI-revised human drafts and human-revised machine outputs. Localization methods such as AdaLoc (Zhang et al., 2024b), SenDetEX (Jiang et al., 2025), HACo-Det (Su et al., 2025), and DAMASHA (Teja et al., 2026) move toward sentence-, word-, or token-level attribution, while Beemo (Artemova et al., 2025) and RealBench (He et al., 2025) show that human post-editing and diverse collaboration patterns further complicate detection. These works motivate mixed-authorship and fine-grained evaluation, but typically analyze completed hybrid samples rather than the revision trajectories that produce them.

Varying degrees of AI involvement. Recent work has begun to move beyond binary AI-text detection by considering text with varying degrees of AI involvement. APT-Eval (Saha & Feizi, 2025) varies the degree of AI polishing and shows that lightly edited text can be difficult to distinguish from fully human writing. PaLD (Lei et al., 2025) estimates the proportion and localization of LLM-written content, while EditLens (Thai et al., 2025) models AI involvement as a continuous

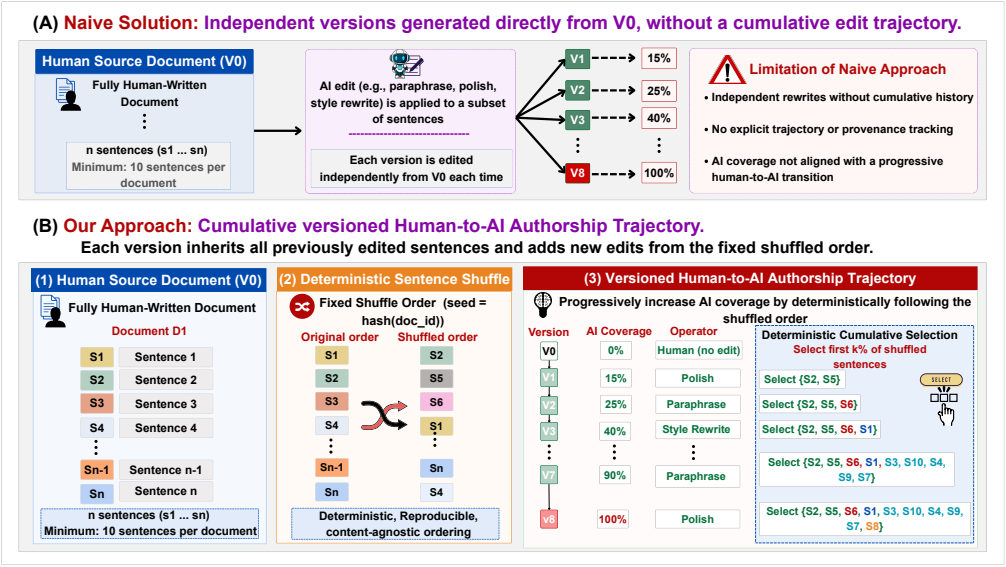

Figure 1: OpAI-Bench construction pipeline. Top: independent version generation without revision history. Bottom: cumulative trajectory construction using a growing set of sentence indices from a deterministic shuffled selection order. The shuffle affects only selection; released versions preserve the original sentence order. Internal <AI_Start> and </AI_End> markers are used only during construction and removed from released text.

edit-magnitude signal relative to an original human draft. HACo-Det (Su et al., 2025) similarly motivates numeric notions of AI ratio through fine-grained attribution. Together, these works move beyond binary labels, but they typically focus on final edited samples or single editing outcomes. In contrast, OpAI-Bench studies explicit cumulative revision trajectories with preserved intermediate versions, controlled AI coverage, and diverse edit operations.

3 OpAI-Bench

3.1 Overview and Design Principles

Most prior AI-text detection benchmarks evaluate a completed text and assign it a final authorship label, such as human, AI-generated, or mixed. This formulation has been valuable for static detection, but it is less informative for studying how detectability evolves as AI involvement is introduced through revision. Prior benchmarks generally do not preserve explicit intermediate versions, do not model cumulative version-to-version editing, and do not jointly control AI coverage, edit type, and multi-granularity provenance within a unified setup. Consequently, they provide only limited support for analyzing when detection becomes reliable, whether matched AI coverage yields different difficulty across edit types, and whether detector behavior depends on revision history as well as on the final edited text. OpAI-Bench is designed to study this setting by representing mixed authorship as a controlled revision trajectory. Given a source document D, we construct $\mathcal{T}(D) = \left(D^{(0)}, D^{(1)}, \ldots, D^{(8)}\right)$, where $D^{(0)}$ is the original human-written document and $D^{(V)}$ denotes version $V \in \{0, \ldots, 8\}$. Rather than generating each version independently from the original source, we construct each $D^{(V)}$ by editing the previous version $D^{(V-1)}$. This yields a cumulative sequence in which the target AI coverage increases monotonically from fully human (v0) to fully AI-edited (v8) under predefined coverage ratios and edit operations.

Each version is paired with provenance annotations at the token, sentence, and document levels. This structure enables analyses that are difficult to carry out on static benchmarks, including coverage-controlled comparisons, edit-type-controlled comparisons, and trajectory-controlled comparisons. In particular, it allows us to study whether detectability depends only on the final amount of AI-edited content, or also on the path through which mixed authorship is formed. Table 1 compares OpAI-Bench with prior resources, and Figure 1 illustrates the construction pipeline.

Benchmark	Authorship		Trajectory		Diversity			Annot.
	Mixed	AI_Cov	Interm.	Cumul.	Edit	Dom.	LLM	Gran.
DetectRL Wu et al. (2024)	✓	✗	✗	✗	✓	✓	✓	D
MixSet (Zhang et al., 2024a)	✓	✗	✗	✗	✓	✓	✓	M
RAID (Dugan et al., 2024)	✗	✗	✗	✗	✓	✓	✓	D
M4GT-Bench (Wang et al., 2024a)	✓	✗	✗	✗	✗	✓	✓	M
RealBench (He et al., 2025)	✓	✗	✗	✗	✓	✓	✓	D
Beemo (Artemova et al., 2025)	✓	✗	✗	✗	✓	✓	✓	D
HACo-Det (Su et al., 2025)	✓	✗	✗	✓	✗	✓	✓	M
SenDetEX (Jiang et al., 2025)	✓	✗	✗	✗	✗	✓	✓	S
APT-Eval (Saha & Feizi, 2025)	✓	✓	✗	✗	✗	✓	✓	D
DAMASHA (Teja et al., 2026)	✓	✗	✗	✗	✗	✓	✓	B
EditLens (Thai et al., 2025)	✓	✗	✗	✗	✓	✓	✓	D
OpAI-Bench (Ours)	✓	✓	✓	✓	✓	✓	✓	M

Table 1: Comparison with prior AI-involved text detection benchmarks. Mixed = mixed human–AI authorship; AI_Cov = controlled AI coverage; Interm = benchmark-wide intermediate stages; Cumul = benchmark-wide cumulative trajectories; Edit = diverse edit operations; Gran. = annotation/evaluation granularity (D: document, S: sentence, B: boundary, M: multi-granularity).

OpAI-Bench is guided by five design principles. First, **versioned trajectories** expand each source document into **nine** ordered versions (`v0`–`v8`): one human source, seven intermediate mixed-authorship checkpoints, and one controlled fully AI-edited reference endpoint. This balances coverage resolution with construction cost, while treating `v8` as a reference endpoint rather than an assumption that real workflows end with all sentences AI-edited. Second, **controlled AI coverage** introduces edits at predefined sentence-level ratios from 0% to 100%, with each intermediate version serving as a valid mixed-authorship stopping point. Third, **edit-type diversity** covers **five** common revision intents through *polish*, *paraphrase*, *style rewrite*, *compress*, and *expand*: light correction, reformulation, register change, condensation, and addition of detail. Repeated operations appear at lower and higher coverage levels, while compression and expansion are placed in the intermediate mixed-authorship region as contrasting deletion/addition operations, enabling the benchmark to probe whether detector behavior changes when substantial human and AI content coexist. Fourth, **cumulative construction** generates each version from the previous one, preserving accumulated revision history. Finally, **multi-granularity provenance** preserves AI-revision labels at token/span, sentence, and document levels.

3.2 Benchmark Construction

Source Domains, Representation, and Filtering. OpAI-Bench is constructed from human-written documents across four domains: student essays Learning Agency Lab (2024), news articles Narayan et al. (2018), government reports Huang et al. (2021), and scientific abstracts Paul (2021). These domains vary in writing style, discourse structure, and sentence length, enabling evaluation across diverse text types. We use *document* as the unit of editing. Depending on the source corpus, a document may be a multi-paragraph text or a single paragraph treated as a document. Each document is normalized and segmented into paragraphs and sentences, represented as $D = (p_1, \ldots, p_m)$, where each paragraph $p_i = (s_1^{(i)}, \ldots, s_{n_i}^{(i)})$. Stable paragraph and sentence identifiers are assigned and preserved across all derived versions for sentence selection, reconstruction, and authorship tracking. To support intermediate revision trajectories, we retain only documents with at least 10 sentences, i.e., $N(D) = \sum_{i=1}^{m} n_i \geq 10$. Each retained document is then expanded into a nine-version trajectory while preserving its paragraph and sentence structure. Summary statistics are reported in Table 2. Train, development, and test splits are constructed at the original source-document level: all versions and generator-specific trajectories derived from the same human-written `v0` source are assigned to the same split, preventing row-level leakage. Detailed split statistics by domain and generator are reported in Appendix C.

Versioned Trajectories and Cumulative Editing. For each source document D, OpAI-Bench constructs an ordered trajectory $\mathcal{T}(D) = (D^{(0)}, D^{(1)}, \ldots, D^{(8)})$, where $D^{(0)}$ is the original human-written document and later versions follow the fixed operation–coverage schedule in Table 4. The trajectory spans the transition from fully human text at `v0` to fully AI-edited text at `v8`, with intermediate versions increasing the amount of edited content while varying the edit operation.

Domain	#Docs	#Traj.	#Samples	Avg. sent.	Avg. tok.
Student essays	3,969	7,906	71,154	21.0	398.8
News articles	3,998	7,892	71,028	24.0	491.3
Government reports	3,993	8,000	72,000	20.6	563.7
Scientific abstracts	3,762	7,290	65,610	11.0	234.3
Total	15,722	31,088	279,792	19.3	426.1

Table 2: Statistics of the OpAI-Bench main split. Docs are human-written `v0` sources; trajectories are generator-specific revision sequences; samples count all released versions `v0`–`v8`.

To select edited sentences, we define a fixed document-specific ordering $\pi(D) = (\pi_1, \ldots, \pi_{N(D)})$ using a deterministic shuffle seeded by the document identifier. This ordering is computed once and reused across all versions, ensuring reproducibility and removing positional bias. Given target sentence-level coverage $c_{\text{sent}}^{(t)}$ at version t, we select $k^{(t)} = \lceil c_{\text{sent}}^{(t)} \cdot N(D) \rceil$ sentences and define the edited set as $S^{(t)} = \{\pi_1, \ldots, \pi_{k^{(t)}}\}$. Because coverage increases monotonically, the edited sets satisfy $S^{(0)} \subseteq S^{(1)} \subseteq \cdots \subseteq S^{(8)}$. This yields a cumulative editing process: once selected, a sentence remains editable in later versions, so subsequent versions both add newly selected sentences and re-edit previously modified ones under the current operation. This preserves not only where AI edits occur, but also the revision history through which they accumulate.

Editing Operations and Coverage Control. OpAI-Bench uses five sentence-level rewrite operations to model different forms of AI-assisted revision: *polish*, *paraphrase*, *style rewrite*, *compress*, and *expand*. These operations vary the form of intervention: *polish* makes fluency edits, *paraphrase* changes wording while preserving meaning, *style rewrite* modifies tone, *compress* removes non-essential phrasing, and *expand* adds limited clarifying detail. All prompts require the model to rewrite only the target sentence, preserve names, numbers, entities, and factual content, avoid unsupported additions, and return a single sentence. We further enforce sentence-level consistency through automatic validation and manual verification: outputs are discarded if they split the target into multiple sentences, merge across sentence boundaries, or violate the expected format. Full prompts are provided in Appendix G.

For each version $D^{(t)}$, the target sentence-level AI coverage $c_{\text{sent}}^{(t)}$ determines the fraction of sentences selected for rewriting according to Table 4. Token- and span-level annotations are projected from AI-edited sentence regions using whitespace tokenization: words whose character offsets overlap an AI-marked region are labeled AI, and consecutive AI-labeled words are merged into spans. These labels are tokenizer-agnostic and can be aligned to any subword vocabulary via character offsets. This convention is consistent with segment-level mixed-authorship settings that label AI-revised regions as machine-involved (Zhang et al., 2024a).

4 Experiments

4.1 Tasks

OpAI-Bench evaluates AI-text detection at three granularities over the same versioned trajectory $\mathcal{T}(D) = (D^{(0)}, \ldots, D^{(8)})$. **Document-level detection** predicts whether a document version contains any AI-edited content. The source version $D^{(0)}$ is labeled human, while later versions are labeled AI-involved. **Sentence-level attribution** predicts a binary authorship label for each sentence, marking whether the sentence has entered the cumulative AI-edited set by the corresponding version. **Token- and span-level localization** evaluates fine-grained provenance by identifying the AI-edited portions of a document at the benchmark token or character-span level. All tasks are evaluated at each version $t \in \{0, \ldots, 8\}$ rather than only at the final edited document. This allows us to measure detector behavior throughout the human-to-AI revision trajectory.

4.2 Evaluation Protocol

Evaluation regimes. We organize detectors into three categories corresponding to distinct levels of task-specific adaptation. **(i) Zero-shot methods**: detectors are evaluated in their original settings, using either released checkpoints or models reproduced from the original training scripts, with no exposure to OpAI-Bench data. This measures how well existing detection methods transfer to trajectory-style mixed authorship without adaptation. Document-level methods in-

Detector	Ver.	Essays		Reports		News		Abstracts	
		Acc	F1-AI	Acc	F1-AI	Acc	F1-AI	Acc	F1-AI
Sentence Level									
Claude-Haiku	$v0$	91.2	0.0	96.4	0.0	91.9	0.0	53.2	0.0
	$v1$	77.2 -13.9	22.0 +22.0	79.9 -16.5	7.0 +7.0	78.0 -13.9	11.2 +11.2	50.6 -2.6	25.9 +25.9
	$v2$	74.7 -2.5	45.5 +23.5	69.4 -10.5	11.4 +4.4	70.5 -7.5	20.3 +9.1	48.9 -1.6	37.6 +11.7
	$v3$	75.0 +0.3	65.5 +20.0	56.2 -13.2	17.1 +5.6	61.9 -8.6	31.3 +11.0	51.4 +2.5	51.2 +13.7
	$v4$	62.9 -12.1	52.7 -12.9	47.8 -8.4	13.4 -3.6	51.2 -10.7	23.0 -8.3	49.8 -1.6	52.2 +1.0
	$v5$	69.8 +7.0	69.7 +17.1	48.7 +0.9	34.2 +20.8	52.9 +1.7	40.3 +17.3	60.0 +10.2	67.4 +15.2
	$v6$	75.2 +5.4	81.5 +11.7	41.1 -7.6	37.8 +3.6	46.2 -6.7	45.2 +4.9	66.8 +6.9	77.5 +10.1
	$v7$	66.0 -9.2	77.5 -4.0	34.4 -6.7	42.3 +4.5	33.4 -12.7	41.7 -3.5	69.1 +2.3	80.5 +3.0
	$v8$	64.3 -1.7	78.0 +0.5	30.7 -3.6	43.0 +0.7	27.3 -6.1	40.4 -1.3	71.7 +2.5	83.1 +2.6
AdaLoc (Zhang et al., 2024b)	$v0$	85.1	0.0	94.7	0.0	86.6	0.0	96.7	0.0
	$v1$	72.3 -12.8	20.5 +20.5	79.0 -15.7	7.9 +7.9	73.9 -12.7	15.8 +15.8	79.6 -17.1	6.1 +6.1
	$v2$	66.2 -6.1	23.7 +3.3	70.5 -8.5	7.9 -0.0	67.4 -6.5	18.6 +2.8	70.3 -9.3	5.7 -0.5
	$v3$	57.1 -9.1	30.1 +6.3	56.9 -13.6	8.8 +0.9	56.3 -11.1	20.8 +2.2	57.2 -13.1	5.2 -0.4
	$v4$	50.9 -6.2	30.9 +0.8	48.1 -8.7	6.5 -2.3	49.4 -6.9	21.1 +0.3	48.4 -8.8	5.1 -0.1
	$v5$	43.0 -8.0	27.5 -3.4	38.6 -9.5	7.2 +0.7	42.0 -7.4	22.5 +1.4	38.3 -10.2	4.9 -0.2
	$v6$	34.7 -8.3	30.9 +3.4	25.4 -13.3	7.4 +0.3	30.7 -11.3	23.0 +0.4	22.3 -16.0	5.1 +0.2
	$v7$	28.6 -6.1	37.8 +6.9	11.6 -13.8	8.2 +0.8	21.2 -9.5	26.3 +3.4	12.1 -10.2	7.1 +2.0
	$v8$	29.8 +1.2	44.7 +6.9	5.0 -6.6	9.5 +1.3	16.9 -4.3	28.8 +2.4	5.6 -6.5	10.6 +3.5
Document Level									
RADAR (Hu et al., 2023)	$v0$	71.2	0.0	99.9	0.0	99.9	0.0	99.3	0.0
	$v1$	33.7 -37.5	50.4 +50.4	0.1 -99.9	0.1 +0.1	0.1 -99.9	0.1 +0.1	0.9 -98.4	1.7 +1.7
	$v2$	36.2 +2.5	53.1 +2.8	0.2 +0.2	0.4 +0.3	0.3 +0.3	0.7 +0.6	1.0 +0.2	2.1 +0.4
	$v3$	45.2 +9.0	62.2 +9.0	0.3 +0.1	0.5 +0.1	0.6 +0.2	1.2 +0.5	1.3 +0.2	2.6 +0.5
	$v4$	42.8 -2.4	59.9 -2.3	0.2 -0.1	0.3 -0.2	0.2 -0.4	0.3 -0.8	0.7 -0.6	1.5 -1.1
	$v5$	49.4 +6.7	66.1 +6.3	0.3 +0.1	0.5 +0.2	0.8 +0.6	1.5 +1.2	1.4 +0.7	2.8 +1.3
	$v6$	53.1 +3.7	68.9 +2.7	0.3 +0.1	0.7 +0.1	1.0 +0.2	2.0 +0.5	1.2 -0.2	2.3 -0.4
	$v7$	52.5 -0.6	68.7 -0.2	0.3 -0.1	0.5 -0.1	1.4 +0.4	2.7 +0.7	0.9 -0.3	1.8 -0.5
	$v8$	55.8 +3.3	71.5 +2.7	0.4 +0.2	0.9 +0.3	2.0 +0.6	3.8 +1.2	0.2 -0.7	0.5 -1.3
Fast-DetectGPT (Bao et al., 2023)	$v0$	83.7	0.0	58.3	0.0	52.3	0.0	82.5	0.0
	$v1$	19.7 -64.0	32.8 +32.8	44.2 -14.1	61.3 +61.3	48.5 -3.8	65.0 +65.0	15.9 -66.6	27.3 +27.3
	$v2$	11.1 -8.6	19.9 -12.9	28.4 -15.8	43.9 -17.3	32.4 -16.1	48.0 -17.0	7.8 -8.0	14.4 -13.0
	$v3$	10.1 -0.9	18.4 -1.6	32.5 +4.1	48.4 +4.5	29.9 -2.5	44.2 -3.8	9.8 +2.0	17.6 +3.2
	$v4$	7.9 -2.3	14.5 -3.8	19.1 -13.4	31.8 -16.6	22.2 -7.7	35.1 -9.0	5.8 -4.1	10.9 -6.8
	$v5$	8.5 +0.7	15.7 +1.1	20.1 +1.0	32.6 +0.8	22.4 +0.2	34.9 -0.2	5.8 +0.0	10.8 -0.0
	$v6$	10.1 +1.6	18.3 +2.6	19.2 -1.0	31.0 -1.5	20.8 -1.6	32.9 -2.0	6.1 +0.3	11.3 +0.4
	$v7$	7.0 -3.1	12.9 -5.4	10.5 -8.7	18.5 -12.6	18.2 -2.6	29.4 -3.5	4.4 -1.7	8.3 -2.9
	$v8$	12.1 +5.1	20.8 +7.9	18.9 +8.4	29.1 +10.7	23.9 +5.6	35.2 +5.8	12.2 +7.8	20.4 +12.1
Token and Span Level									
DAMASHA (Teja et al., 2026)	$v0$	89.5	0.0	79.4	0.0	98.6	0.0	83.1	0.0
	$v1$	75.8 -13.7	19.2 +19.2	68.5 -10.9	19.4 +19.4	82.0 -16.6	2.4 +2.4	70.2 -12.9	20.6 +20.6
	$v2$	68.7 -7.1	25.0 +5.8	62.5 -6.0	22.5 +3.1	72.4 -9.6	4.8 +2.3	63.6 -6.5	26.1 +5.5
	$v3$	59.0 -9.7	38.3 +13.3	52.9 -9.6	31.7 +9.2	57.5 -14.9	9.4 +4.6	57.2 -6.5	35.3 +9.2
	$v4$	56.2 -2.9	30.9 -7.4	51.4 -1.5	26.3 -5.5	53.0 -4.5	5.2 -4.2	53.8 -3.3	31.9 -3.4
	$v5$	50.4 -5.8	48.7 +17.8	45.0 -6.4	39.2 +13.0	39.0 -14.0	13.3 +8.2	46.1 -7.7	41.2 +9.3
	$v6$	51.2 +0.8	59.8 +11.1	41.6 -3.5	46.8 +7.6	27.9 -11.1	17.2 +3.9	41.9 -4.2	47.9 +6.6
	$v7$	42.3 -8.9	55.9 -3.9	32.8 -8.7	43.5 -3.3	14.1 -13.9	12.8 -4.3	35.1 -6.7	46.6 -1.3
	$v8$	43.2 +0.9	59.9 +4.0	32.5 -0.3	48.2 +4.7	7.6 -6.4	13.5 +0.7	34.9 -0.3	51.5 +4.9
GigaCheck (Tolstykh et al., 2024)	$v0$	98.7 –	0.0 –	98.5 –	0.0 –	99.5 –	0.0 –	97.5 –	0.0 –
	$v1$	82.6 -16.1	7.6 +7.6	81.8 -16.8	2.6 +2.6	82.9 -16.6	1.5 +1.5	79.6 -17.9	6.8 +6.8
	$v2$	74.2 -8.4	19.0 +11.4	71.9 -9.9	3.3 +0.6	72.6 -10.4	3.0 +1.5	69.4 -10.1	12.1 +5.3
	$v3$	66.1 -8.1	39.0 +20.0	56.6 -15.3	5.7 +2.5	57.2 -15.3	8.2 +5.2	58.5 -11.0	22.4 +10.3
	$v4$	58.9 -7.2	26.6 -12.4	52.6 -4.0	8.7 +3.0	51.9 -5.3	6.8 -1.4	53.3 -5.2	21.4 -1.0
	$v5$	54.6 -4.3	44.8 +18.3	38.6 -14.1	11.5 +2.7	38.7 -13.3	12.1 +5.3	45.9 -7.4	29.9 +8.5
	$v6$	56.7 +2.1	62.3 +17.4	28.9 -9.7	18.3 +6.8	30.1 -8.6	20.7 +8.7	43.9 -2.0	46.5 +16.6
	$v7$	41.6 -15.1	53.8 -8.4	17.0 -11.9	17.7 -0.6	19.0 -11.1	20.0 -0.7	32.8 -11.1	40.5 -5.9
	$v8$	51.1 +9.4	66.7 +12.8	11.2 -5.8	18.1 +0.4	15.7 -3.3	23.9 +3.8	30.9 -1.9	45.2 +4.7

Table 3: Aggregate main results across revision versions and domains, averaged over generators. We report accuracy and F1-AI for zero-shot detectors and LLM-as-detectors at their native evaluation granularity: document, sentence, or fine-grained token/span level. Colored deltas show the change from the previous version.

clude Desklib (Desklib, 2024), DetectLLM (Su et al., 2023), E5-Small (Wang et al., 2022), Fast-DetectGPT (Bao et al., 2023), OOD-LLM Detect (Zeng et al., 2025), RADAR (Hu et al., 2023), RoBERTa OpenAI (Solaiman et al., 2019), and GigaCheck (Tolstykh et al., 2024); sentence-level methods include AdaLoc (Zhang et al., 2024b), GenAI Sentence (Teja et al., 2025), GL-CLiC (Adi et al., 2025), and SeqXGPT (Wang et al., 2023); token- and span-level methods include DAMASHA (Teja et al., 2026) and GigaCheck (Tolstykh et al., 2024).

(ii) LLM-as-detector: three frontier language models: GPT-5.4, Gemini 3 Flash, and Claude Haiku 4.5 are queried as zero-shot classifiers via sentence-level confidence prompting, without any gradient-based adaptation; they serve as prompt-based reference points.

(iii) Trained on OpAI-Bench: detectors are fine-tuned on the OpAI-Bench training split using sentence-level provenance labels drawn from three in-distribution generators (GPT-5.4, GPT-5.4-nano, Gemini 2.5 Flash). Qwen3-8B is withheld from training and evaluated as a held-out generator,

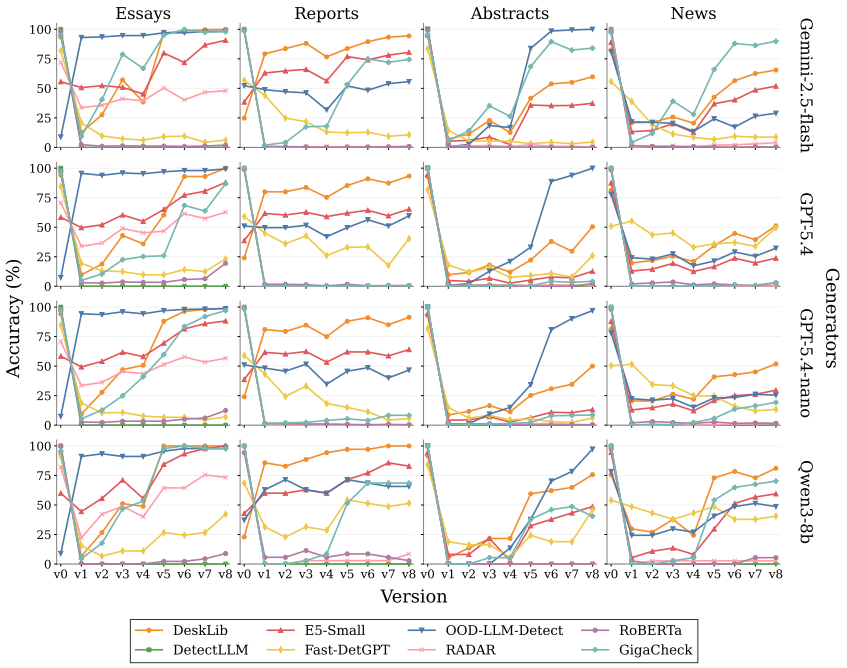

Figure 2: Document-level accuracy across revision versions, domains, and generators. Each curve represents a document-level detector, illustrating how performance changes along the progressive human-to-AI revision trajectory.

isolating cross-generator transfer while holding the training protocol fixed. Document-level methods include Fast-DetectGPT and GigaCheck; sentence-level methods include AdaLoc, GenAI-Sentence, GL-CLiC, and SeqXGPT; token-level and span-level methods include DAMASHA and GigaCheck. Full implementation details and training hyperparameters are provided in the appendix.

Domain and generator splits. We report results by domain and generator to evaluate robustness across writing settings and generation backends. For fine-tuned methods, Qwen3-8B is used only as a held-out generator for cross-generator evaluation.

Metrics. For each task and version, we report accuracy and AI-class F_1 (F_1^{AI}) as complementary measures of overall performance and AI-class detection. Because class prevalence varies across the trajectory, we additionally report Macro-F1 and false-negative rate (FNR) for sentence- and fine-grained token/span-level evaluation in Tables 11 and 12. Document-level AUROC for zero-shot detectors is reported in Appendix D.3. Figures 2 and 3 visualize accuracy by domain and generator, while detailed F_1^{AI} breakdowns are provided in Appendix C.1.

5 Main Results

Detection performance does not increase smoothly as AI coverage grows, but instead varies across revision stages, domains, and detector types. Table 3 summarizes aggregate results across granularities, averaged over generators. Figures 2 and 3 show document- and sentence-level accuracy by domain and generator, with full breakdowns in Appendices C.1 and D.

Document-level detectors show domain-specific failure patterns. While aggregate results in Table 3 suggest overall trends, Figure 2 shows that zero-shot document-level detectors exhibit substantial variation across domains and revision stages, rather than following a consistent trend with increasing AI coverage. RADAR illustrates a domain- and threshold-dependent failure mode. Under its native threshold, it obtains meaningful F1-AI mainly on essays, increasing from 50.4 at `v1` to 71.5 at `v8`, while remaining near zero on reports, news, and abstracts. However, Appendix D.3 shows that RADAR retains AUROC on news comparable to essays, suggesting that the low F1-AI on news may partly reflect sensitivity to the native decision threshold rather than a complete absence of ranking signal. In contrast, RADAR is near chance on reports and exhibits below-chance AUROC on abstracts, indicating that its ranking behavior differs substantially across domains. Fast-DetectGPT

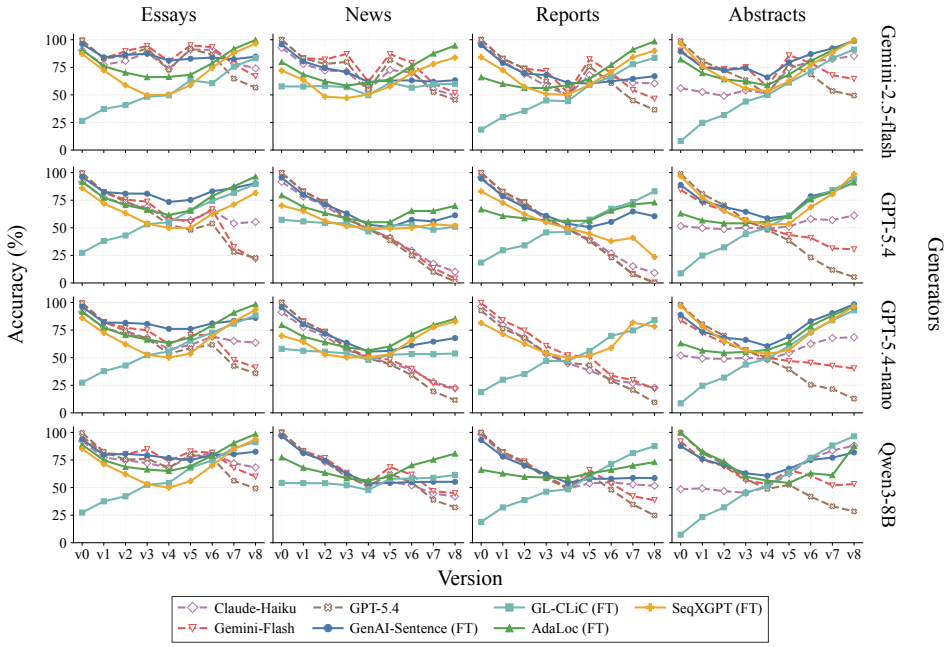

Figure 3: Sentence-level accuracy across revision versions, broken down by domain and generator. The figure compares LLM-as-detectors with fine-tuned sentence-level detectors (FT), showing how detector behavior changes across the progressive human-to-AI revision trajectory.

shows a different pattern: it detects early edits in reports and news, reaching 61.3 and 65.0 F1-AI at v1, but declines as revision progresses, falling to 29.1 and 35.2 at v8. These results suggest that document-level detector behavior is sensitive to domain, decision threshold, and revision stage, so increasing AI coverage alone does not guarantee easier detection.

Sentence-level supervision improves stability, but does not remove revision effects. Sentence-level results show a clear gap between detector families. LLM-as-detectors often improve through early revisions, drop near the compression step, and only partially recover afterward. For example, Gemini-Flash on essays rises from 9.5 F1-AI at v_1 to 71.5 at v_3, drops to 53.6 at v_4, and reaches 57.9 at v_8. Zero-shot sentence-level detectors are less consistent: AdaLoc and GL-CLiC recover in some settings, while SeqXGPT remains near-zero in F1-AI across versions despite high early accuracy. Fine-tuned sentence-level models are generally more stable, with GenAI-Sentence and AdaLoc often recovering at higher AI coverage. However, Figure 3 shows that performance still varies across domains and generators, indicating that sentence-level supervision improves robustness but does not eliminate sensitivity to revision stage or edit operation.

Fine-grained localization follows a different trade-off. DAMASHA operates at the token level and GigaCheck at the span level, so we treat them as localization methods rather than directly comparable document classifiers. Unlike document- and sentence-level detectors, fine-grained methods generally show decreasing accuracy but increasing F1-AI as AI-revision-labeled regions become denser. This reflects class imbalance in early versions: predicting mostly human yields high accuracy when few word-level tokens or spans are labeled AI, while F1-AI becomes more informative at higher AI coverage. Nevertheless, localization remains domain- and generator-dependent, with news consistently harder than essays or abstracts.

Overall, the main trajectory shows that AI-text detection depends on more than the amount of AI-edited content. The ablations in Section 5.2 further analyze this by varying coverage, operation, and cumulative construction separately, suggesting that the degradation around v4 is associated with intermediate coverage, compression, and accumulated revision history rather than coverage alone.

5.1 Edit Distance and Semantic Similarity

To characterize edit magnitude, we measure stepwise and cumulative edit distances and embedding-based semantic similarity for AI-labeled sentences; full statistics are reported in Appendix D.4.

Ver.	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
Op.	none	polish	para.	style	comp.	exp.	style	para.	polish
Cov.	0%	15%	25%	40%	50%	60%	75%	90%	100%

Table 4: OpAI-Bench version schedule. Op. denotes edit operation; Cov. denotes target sentence-level AI coverage.

The results show that OpAI-Bench contains both light polishing and substantial reformulation while maintaining high semantic similarity. They also show that stepwise edit magnitude alone does not explain detectability: the final polishing stage has small stepwise change but substantial cumulative distance from the original source, reflecting the effect of accumulated revision history.

5.2 Controlled Analysis of Coverage, Operation, and Revision History

The main trajectory combines three factors by design: AI coverage, edit operation, and cumulative revision history. Thus, version-level results characterize detector behavior along the scheduled cumulative trajectory. We therefore run controlled analyses to clarify the non-monotonic patterns in the main results, especially around `v4`, where intermediate human–AI coverage coincides with compression. We report the primary ablation results with *Gemini-2.5-Flash* and repeat the ablations with *GPT-5.4-nano* in Appendix E.4 to assess whether the findings are robust to the choice of editing model. This second-generator analysis supports the same qualitative conclusion: AI coverage alone does not fully explain detectability, although the relative difficulty of specific operations varies across domains and detector families.

Varying coverage within each operation. We first fix the edit operation and vary target AI coverage. Figures 12 and 13 show that sentence-level performance does not uniformly collapse at 50% coverage. Instead, trends differ by operation, with compression generally harder to detect than paraphrase or expansion at comparable coverage. Document-level results show a similar but more detector-dependent pattern (Figures 14 and 15). Thus, the degradation around v_4 is not explained by the human–AI ratio alone.

Comparing operations at fixed coverage. We next fix one coverage level at a time (25%, 50%, or 75%) and compare *compress*, *paraphrase*, and *expand*. Figures 16 and 17 show that, for LLM-as-detectors, detectability generally increases from compression to expansion, especially at higher coverage levels. Some zero-shot detectors remain weak or nearly flat across operations. Document-level results show a similar but less uniform trend; full results are provided in Appendix E.2. These results show that edit operation affects detectability even when AI coverage is fixed.

Cumulative versus independent editing. Finally, we compare the main cumulative trajectory with an independent-editing setting, where each version is edited directly from the original human source. This preserves the same version schedule while removing accumulated re-editing. As shown in Appendix E.3, both sentence- and document-level results remain non-monotonic, including a visible drop around the compression step. The independent trajectories are generally smoother and the drop is less pronounced, indicating that cumulative revision can amplify, but does not fully account for, the observed instability.

6 Conclusion

We introduced **OpAI-Bench**, an operation-guided benchmark for evaluating AI-text detection under progressive human–AI revision. Unlike endpoint-based benchmarks, OpAI-Bench preserves intermediate revision states and multi-granularity provenance, enabling detection to be studied as a trajectory rather than a single binary decision. Our results show that AI-text detectability is not governed by AI coverage alone. Across document, sentence, token, and span granularities, detector behavior depends strongly on edit operation, domain, generator, and detector family. In particular, mixed-authorship intermediate versions can be harder to detect than both human-written and heavily AI-edited endpoints, revealing non-monotonic failure modes that static evaluations miss. Controlled ablations further show that compression is often harder to detect than expansion at matched coverage. These findings suggest that reliable AI-text detection requires moving beyond endpoint human-versus-AI classification toward trajectory-aware and operation-aware evaluation. OpAI-Bench provides a controlled testbed for this setting and highlights the need for detectors that can localize and reason about partial, incremental, and operation-specific AI involvement.

AI USE STATEMENT

We used generative AI tools to help with language polishing, grammar correction, and clarity improvements during paper writing. We also used large language models as revision generators to construct the model-edited versions in OpAI-Bench, as described in the benchmark construction section. We did not use generative AI tools for retrieval and discovery, proving mathematical claims, or making final research decisions. All AI-assisted writing and generated benchmark artifacts were reviewed, validated, and edited by the authors.

ETHICS STATEMENT

This work follows the ICLR Code of Ethics. OpAI-Bench is constructed from cited source datasets and model-generated revisions, and does not involve human-subject experiments or the collection of private user data. The benchmark is intended to support reproducible research on AI-text detection under progressive editing scenarios. Because AI-text detection can affect authorship attribution and may be used in high-stakes settings, OpAI-Bench should be used as an evaluation resource rather than as a definitive authority for individual authorship judgment. We release the benchmark and code to support transparent, reproducible, and responsible study of detector behavior.

REPRODUCIBILITY STATEMENT

We support reproducibility by providing OpAI-Bench as a versioned benchmark together with the construction code, deterministic sentence-selection procedure, editing prompts, validation pipeline, data splits, and evaluation scripts in the supplemental materials. The benchmark construction procedure, version schedule, detector settings, evaluation metrics, and additional experimental details are described in the main paper and appendices.

REFERENCES

Rizky Adi, Bassamtiano Renaufalgi Irnawan, Yoshimi Suzuki, and Fumiyo Fukumoto. Gl-clic: Global-local coherence and lexical complexity for sentence-level ai-generated text detection. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 3600–3617, 2025.

Ekaterina Artemova, Jason S Lucas, Saranya Venkatraman, Joo-Young Lee, Sergei Tilga, Adaku Uchendu, and Vladislav Mikhailov. Beemo: Benchmark of expert-edited machine-generated outputs. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6992–7018, 2025.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. *arXiv preprint arXiv:2310.05130*, 2023.

Desklib. Desklib AI Text Detector v1.01. Hugging Face model, 2024. URL `https://huggingface.co/desklib/ai-text-detector-v1.01`. Fine-tuned DeBERTa-v3-large for AI-generated text detection. Accessed: 2026-05-04.

Liam Dugan, Alyssa Hwang, Filip Trhlík, Andrew Zhu, Josh Magnus Ludan, Hainiu Xu, Daphne Ippolito, and Chris Callison-Burch. Raid: A shared benchmark for robust evaluation of machine-generated text detectors. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12463–12492, 2024.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arXiv:2301.07597*, 2023.

Xinlei He, Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang. Mgtbench: Benchmarking machine-generated text detection. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 2251–2265, 2024.

Yongxin He, Shan Zhang, Yixuan Cao, Lei Ma, and Ping Luo. Detree: Detecting human-ai collaborative texts via tree-structured hierarchical representation learning. *arXiv preprint arXiv:2510.17489*, 2025.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. Radar: Robust ai-text detection via adversarial learning. *Advances in neural information processing systems*, 36:15077–15095, 2023.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. In *Proceedings of the 2021 conference of the north American chapter of the association for computational linguistics: Human language technologies*, pp. 1419–1436, 2021.

Lei Jiang, Desheng Wu, and Xiaolong Zheng. Sendetex: Sentence-level ai-generated text detection for human-ai hybrid content via style and context fusion. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5287–5302, 2025.

Learning Agency Lab. Learning agency lab – automated essay scoring 2.0. Kaggle competition, 2024. URL https://www.kaggle.com/competitions/learning-agency-lab-automated-essay-scoring-2. Accessed: 2026-05-07.

Eric Lei, Hsiang Hsu, and Chun-Fu Chen. Pald: Detection of text partially written by large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Dominik Macko, Robert Moro, Adaku Uchendu, Jason Lucas, Michiharu Yamashita, Matúš Pikuliak, Ivan Srba, Thai Le, Dongwon Lee, Jakub Simko, et al. Multitude: Large-scale multilingual machine-generated text detection benchmark. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9960–9987, 2023.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 1797–1807, 2018.

Sayak Paul. arxiv paper abstracts. Kaggle dataset, 2021. URL https://www.kaggle.com/datasets/spsayakpaul/arxiv-paper-abstracts. Accessed: 2026-04-17.

Shoumik Saha and Soheil Feizi. Almost ai, almost human: The challenge of detecting ai-polished writing. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25414–25431, 2025.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*, 2019.

Jinyan Su, Terry Zhuo, Di Wang, and Preslav Nakov. Detectllm: Leveraging log rank information for zero-shot detection of machine-generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 12395–12412, 2023.

Zhixiong Su, Yichen Wang, Herun Wan, Zhaohan Zhang, and Minnan Luo. Haco-det: A study towards fine-grained machine-generated text detection under human-ai coauthoring. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22015–22036, 2025.

LDM S Sai Teja, Annepaka Yadagiri, Partha Pakray, Chukhu Chunka, and Mangadoddi Srikar Vardhan. Fine-grained detection of ai-generated text using sentence-level segmentation. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 814–828, 2025.

LDM S Sai Teja, N Siva Gopala Krishna, Ufaq Khan, Muhammad Haris Khan, and Atul Mishra. Damasha: Detecting ai in mixed adversarial texts via segmentation with human-interpretable attribution. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 6189–6206, 2026.

Katherine Thai, Bradley Emi, Elyas Masrour, and Mohit Iyyer. Editlens: Quantifying the extent of ai editing in text. *arXiv preprint arXiv:2510.03154*, 2025.

Irina Tolstykh, Aleksandra Tsybina, Sergey Yakubson, Aleksandr Gordeev, Vladimir Dokholyan, and Maksim Kuprashevich. Gigacheck: Detecting llm-generated content. *arXiv preprint arXiv:2410.23728*, 2024.

Adaku Uchendu, Zeyu Ma, Thai Le, Rui Zhang, and Dongwon Lee. Turingbench: A benchmark environment for turing test in the age of neural text generation. In *Findings of the association for computational linguistics: EMNLP 2021*, pp. 2001–2016, 2021.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Pengyu Wang, Linyang Li, Ke Ren, Botian Jiang, Dong Zhang, and Xipeng Qiu. Seqxgpt: Sentence-level ai-generated text detection. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1144–1156, 2023.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, et al. M4gt-bench: Evaluation benchmark for black-box machine-generated text detection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3964–3992, 2024a.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Toru Sasaki, et al. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1369–1407, 2024b.

Junchao Wu, Runzhe Zhan, Derek F Wong, Shu Yang, Xinyi Yang, Yulin Yuan, and Lidia S Chao. Detectrl: Benchmarking llm-generated text detection in real-world scenarios. *Advances in Neural Information Processing Systems*, 37:100369–100401, 2024.

Cong Zeng, Shengkun Tang, Yuanzhou Chen, Zhiqiang Shen, Wenchao Yu, Xujiang Zhao, Haifeng Chen, Wei Cheng, and Zhiqiang Xu. Human texts are outliers: Detecting llm-generated texts via out-of-distribution detection. *arXiv preprint arXiv:2510.08602*, 2025.

Qihui Zhang, Chujie Gao, Dongping Chen, Yue Huang, Yixin Huang, Zhenyang Sun, Shilin Zhang, Weiye Li, Zhengyan Fu, Yao Wan, et al. Llm-as-a-coauthor: Can mixed human-written and machine-generated text be detected? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 409–436, 2024a.

Zhongping Zhang, Wenda Qin, and Bryan Plummer. Machine-generated text localization. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 8357–8371, 2024b.

APPENDIX

A DETECTOR DESCRIPTIONS

We evaluate the following AI-text detectors.

1. ***AdaLoc*** (Zhang et al., 2024b) is a sentence-level classifier over RoBERTa-large with a sliding window of three adjacent sentences: every window position emits an AI/human label, and overlapping scores are averaged into one prediction per sentence.
2. ***RADAR*** (Hu et al., 2023) is a Vicuna-7B classifier trained adversarially against a paraphraser: the paraphraser produces hard negatives during training, pushing the classifier to learn signals that survive paraphrasing.
3. ***Fast-DetectGPT*** (Bao et al., 2023) is zero-shot and scores a candidate by its *conditional probability curvature*: the likelihood of the observed tokens under a scoring LM is compared to the average likelihood of perturbed alternatives drawn from a sampling LM. AI text has higher curvature than human text and the whole score is computed in a single forward pass.
4. ***DAMASHA*** (Teja et al., 2026) is a token-level CRF tagger over a dual encoder. RoBERTa-base and ModernBERT-base read the same input; their hidden states are fused by an Info-Mask layer driven by simple stylistic features, and the CRF decodes the per-token AI/human tag sequence.
5. ***GigaCheck*** (Tolstykh et al., 2024) is a DETR-style span detector on top of Mistral-7B: the LM encodes tokens and a DETR decoder predicts a fixed-size set of character intervals, each labelled AI or human, alongside a coarse document-level head.
6. ***Desklib*** (Desklib, 2024) is a single-transformer document-level classifier; we use the public Hugging Face release out of the box, with no further training.
7. ***E5-small*** (Wang et al., 2022) is an E5-small encoder with a LoRA adapter trained for AI-text classification by the original authors. We use the public weights directly as a document-level binary classifier.
8. ***OOD-LLM-Detect*** (Zeng et al., 2025) treats AI-text detection as one-class classification: a Deep SVDD model is fitted to language-model embeddings of human text only, and a candidate is scored by its distance from the learnt human-text region.
9. ***RoBERTa-OpenAI*** (Solaiman et al., 2019) is RoBERTa-base fine-tuned by OpenAI on GPT-2 outputs; we use the released document-level binary classifier as is.
10. ***DetectLLM*** (Su et al., 2023) is zero-shot and combines two ranking statistics under a single reference causal LM: the log-rank ratio (LRR) and the normalised perturbation rank (NPR). Both capture how unusually high-ranked the observed tokens are under the reference distribution.
11. ***GL-CLiC*** (Adi et al., 2025) is a sentence-level classifier whose feature vector concatenates a DeBERTa contextual embedding, per-sentence global–local coherence scores, and per-sentence lexical complexity statistics; the resulting features are passed through a small classification head.
12. ***SeqXGPT*** (Wang et al., 2023) represents each token by its log-probability under four reference LMs (`gpt2-xl`, `gpt-neo-2.7B`, `gpt-j-6B`, `llama-7B`), yielding a $(T, 4)$ feature matrix. A small CNN + Transformer + CRF stack reads this matrix and emits per-word labels, which are aggregated to sentence level.
13. ***GPT-5.4*** (reasoning level: none) is an API-based judge prompted with the candidate document and asked to return a per-sentence AI/human label list directly.
14. ***Gemini 3 Flash*** (thinking level: minimal) is an API-based judge prompted with the candidate document and asked to return a per-sentence AI/human label list directly.
15. ***Claude Haiku 4.5*** (reasoning level: minimal) is an API-based judge prompted with the candidate document and asked to return a per-sentence AI/human label list directly.

16. ***GenAI-Sentence*** (Teja et al., 2025) is a token-level CRF tagger: a DeBERTa backbone feeds a BiGRU encoder, a linear classifier, and a CRF decoder that emits per-token AI/human labels; sentence labels are obtained by aggregation.

B Implementation Details

B.1 Text Normalization and Segmentation

All source documents are normalized prior to processing: line endings are standardized, consecutive blank lines are collapsed to a single paragraph boundary, and leading and trailing whitespace is removed. Documents are segmented into paragraphs by splitting on blank lines, and each paragraph is assigned a stable identifier preserved across all derived versions. Sentence segmentation is performed using NLTK's `sent_tokenize`, applied independently to each paragraph. Sentences receive stable identifiers assigned sequentially across paragraphs, used for sentence selection, provenance tracking, and document reconstruction across all nine versions.

B.2 Deterministic Sentence Selection

For each source document D with N sentences, a single fixed ordering $\pi(D) = (\pi_1, \ldots, \pi_N)$ is computed once and reused across all versions. The shuffle seed is derived deterministically from the document identifier, ensuring reproducibility and content-agnostic ordering. Given target coverage $c_{\text{sent}}^{(t)} \in [0, 1]$ at version t, the number of sentences selected is:

$$k^{(t)} = \left\lceil c_{\text{sent}}^{(t)} \cdot N \right\rceil . \tag{1}$$

The edited set is constructed cumulatively: previously selected sentences are always retained, and new sentences are drawn from $\pi(D)$ to reach $k^{(t)}$, guaranteeing $S^{(0)} \subseteq S^{(1)} \subseteq \cdots \subseteq S^{(8)}$.

B.3 Token- and Span-Level Provenance

Token- and span-level provenance labels are derived deterministically from the benchmark construction process using construction-time sentence markers, rather than from post-hoc source-to-rewrite alignment. This makes the annotations exact and reproducible with respect to the regions selected for AI rewriting.

Span derivation. During construction, each sentence selected for AI rewriting is wrapped in XML-style delimiters before being passed to the LLM, and the returned rewrite is stored with those delimiters intact. After each version is constructed, the tagged document $D_{\text{tagged}}^{(t)}$ is parsed to extract character-level AI spans $\mathcal{A}_{\text{char}}^{(t)} = \{(a_\ell, b_\ell)\}_{\ell=1}^{L}$, where a_ℓ and b_ℓ are character offsets in the clean text $D^{(t)}$. These span boundaries identify the regions corresponding to sentences produced through the AI rewriting step.

Word-level projection. Word-level labels are projected from character spans via whitespace tokenization. For word w_j with character span $[s_j, e_j]$:

$$y_j^{(t)} = \mathbf{1}\left[\exists\, (a_\ell, b_\ell) \in \mathcal{A}_{\text{char}}^{(t)} \text{ s.t. } \max(s_j, a_\ell) < \min(e_j, b_\ell)\right] . \tag{2}$$

Consecutive AI-labeled words are merged into contiguous spans. Because the editing unit is the sentence, each AI span corresponds to one rewritten sentence; boundaries therefore reflect sentence boundaries rather than intra-sentence partial rewrites. Accordingly, token- and span-level provenance denotes membership in an AI-revised region at token/span resolution and does not distinguish retained, modified, or inserted words within a rewritten sentence.

C BENCHMARK STATISTICS

Domain	Generator	# Traj.	# Versioned Samples	Avg. sents	Avg. tokens
Student essays	GPT-5.4	1,969	17,721	20.9	383.6
	GPT-5.4-nano	1,968	17,712	20.9	397.1
	Gemini-2.5-flash	3,969	35,721	21.0	407.2
News articles	GPT-5.4	1,904	17,136	24.0	477.0
	GPT-5.4-nano	1,998	17,982	24.0	479.4
	Gemini-2.5-flash	3,990	35,910	23.9	504.1
Government reports	GPT-5.4	2,000	18,000	20.5	539.4
	GPT-5.4-nano	2,000	18,000	20.6	552.5
	Gemini-2.5-flash	4,000	36,000	20.6	581.4
Scientific abstracts	GPT-5.4	1,764	15,876	11.0	224.1
	GPT-5.4-nano	1,764	15,876	11.0	227.7
	Gemini-2.5-flash	3,762	33,858	11.0	242.2
Total		31,088	279,792	19.3	426.1

Table 5: Main split statistics by domain and generator. Trajectories are generator-specific revision sequences initialized from human-written `v0` texts, and versioned samples count all versions from `v0` to `v8`.

Domain	Train	Development	Test	Total
Student essays	49,149	11,205	10,800	71,154
News articles	50,688	10,566	9,774	71,028
Government reports	50,049	10,881	11,070	72,000
Scientific abstracts	46,143	9,711	9,756	65,610
Total	196,029	42,363	41,400	279,792

Table 6: Train, development, and test row counts for the primary-generator benchmark subset, broken down by domain. Counts are reported over GPT-5.4, GPT-5.4-nano, and Gemini-2.5-Flash.

C.1 DETAILED RESULTS BY DOMAIN AND GENERATOR

This appendix provides the full per-domain and per-generator results that complement the aggregate analyses in the main text. For each evaluation granularity, we report both accuracy and F1-AI. Accuracy captures overall classification behavior, while F1-AI focuses on the detector's ability to identify AI-edited content, which is especially important under mixed-authorship settings where the class distribution changes across versions.

C.1.1 DOCUMENT-LEVEL RESULTS

Figures 4 and 5 report document-level accuracy and F1-AI, respectively, broken down by domain and generator. These figures show how zero-shot document-level detectors behave across the full revision trajectory. They also expose detector-specific and domain-specific variation that is hidden in the aggregate table, including cases where recovery at high AI coverage occurs only for particular generator–domain combinations.

C.1.2 SENTENCE-LEVEL RESULTS

Figures 6 and 7 report sentence-level accuracy and F1-AI for zero-shot and LLM-as-detector methods. Figures 8 and 9 additionally include fine-tuned sentence-level detectors. These breakdowns allow us to compare how detector families respond to progressive rewriting across domains and generators, and to assess whether sentence-level supervision improves stability under mixed-authorship revisions.

C.1.3 Token- and Span-Level Results

Figures 10 and 11 report fine-grained token/span-level performance by domain and generator. DAMASHA operates at the token level, while GigaCheck produces span-level predictions. These results complement the coarser document- and sentence-level analyses by showing how localization performance changes as AI-marked regions become denser across the revision trajectory.

D Aggregated Results

This appendix provides extended aggregate tables averaged over generators. Unlike Appendix C.1, which shows full domain–generator breakdowns, the tables below aggregate over generators and report results by domain and revision version. These tables provide a compact view of detector behavior across the full trajectory and include additional metrics beyond those shown in the main text.

D.1 Sentence Level

Table 10 reports sentence-level accuracy and F1-AI for all sentence-level detectors, averaged over generators. The table includes zero-shot detectors, LLM-as-detectors, and fine-tuned sentence-level models, enabling direct comparison across detector families. Table 11 provides the corresponding extended sentence-level metrics, including Macro-F1 and false negative rate. These additional metrics help characterize whether detectors fail by missing AI-edited content or by over-predicting the AI class.

D.2 Token and Span Level

Table 12 reports fine-grained token/span-level accuracy and F1-AI, averaged over generators. Table 13 provides the corresponding extended metrics. DAMASHA is evaluated at the token level, while GigaCheck is evaluated at the span level. These results complement the document- and sentence-level tables by evaluating whether detectors can localize AI-edited regions rather than only assign a global label. Because early revision versions contain relatively few AI-marked words or spans, accuracy can remain high even when the detector misses AI-edited regions. F1-AI is therefore particularly important for interpreting fine-grained performance, as it directly measures recovery of the AI-labeled class. The extended metrics further help distinguish between detectors that remain conservative and miss AI edits, and detectors that recover more AI-marked regions as coverage increases.

D.3 Document-level AUROC for Zero-shot Detectors

Table 7 reports threshold-independent AUROC for zero-shot document-level detectors. We compute AUROC by treating `v0` documents as the human class and `v1`–`v8` documents as the AI-involved class. This analysis complements native-threshold accuracy and F_1^{AI} by evaluating whether detector scores separate human from AI-involved versions independently of the default decision threshold.

Several detectors retain meaningful score separation despite weak native-threshold performance. *DeskLib*, *GigaCheck*, and *OOD-LLM-Detect* achieve average AUROCs of 0.724, 0.716, and 0.629, respectively, indicating that part of their weak binary performance is attributable to threshold transfer rather than a complete absence of ranking signal. In contrast, *RoBERTa-OpenAI* is close to chance on average (0.510), while *DetectLLM* and *Fast-DetectGPT* fall below chance under the original score-direction convention, suggesting weak or systematically inverted ranking on progressive mixed-authorship inputs. *RADAR* shows domain-dependent behavior: it retains ranking signal on essays and news, but is near chance on reports and inverted on abstracts.

D.4 Edit Distance and Semantic Similarity

Table 8 reports edit-distance, semantic-similarity, and sentence-level recall statistics for each trajectory stage. For each AI-labeled sentence at version v_i, we compute both stepwise and cumulative measurements. Stepwise measurements compare the sentence at v_i with the corresponding sentence

Domain	DeskLib	DetectLLM	E5-Small	Fast-DetectGPT	GigaCheck	OOD-LLM-Detect	RADAR	RoBERTa	Avg.
Abstracts	0.723	0.367	0.611	0.437	0.688	**0.813**	0.311	0.497	0.556
Essays	**0.873**	0.450	0.692	0.463	0.842	0.655	0.670	0.542	0.648
News	0.653	0.390	0.629	0.375	**0.687**	0.536	0.685	0.507	0.558
Reports	0.646	0.371	0.537	0.408	**0.647**	0.512	0.504	0.495	0.515
Average	**0.724**	0.395	0.617	0.421	0.716	0.629	0.542	0.510	0.569

Table 7: Document-level AUROC for zero-shot detectors by domain. AUROC is computed by treating `v0` documents as the human class and `v1`–`v8` documents as the AI-involved class.

Ver.	Operation	Cov.	Lev. step	Lev. cum.	Jac. step	Jac. cum.	Cos. step	Cos. cum.	Recall
v1	Polish	15%	0.112	0.112	0.159	0.159	0.968	0.968	0.215
v2	Paraphrase	25%	0.419	0.437	0.517	0.535	0.906	0.892	0.217
v3	Style	40%	0.337	0.414	0.458	0.520	0.919	0.881	0.248
v4	Compress	50%	0.308	0.416	0.375	0.501	0.942	0.883	0.220
v5	Expand	60%	0.336	0.482	0.350	0.544	0.925	0.859	0.290
v6	Style	75%	0.263	0.482	0.367	0.572	0.945	0.852	0.318
v7	Paraphrase	90%	0.297	0.466	0.410	0.553	0.942	0.860	0.317
v8	Polish	100%	0.045	0.432	0.071	0.515	0.991	0.868	0.342

Table 8: Edit distance, semantic similarity, and sentence-level recall by trajectory stage. Stepwise metrics compare each version with the previous version, while cumulative metrics compare each version with the original human sentence. Distances are computed over AI-labeled sentences. Cosine similarity is computed using `sentence-transformers/all-mpnet-base-v2`. Recall denotes sentence-level true positive rate averaged over sentence-level detectors.

at v_{i-1}, capturing the local change introduced by the current operation. Cumulative measurements compare the sentence at v_i with the corresponding original human sentence at `v0`, capturing the accumulated change over the full trajectory.

We report normalized Levenshtein distance, Jaccard distance, and embedding cosine similarity. Normalized Levenshtein distance measures surface-level edit distance after length normalization, while Jaccard distance measures token-set overlap. Semantic similarity is measured using cosine similarity between sentence embeddings from `sentence-transformers/all-mpnet-base-v2`. All statistics are averaged over AI-labeled sentences at each version, and recall denotes sentence-level true positive rate averaged over sentence-level detectors.

The results show that OpAI-Bench includes both light polishing and substantial reformulation. Polishing produces the smallest stepwise changes, with normalized Levenshtein distances of 0.112 at `v1` and 0.045 at `v8`. However, these two polishing stages have different trajectory contexts: `v1` is an early light edit, whereas `v8` is a final polishing stage applied after substantial accumulated revision. In contrast, paraphrasing, style rewriting, compression, and expansion introduce larger stepwise changes, with normalized Levenshtein distances ranging from 0.263 to 0.419. The cumulative measurements show that later versions are not simply lightly edited human originals. From `v2` onward, cumulative Levenshtein distance remains between 0.414 and 0.482, and even the final polishing stage at `v8` has a cumulative distance of 0.432 from the original human sentence. At the same time, cosine similarity remains high across stages, indicating that the revisions largely preserve semantic content while changing surface form, structure, and style. Detector recall is therefore not determined by edit distance alone. For example, `v4` has substantial stepwise and cumulative edit distance, yet recall decreases relative to adjacent stages. Conversely, `v8` has the smallest stepwise edit distance but the highest recall, reflecting the effect of accumulated revision history. These results reinforce the central finding that detectability depends jointly on coverage, operation type, and trajectory history, rather than only on the amount of lexical change.

E Ablations

We provide three controlled ablations to isolate the factors that vary jointly in the main trajectory. Ablation 1 fixes the edit operation and varies AI coverage, testing whether detector behavior is

driven by coverage alone. Ablation 2 fixes AI coverage and varies the edit operation, testing whether different operations remain distinguishable at matched coverage. Ablation 3 compares cumulative editing with independent editing from the original source, testing the effect of accumulated revision history. Sections E.1–E.3 report the primary ablation results using *Gemini-2.5-Flash*; Section E.4 repeats the same ablation protocol with *GPT-5.4-nano* to assess whether the findings are robust to the choice of editing model.

E.1 Ablation 1: Coverage-controlled Edit Operations

This ablation fixes the edit operation and varies the target AI coverage. Sentence-level accuracy and F1-AI are shown in Figures 12 and 13, respectively. Document-level accuracy and F1-AI are shown in Figures 14 and 15, respectively.

E.2 Ablation 2: Fixed-coverage Edit Operations

This ablation isolates the effect of edit operation at a fixed amount of AI editing. For each target AI coverage level separately (25%, 50%, and 75%), we hold coverage constant and vary only the edit operation among compress, paraphrase, and expand. Sentence-level accuracy and F1-AI are shown in Figures 16 and 17, respectively. Document-level accuracy and F1-AI are shown in Figures 18 and 19, respectively.

E.3 Ablation 3: Independent versus Cumulative Editing

This ablation compares the main cumulative trajectory with an alternative setting in which each version is edited independently from the original human source. Sentence-level results are shown in Figure 20.

E.4 Second-generator Robustness Analysis

To assess whether the controlled-ablation findings are specific to a single editing model, we repeat the three ablations with *GPT-5.4-nano*. We keep the ablation protocol unchanged: the same coverage levels, edit operations, cumulative-versus-independent editing setup, detector configurations, and evaluation metrics are used as in Sections E.1–E.3. This second-generator analysis supports the same qualitative conclusion as the *Gemini-2.5-Flash* ablations: AI coverage alone does not fully explain detectability, and detector behavior also depends on edit operation and cumulative revision history. At the same time, the relative difficulty of specific operations varies across domains and detector families, indicating that operation effects should be interpreted as generator- and detector-dependent rather than as a universal ordering.

Coverage-controlled edit operations. Sentence-level accuracy and F1-AI are shown in Figures 22 and 23, respectively. Document-level accuracy and F1-AI are shown in Figures 24 and 25, respectively.

Fixed-coverage edit operations. Sentence-level accuracy and F1-AI are shown in Figures 26 and 27, respectively. Document-level accuracy and F1-AI are shown in Figures 28 and 29, respectively.

Independent versus cumulative editing. Sentence-level results are shown in Figure 30, and document-level results are shown in Figure 31.

F Experimental Details

For clarity, we summarize the label notation used across the experimental settings. For a document trajectory $\mathcal{T}(D) = (D^{(0)}, \ldots, D^{(8)})$, document-level labels are $y_{\text{doc}}^{(t)} = \mathbf{1}[t > 0]$. For sentence-level attribution, the label of sentence i at version t is $y_i^{(t)} = \mathbf{1}[i \in S^{(t)}]$, where $S^{(t)}$ is the cumulative set of sentences edited by version t. For token-level localization, tokens refer to word-level units:

$Y_{\text{tok}}^{(t)} = (y_1^{(t)}, \ldots, y_M^{(t)})$, with $y_j^{(t)} \in \{0, 1\}$ indicating whether word j belongs to an AI-edited span. The realized token coverage is

$$c_{\text{tok}}^{(t)} = \frac{1}{M} \sum_{j=1}^{M} y_j^{(t)}. \tag{3}$$

Span-level annotations are represented as character spans $\mathcal{A}_{\text{char}}^{(t)} = \{(a_\ell, b_\ell)\}_{\ell=1}^{L}$, where a_ℓ and b_ℓ are character offsets in $D^{(t)}$.

F.1 Zero-Shot Detectors

For zero-shot detectors, we use either released checkpoints or models reproduced from the original training scripts, following the inference settings provided by the original papers or released implementations. These methods are evaluated directly on OpAI-Bench without task-specific training or threshold tuning on the benchmark. Depending on the detector output, we evaluate document-level predictions, sentence-level predictions, or fine-grained token and span predictions under the protocol described in the main text.

F.2 Language Models as Detectors

We evaluate frontier language models as prompted sentence-level detectors. For each document version, the text is split into numbered sentences and passed to the model with a shared prompt template. The model returns a binary label and a confidence score for each sentence. We use the returned binary labels for sentence-level evaluation.

Sentence-Level LLM-as-Detector Prompt

User prompt

```
You are an expert linguist and writing analyst specializing in
distinguishing human-written text from AI-generated text.

The following text has been split into numbered sentences.  For EACH
sentence, classify it as human-written (0) or AI-generated (1), and
estimate the probability that it is AI-generated.

Text:
"""
[numbered_sentences]
"""

Respond in JSON format:
{"labels":  [0, 1, ...], "confidences":  [0.1, 0.9, ...]}

- labels:  array of integers (0 = human-written, 1 = AI-generated),
one per sentence.
- confidences:  array of floats (0.0 to 1.0), one per sentence.
- Both arrays must contain exactly [num_sentences] elements, in
sentence order.
- Do not include any other keys or text outside the JSON object.
```

F.3 Training of OpAI-Bench detector variants

For the trained setting we re-train (or LoRA-fine-tune) the architecture of each method on the OpAI-Bench training split from the three in-distribution generators (GPT-5.4, GPT-5.4-nano, Gemini-2.5-Flash). Qwen3-8B is excluded from training and used only for cross-generator evaluation. For GPT-5.4-nano, Gemini-2.5-Flash we sample a subset for data-balance. Sentence-level models consume sentence provenance labels, document-level labels are derived from whether a version contains AI-edited content, and fine-grained models consume the token or span annotations available in the benchmark. Models are selected on the development split and evaluated on the test split.

We follow the architecture, optimiser, and trainable-parameter choices of each method's original implementation; deviations are listed in the *Setting* column of Table 9. All LoRA-tuned variants

Detector	Backbone	LoRA targets	LR	Batch	Epochs	Setting
AdaLoc	RoBERTa-large (openai-detector)	q, v	10^{-5}	32	2	Adam
GenAI-Sentence	DeBERTa-v3-base	query_proj, value_proj	10^{-5}	32	2	AdamW, bf16
SeqXGPT	CNN+Transformer+CRF over 4-LM feats.	—	5×10^{-5}	32	20	weight decay 0.1
DAMASHA	RoBERTa-base + ModernBERT-base	q, v / Wqkv	2×10^{-4}	16	5	AdamW, wd 0.01, ext. CRF NLL
GigaCheck	Mistral-7B-v0.3	q_proj, v_proj	3×10^{-5}	$4\times ga\,4$	5	bf16, DeepSpeed ZeRO-2

Table 9: Training hyperparameters for the OpAI-Bench-trained detector variants. The *LoRA targets* column lists the modules to which the LoRA adapter is attached; entries marked "–" indicate full fine-tuning of the listed module set, with no LoRA. Rank r=8, α=16, dropout 0.1 throughout.

share a single adapter configuration: rank r=8, α=16, dropout 0.1 — so the table reports only the target modules. Backbone weights are kept frozen, while the task-specific heads (classifier, BiGRU, CRF, fusion module, etc.) are trained in full. A linear schedule with 10% warmup is used everywhere.

GLCLIC. GLCLIC is trained with the official upstream pipeline at its released default configuration; we do not override any of its hyperparameters.

Compute resources. All local detector training, inference, and aggregation experiments were run on an internal Linux server equipped with NVIDIA GeForce RTX 4090 GPUs (24GB memory each). Lightweight preprocessing, metric aggregation, and table generation were run on CPU on the same server. API-based dataset generation and LLM-as-detector experiments used external model APIs, with local compute used for request orchestration, parsing, and metric computation. Preliminary checks used the same compute environment and are not separately reported.

G Editing Prompts

The following prompts are used to generate the five editing operations in OpAI-Bench. In all cases, the model is instructed to rewrite only the target sentence, preserve factual content and named entities, and return exactly one sentence.

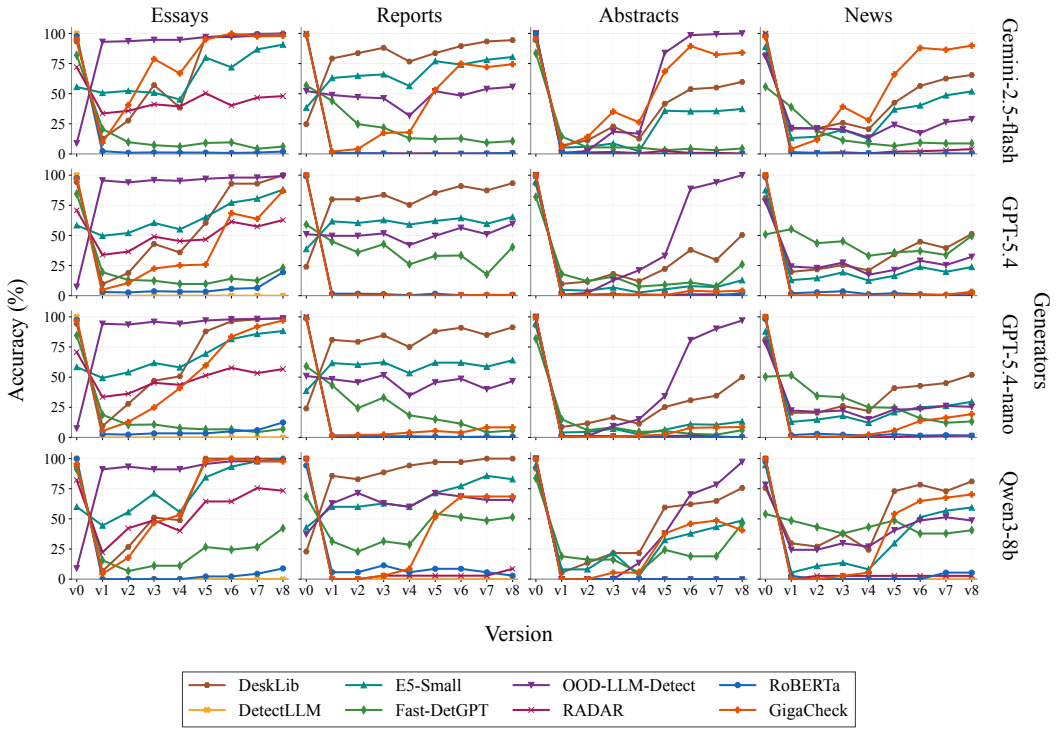

Figure 4: Document-level accuracy broken down by domain and generator.

Polish Prompt

System instruction

```
You are a precise text rewriting assistant.  You must output ONLY
the rewritten target sentence.    Return ONLY the rewritten text ---
nothing else.

STRICT OUTPUT RULES:
- Do NOT start with phrases like 'Here is', 'Here's', 'Sure',
'Certainly',
'Of course', 'Rewritten:', 'Revised:', 'Result:'.
- Do NOT explain what you did.
- Do NOT add quotes around your output.
- Do NOT add any prefix or suffix.
- Your entire response = the rewritten text only.
- Preserve meaning.  Do NOT add new facts.
- Preserve all names, numbers, dates, locations, and other entities
exactly.
- Do NOT introduce any new named entities, numbers, or specific
claims.
- Your task is proofreading --- fix errors and improve fluency only.
```

User prompt

```
Operation:  polish


Constraints:
- Rewrite ONLY the target sentence.
- Do NOT add new facts.
- Preserve names, numbers, and entities.
- Keep the same language as the target.
- Return ONLY the rewritten text.  No explanations, no quotes, no
prefixes.
- Length constraint:  85%--100% of original word count.
- No line breaks.
- Output EXACTLY ONE sentence.

Guidance:  Make light edits to improve grammar, punctuation,
fluency, and clarity while preserving meaning.  Keep the sentence
structure mostly unchanged.  The output must not be identical to the
input.  Keep it as exactly one sentence.  Do not add new facts.

Paragraph context (for coherence only; do not rewrite it):
[context]

Target sentence:
[sentence]
```

Paraphrase Prompt

System instruction

```
You are a precise text rewriting assistant.  You must output ONLY
the rewritten target sentence.
Return ONLY the rewritten text --- nothing else.

STRICT OUTPUT RULES:
- Do NOT start with phrases like 'Here is', 'Here's', 'Sure',
'Certainly',
'Of course', 'Rewritten:', 'Revised:', 'Result:'.
- Do NOT explain what you did.
- Do NOT add quotes around your output.
- Do NOT add any prefix or suffix.
- Your entire response = the rewritten text only.
- Preserve meaning.  Do NOT add new facts.
- Preserve all names, numbers, dates, locations, and other entities
exactly.
- Do NOT introduce any new named entities, numbers, or specific
claims.
- Your task is paraphrasing --- same meaning, different wording.
```

User prompt

```
Operation:  paraphrase


Constraints:
- Rewrite ONLY the target sentence.
- Do NOT add new facts.
- Preserve names, numbers, and entities.
- Keep the same language as the target.
- Return ONLY the rewritten text.  No explanations, no quotes, no
prefixes.
- Length constraint:  85%--115% of original word count.
- No line breaks.
- Output EXACTLY ONE sentence.

Guidance:  Rewrite the sentence with clearly different wording while
preserving meaning.  You may restructure the phrasing, but keep the
same core content.  The output must not be identical to the input.
Keep it as exactly one sentence.  Do not add new facts.

Paragraph context (for coherence only; do not rewrite it):
[context]

Target sentence:
[sentence]
```

Style Rewrite Prompt

System instruction

```
You are a precise text rewriting assistant.  You must output ONLY
the rewritten target sentence.
Return ONLY the rewritten text --- nothing else.

STRICT OUTPUT RULES:
- Do NOT start with phrases like 'Here is', 'Here's', 'Sure',
'Certainly',
'Of course', 'Rewritten:', 'Revised:', 'Result:'.
- Do NOT explain what you did.
- Do NOT add quotes around your output.
- Do NOT add any prefix or suffix.
- Your entire response = the rewritten text only.
- Preserve meaning.  Do NOT add new facts.
- Preserve all names, numbers, dates, locations, and other entities
exactly.
- Do NOT introduce any new named entities, numbers, or specific
claims.
- Your task is style transfer --- same meaning, different
tone/voice.
```

User prompt

```
Operation:  style


Constraints:
- Rewrite ONLY the target sentence.
- Do NOT add new facts.
- Preserve names, numbers, and entities.
- Keep the same language as the target.
- Return ONLY the rewritten text.  No explanations, no quotes, no
prefixes.
- Length constraint:  85%--115% of original word count.
- No line breaks.
- Output EXACTLY ONE sentence.

Guidance:  Rewrite the sentence in a formal, natural, human-written
style by changing tone, register, and phrasing while preserving
meaning.  Do not change the underlying facts or add new content.
The output must not be identical to the input.  Keep it as exactly
one sentence.

Paragraph context (for coherence only; do not rewrite it):
[context]

Target sentence:
[sentence]
```

Compress Prompt

System instruction

```
You are a precise text rewriting assistant.  You must output ONLY
the rewritten target sentence.
Return ONLY the rewritten text --- nothing else.

STRICT OUTPUT RULES:
- Do NOT start with phrases like 'Here is', 'Here's', 'Sure',
'Certainly',
'Of course', 'Rewritten:', 'Revised:', 'Result:'.
- Do NOT explain what you did.
- Do NOT add quotes around your output.
- Do NOT add any prefix or suffix.
- Your entire response = the rewritten text only.
- Preserve meaning.  Do NOT add new facts.
- Preserve all names, numbers, dates, locations, and other entities
exactly.
- Do NOT introduce any new named entities, numbers, or specific
claims.
- Your task is compression --- shorter text, same meaning, no facts
lost.
```

User prompt

```
Operation:  compress
Style target (if applicable):  formal, natural, human-written

Constraints:
- Rewrite ONLY the target sentence.
- Do NOT add new facts.
- Preserve names, numbers, and entities.
- Keep the same language as the target.
- Return ONLY the rewritten text.  No explanations, no quotes, no
prefixes.
- Length constraint:  60%--80% of original word count.
- No line breaks.
- Output EXACTLY ONE sentence.

Guidance:  Rewrite the sentence to be shorter and more concise
while preserving all essential meaning.  Remove redundancy and
non-essential phrasing, but do not omit important information.  The
output must not be identical to the input.  Keep it as exactly one
sentence.  Do not add new facts.

Paragraph context (for coherence only; do not rewrite it):
[context]

Target sentence:
[sentence]
```

Expand Prompt

System instruction

```
You are a precise text rewriting assistant.  You must output ONLY
the rewritten target sentence.
Return ONLY the rewritten text --- nothing else.

STRICT OUTPUT RULES:
- Do NOT start with phrases like 'Here is', 'Here's', 'Sure',
'Certainly',
'Of course', 'Rewritten:', 'Revised:', 'Result:'.
- Do NOT explain what you did.
- Do NOT add quotes around your output.
- Do NOT add any prefix or suffix.
- Your entire response = the rewritten text only.
- Preserve meaning.  Do NOT add new facts.
- Preserve all names, numbers, dates, locations, and other entities
exactly.
- Do NOT introduce any new named entities, numbers, or specific
claims.
- Your task is expansion --- extend current text.
```

User prompt

```
Operation:  expand
Style target (if applicable):  formal, natural, human-written

Constraints:
- Rewrite ONLY the target sentence.
- Do NOT add new facts.
- Preserve names, numbers, and entities.
- Keep the same language as the target.
- Return ONLY the rewritten text.  No explanations, no quotes, no
prefixes.
- Length constraint:  120%--150% of original word count.
- No line breaks.
- Output EXACTLY ONE sentence.

Guidance:  Rewrite the sentence to be slightly more detailed using
only information already stated or directly implied.  You may add
brief clarifying or descriptive phrasing, but do not introduce new
facts, examples, names, dates, numbers, or claims.  The output must
not be identical to the input.  Keep it as exactly one sentence.

Paragraph context (for coherence only; do not rewrite it):
[context]

Target sentence:
[sentence]
```

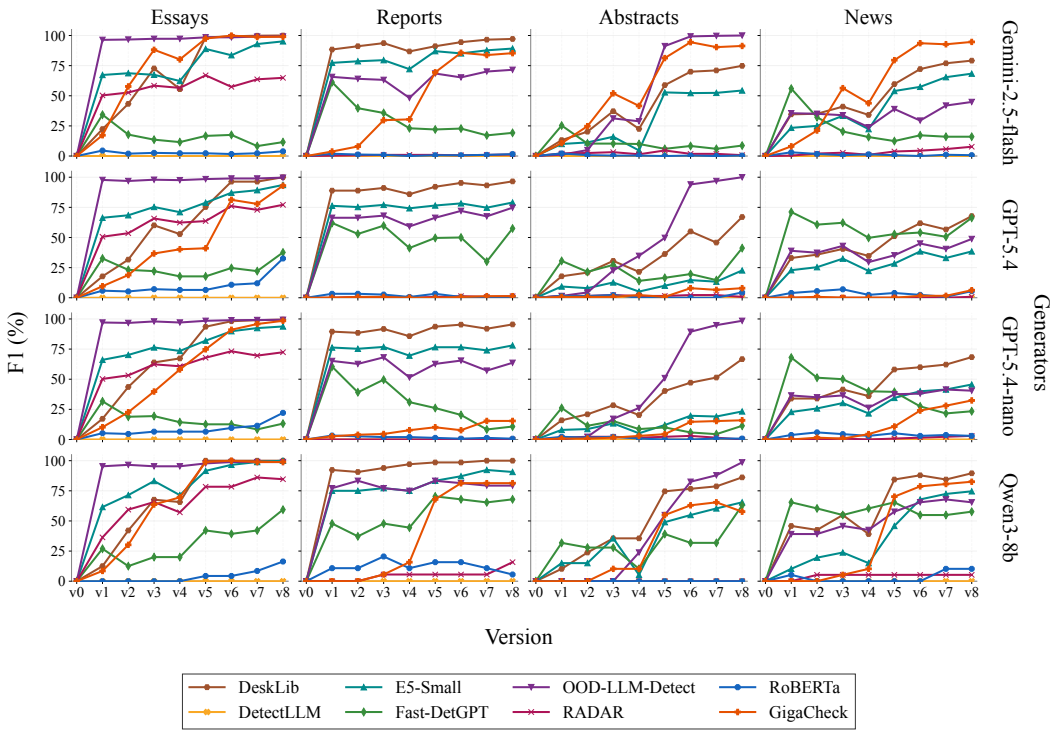

Figure 5: Document-level F1-AI across revision versions, domains, and generators. Each curve represents a document-level detector, showing how AI-targeted detection performance changes along the progressive human-to-AI revision trajectory.

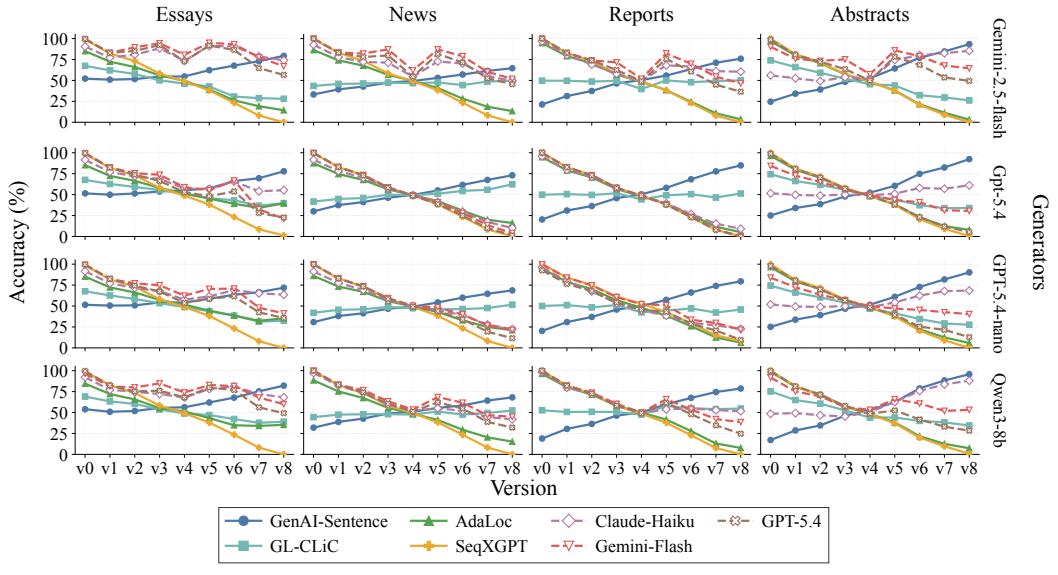

Figure 6: Sentence-level accuracy across revision versions, domains, and generators. The figure compares sentence-level detectors across the full revision trajectory, including zero-shot methods and LLM-as-detectors sentence-level models.

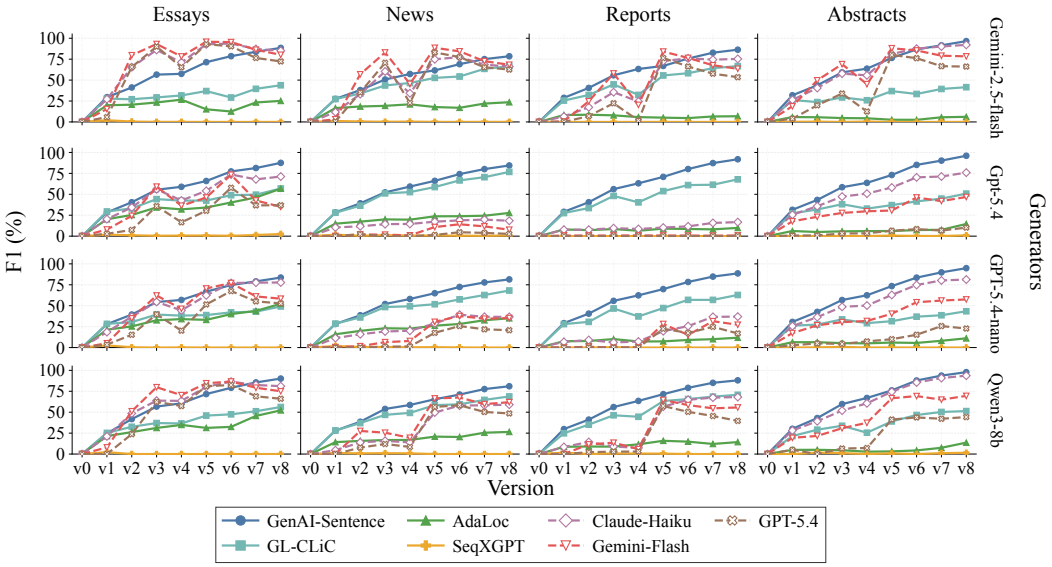

Figure 7: Sentence-level F1-AI across revision versions, domains, and generators. The figure compares sentence-level detectors across the full revision trajectory, including zero-shot methods and LLM-as-detectors.

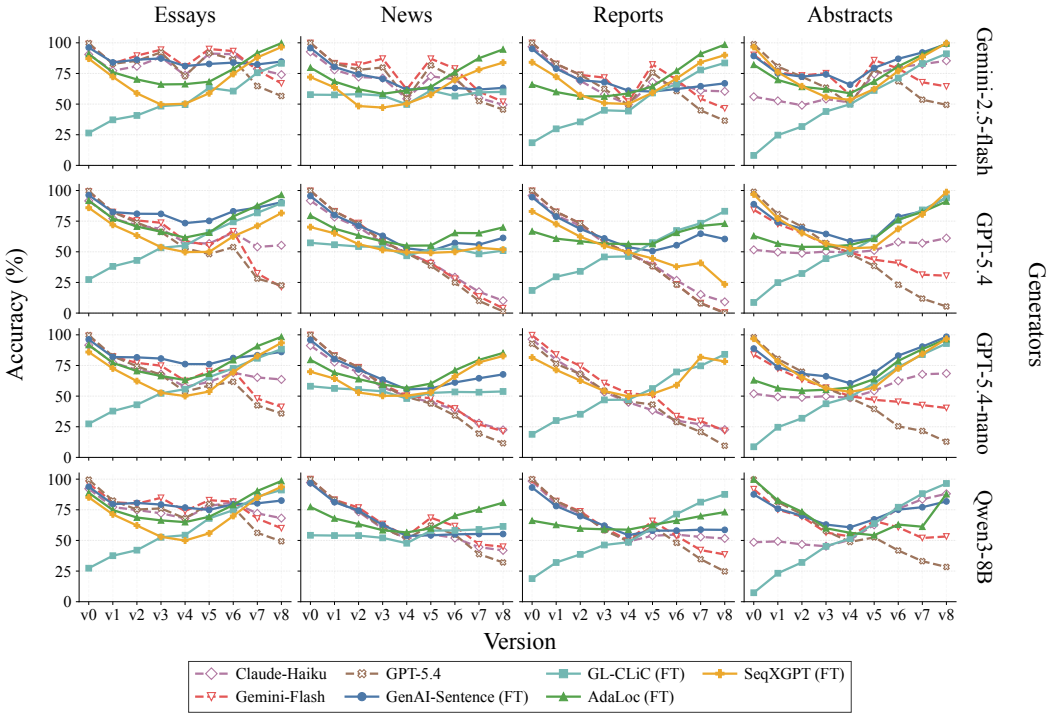

Figure 8: Sentence-level accuracy across revision versions, domains, and generators, including fine-tuned sentence-level models. The figure highlights how fine-tuned detectors compare with LLM-as-detector methods along the progressive human-to-AI revision trajectory.

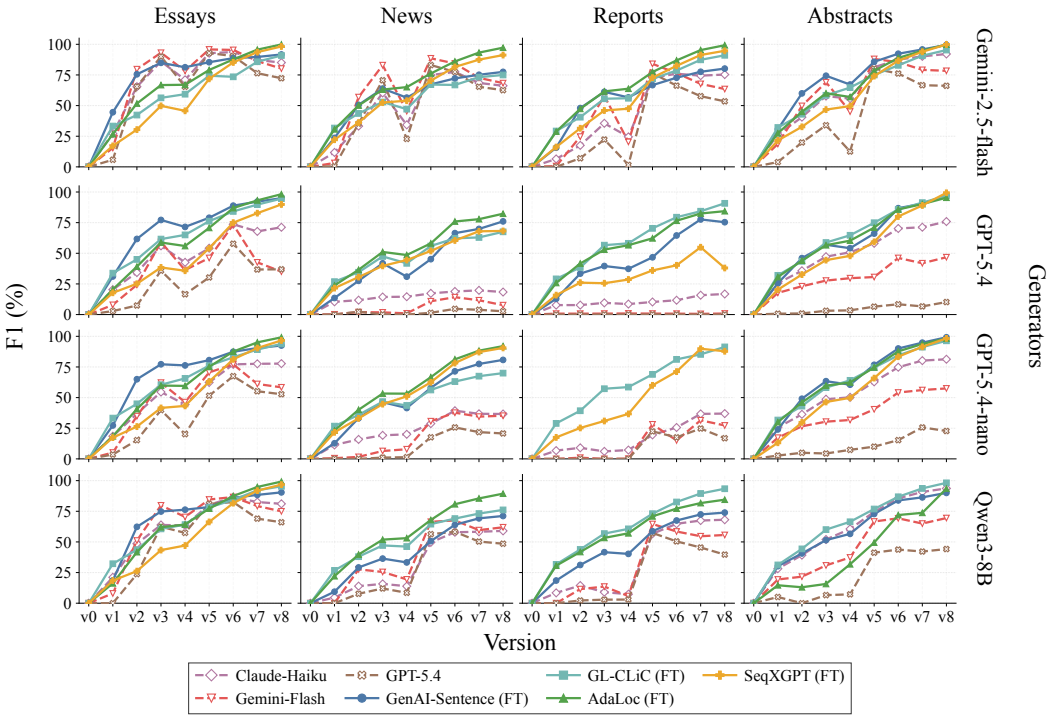

Figure 9: Sentence-level F1-AI across revision versions, domains, and generators, including fine-tuned sentence-level models. The figure highlights how fine-tuned detectors compare with LLM-as-detector methods along the progressive human-to-AI revision trajectory.

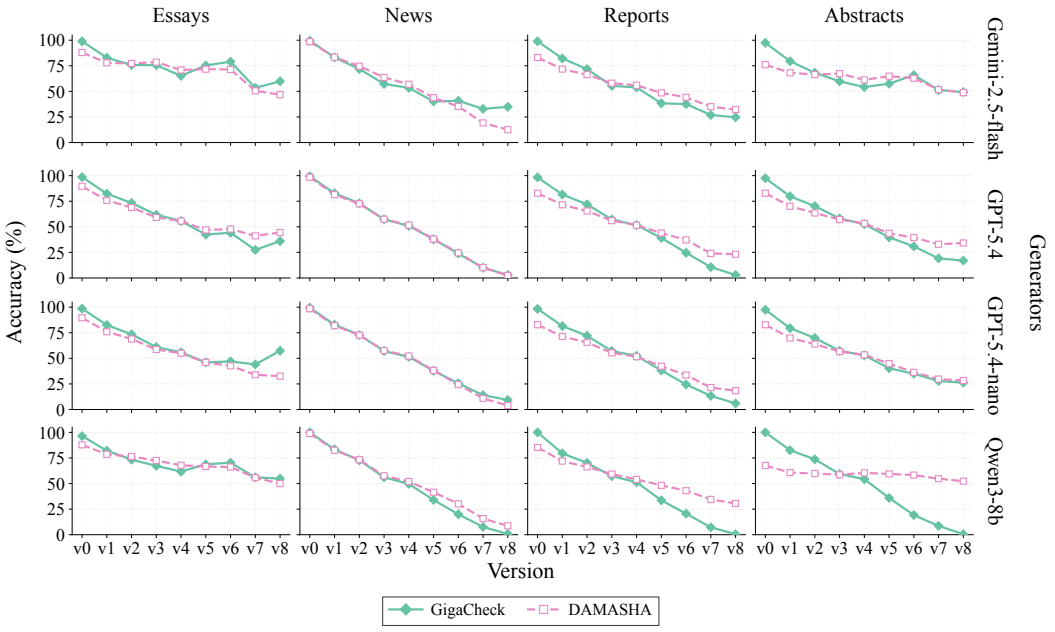

Figure 10: Token- and span-level accuracy broken down by domain and generator.

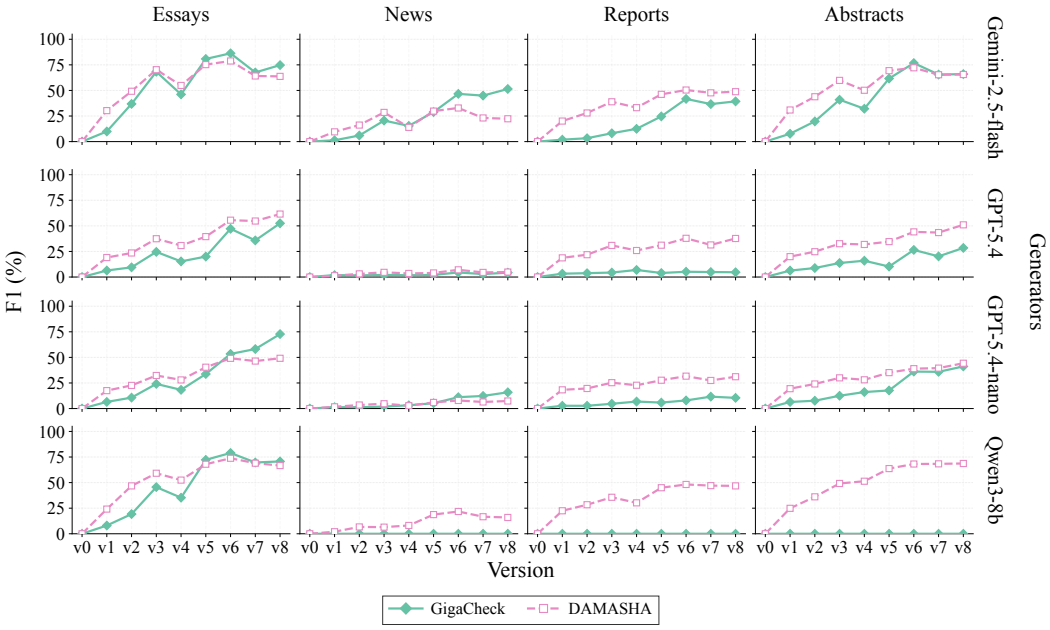

Figure 11: Token- and span-level F1-AI broken down by domain and generator.

Detector	Ver.	Essays		Reports		News		Abstracts	
		Acc	F1-AI	Acc	F1-AI	Acc	F1-AI	Acc	F1-AI
LLM-as-detector									
Claude-Haiku	$v0$	91.2	0.0	96.4	0.0	91.9	0.0	53.2	0.0
	$v1$	77.2 -13.9	22.0 +22.0	79.9 -16.5	7.0 +7.0	78.0 -13.9	11.2 +11.2	50.6 -2.6	25.9 +25.9
	$v2$	74.7 -2.5	45.5 +23.5	69.4 -10.5	11.4 +4.4	70.5 -7.5	20.3 +9.1	48.9 -1.6	37.6 +11.7
	$v3$	75.0 +0.3	65.5 +20.0	56.2 -13.2	17.1 +5.6	61.9 -8.6	31.3 +11.0	51.4 +2.5	51.2 +13.7
	$v4$	62.9 -12.1	52.7 -12.9	47.8 -8.4	13.4 -3.6	51.2 -10.7	23.0 -8.3	49.8 -1.6	52.2 +1.0
	$v5$	69.8 +7.0	69.7 +17.1	48.7 +0.9	34.2 +20.8	52.9 +1.7	40.3 +17.3	60.0 +10.2	67.4 +15.2
	$v6$	75.2 +5.4	81.5 +11.7	41.1 -7.6	37.8 +3.6	46.2 -6.7	45.2 +4.9	66.8 +6.9	77.5 +10.1
	$v7$	66.0 -9.2	77.5 -4.0	34.4 -6.7	42.3 +4.5	33.4 -12.7	41.7 -3.5	69.1 +2.3	80.5 +3.0
	$v8$	64.3 -1.7	78.0 +0.5	30.7 -3.6	43.0 +0.7	27.3 -6.1	40.4 -1.3	71.7 +2.5	83.1 +2.6
Gemini-Flash	$v0$	99.1	0.0	99.8	0.0	99.8	0.0	86.1	0.0
	$v1$	82.6 -16.5	9.5 +9.5	83.1 -16.7	0.5 +0.5	83.1 -16.6	1.3 +1.3	73.7 -12.5	17.9 +17.9
	$v2$	80.8 -1.8	46.3 +36.8	73.9 -9.2	8.8 +8.3	76.4 -6.8	20.2 +18.9	67.4 -6.3	33.1 +15.2
	$v3$	81.0 +0.2	71.5 +25.2	63.4 -10.4	19.6 +10.8	68.4 -8.0	30.4 +10.2	62.7 -4.8	42.4 +9.2
	$v4$	67.3 -13.7	53.6 -17.9	51.0 -12.4	7.1 -12.4	53.8 -14.6	18.1 -12.3	52.3 -10.4	35.5 -6.9
	$v5$	73.5 +6.3	70.9 +17.2	57.2 +6.2	37.7 +30.5	58.7 +4.8	43.4 +25.3	58.8 +6.5	53.0 +17.5
	$v6$	76.9 +3.4	82.0 +11.1	42.3 -14.9	30.8 -6.9	49.0 -9.7	45.4 +2.0	55.1 -3.7	61.9 +8.9
	$v7$	52.7 -24.2	63.3 -18.6	30.8 -11.5	33.3 +2.5	33.6 -15.4	39.6 -5.8	47.3 -7.8	59.1 -2.9
	$v8$	43.1 -9.6	57.9 -5.4	22.9 -7.9	30.5 -2.7	25.8 -7.8	37.1 -2.5	45.1 -2.2	60.9 +1.8
GPT-5.4	$v0$	99.7	0.0	97.6	0.0	99.9	0.0	98.6	0.0
	$v1$	82.6 -17.1	4.1 +4.1	80.7 -16.9	0.0 +0.0	83.1 -16.8	0.2 +0.2	80.6 -18.0	2.4 +2.4
	$v2$	77.8 -4.8	29.4 +25.4	71.7 -9.0	2.3 +2.3	74.7 -8.4	12.5 +12.4	70.9 -9.6	8.6 +6.2
	$v3$	75.3 -2.5	55.2 +25.8	58.3 -13.4	7.4 +5.1	65.6 -9.1	23.9 +11.4	59.1 -11.8	13.8 +5.2
	$v4$	59.8 -15.4	34.0 -21.2	47.8 -10.5	0.5 -6.9	50.9 -14.7	8.1 -15.8	48.8 -10.3	7.8 -6.0
	$v5$	66.1 +6.3	58.2 +24.2	52.3 +4.5	33.0 +32.4	54.8 +3.9	34.0 +26.0	52.1 +3.2	32.2 +24.4
	$v6$	67.3 +1.2	71.9 +13.7	37.6 -14.8	27.9 -5.0	43.5 -11.3	36.0 +1.9	39.0 -13.0	33.3 +1.1
	$v7$	45.1 -22.2	56.1 -15.8	24.5 -13.1	27.5 -0.5	27.3 -16.3	30.4 -5.6	29.0 -10.0	33.0 -0.3
	$v8$	38.4 -6.7	54.0 -2.1	15.3 -9.2	23.5 -4.0	19.5 -7.8	28.7 -1.7	22.5 -6.5	33.0 +0.0
Zero-shot									
AdaLoc (Zhang et al., 2024b)	$v0$	85.1	0.0	94.7	0.0	86.6	0.0	96.7	0.0
	$v1$	72.3 -12.8	20.5 +20.5	79.0 -15.7	7.9 +7.9	73.9 -12.7	15.8 +15.8	79.6 -17.1	6.1 +6.1
	$v2$	66.2 -6.1	23.7 +3.3	70.5 -8.5	7.9 -0.0	67.4 -6.5	18.6 +2.8	70.3 -9.3	5.7 -0.5
	$v3$	57.1 -9.1	30.1 +6.3	56.9 -13.6	8.8 +0.9	56.3 -11.1	20.8 +2.2	57.2 -13.1	5.2 -0.4
	$v4$	50.9 -6.2	30.9 +0.8	48.1 -8.7	6.5 -2.3	49.4 -6.9	21.1 +0.3	48.4 -8.8	5.1 -0.1
	$v5$	43.0 -8.0	27.5 -3.4	38.6 -9.5	7.2 +0.7	42.0 -7.4	22.5 +1.4	38.3 -10.2	4.9 -0.2
	$v6$	34.7 -8.3	30.9 +3.4	25.4 -13.3	7.4 +0.3	30.7 -11.3	23.0 +0.4	22.3 -16.0	5.1 +0.2
	$v7$	28.6 -6.1	37.8 +6.9	11.6 -13.8	8.2 +0.8	21.2 -9.5	26.3 +3.4	12.1 -10.2	7.1 +2.0
	$v8$	29.8 +1.2	44.7 +6.9	5.0 -6.6	9.5 +1.3	16.9 -4.3	28.8 +2.4	5.6 -6.5	10.6 +3.5
GenAI-Sentence (Teja et al., 2025)	$v0$	51.7	0.0	20.6	0.0	31.4	0.0	25.0	0.0
	$v1$	50.5 -1.2	28.6 +28.6	31.0 +10.4	29.0 +29.0	38.3 +6.9	28.0 +28.0	34.0 +9.0	31.3 +31.3
	$v2$	51.2 +0.8	40.4 +11.8	36.9 +5.8	40.6 +11.5	41.7 +3.4	38.6 +10.6	39.0 +5.0	43.2 +11.9
	$v3$	54.2 +3.0	55.5 +15.1	46.0 +9.1	55.9 +15.3	46.8 +5.1	51.7 +13.1	47.7 +8.7	58.2 +14.9
	$v4$	54.7 +0.5	57.8 +2.3	50.2 +4.2	62.9 +7.0	49.4 +2.6	58.2 +6.5	52.1 +4.4	63.3 +5.1
	$v5$	59.5 +4.8	68.0 +10.2	57.1 +6.9	69.2 +6.3	54.1 +4.7	64.2 +6.0	62.0 +9.9	74.3 +11.1
	$v6$	65.8 +6.3	76.9 +8.9	66.0 +8.9	78.1 +8.9	59.6 +5.5	71.9 +7.7	74.9 +13.0	85.1 +10.8
	$v7$	69.7 +3.9	81.4 +4.6	74.3 +8.3	84.9 +6.8	64.5 +4.9	77.7 +5.8	83.0 +8.0	90.5 +5.4
	$v8$	76.4 +6.7	86.6 +5.1	80.1 +5.8	88.9 +4.0	68.8 +4.2	81.5 +3.8	92.0 +9.0	95.8 +5.3
GL-CLiC (Adi et al., 2025)	$v0$	67.6	0.0	49.9	0.0	42.3	0.0	74.2	0.0
	$v1$	62.5 -5.1	28.6 +28.6	50.4 +0.5	27.1 +27.1	45.4 +3.0	28.0 +28.0	66.1 -8.1	26.6 +26.6
	$v2$	58.4 -4.1	30.2 +1.7	48.9 -1.5	32.2 +5.1	46.1 +0.7	35.5 +7.5	60.4 -5.7	27.7 +1.1
	$v3$	53.1 -5.4	37.7 +7.4	50.8 +1.9	46.5 +14.4	48.7 +2.6	47.7 +12.2	53.9 -6.5	33.7 +6.0
	$v4$	48.6 -4.5	37.4 -0.2	42.2 -8.6	36.7 -9.9	47.8 -0.9	49.5 +1.8	46.9 -7.1	29.3 -4.4
	$v5$	44.1 -4.4	39.4 +2.0	48.0 +5.8	52.2 +15.5	48.3 +0.5	54.4 +4.9	43.2 -3.7	35.1 +5.8
	$v6$	37.5 -6.6	40.1 +0.7	48.5 +0.6	58.7 +6.6	48.4 +0.2	59.5 +5.1	34.7 -8.4	37.3 +2.1
	$v7$	32.3 -5.2	44.0 +3.9	46.0 -2.5	60.9 +2.1	50.7 +2.2	65.5 +6.0	30.9 -3.8	40.9 +3.7
	$v8$	33.5 +1.2	50.0 +6.0	49.1 +3.1	65.8 +4.9	54.6 +3.9	70.4 +4.9	29.3 -1.6	45.2 +4.3
SeqXGPT (Wang et al., 2023)	$v0$	98.8	0.0	100.0	0.0	99.5	0.0	100.0	0.0
	$v1$	81.9 -16.9	2.1 +2.1	83.2 -16.8	0.0 +0.0	82.9 -16.6	1.6 +1.6	81.2 -18.7	0.1 +0.1
	$v2$	73.2 -8.8	0.6 -1.5	73.8 -9.4	0.0 +0.0	73.3 -9.6	0.6 -1.0	71.6 -9.7	0.2 +0.1
	$v3$	58.1 -15.0	0.2 -0.5	59.1 -14.7	0.1 +0.1	58.4 -14.9	0.7 +0.1	57.7 -13.8	0.3 +0.1
	$v4$	48.9 -9.2	0.3 +0.1	50.1 -9.0	0.0 -0.1	48.9 -9.5	0.5 -0.3	48.5 -9.3	0.2 -0.1
	$v5$	38.1 -10.7	0.3 -0.1	39.7 -10.3	0.1 +0.0	38.4 -10.5	0.4 -0.0	37.8 -10.7	0.3 +0.1
	$v6$	23.3 -14.9	0.1 -0.1	25.2 -14.6	0.0 -0.1	23.4 -15.0	0.3 -0.2	20.8 -17.0	0.1 -0.2
	$v7$	8.2 -15.1	0.4 +0.3	10.3 -14.8	0.0 +0.0	8.3 -15.1	0.2 -0.1	9.1 -11.6	0.1 -0.0
	$v8$	0.4 -7.7	0.9 +0.4	2.6 -7.7	0.0 +0.0	0.2 -8.2	0.3 +0.1	0.2 -8.9	0.4 +0.3

Table 10: Sentence-level Main split results for LLM-as-detectors and zero-shot sentence-level detectors across revision versions and domains, aggregated over the three primary generators. We report accuracy and F1-AI. Colored deltas indicate changes relative to the previous version; no delta is shown for v_0.

Detector	Ver.	Essays		Reports		News		Abstracts	
		Macro-F1	FNR	Macro-F1	FNR	Macro-F1	FNR	Macro-F1	FNR
LLM-as-detector									
Claude-Haiku	$v0$	47.7	–	49.1	–	47.9	–	34.7	–
	$v1$	54.3 +6.6	81.2	47.9 -1.2	95.5	49.3 +1.4	91.8	44.4 +9.7	53.8
	$v2$	64.5 +10.2	57.8 -23.4	46.4 -1.4	92.3 -3.2	51.1 +1.8	85.1 -6.6	47.2 +2.8	45.8 -8.0
	$v3$	72.9 +8.4	40.0 -17.8	43.5 -2.9	87.7 -4.6	52.3 +1.2	75.5 -9.6	51.2 +4.1	39.0 -6.8
	$v4$	60.9 -12.0	57.5 +17.5	38.0 -5.5	91.6 +3.9	43.6 -8.7	85.3 +9.8	49.6 -1.7	46.5 +7.5
	$v5$	69.6 +8.7	38.9 -18.6	44.7 +6.8	70.7 -20.9	50.0 +6.5	68.6 -16.7	57.4 +7.8	31.6 -15.0
	$v6$	71.8 +2.1	26.7 -12.2	38.3 -6.4	69.3 -1.5	44.6 -5.4	65.6 -3.0	56.3 -1.0	26.3 -5.3
	$v7$	51.9 -19.9	34.4 +7.6	28.9 -9.4	68.3 -1.0	29.6 -15.0	70.9 +5.4	49.0 -7.4	27.3 +1.0
	$v8$	39.0 -12.8	35.7 +1.3	21.5 -7.4	68.6 +0.3	20.2 -9.4	72.7 +1.7	41.5 -7.4	28.3 +1.1
Gemini-Flash	$v0$	49.8	–	49.9	–	49.9	–	46.3	–
	$v1$	49.9 +0.2	94.6	45.6 -4.3	99.8	46.0 -3.9	99.3	51.1 +4.8	84.6
	$v2$	67.2 +17.3	61.9 -32.7	46.7 +1.1	94.4 -5.4	53.1 +7.1	84.5 -14.8	55.8 +4.7	70.7 -14.0
	$v3$	78.6 +11.3	38.1 -23.8	47.7 +1.0	84.3 -10.0	54.8 +1.7	73.4 -11.1	57.2 +1.5	64.2 -6.5
	$v4$	64.0 -14.6	59.0 +20.9	36.9 -10.8	95.8 +11.5	42.8 -12.0	88.1 +14.7	48.8 -8.4	74.2 +10.1
	$v5$	73.0 +9.0	39.4 -19.6	51.9 +15.0	69.1 -26.7	54.9 +12.1	64.7 -23.4	57.7 +8.9	55.8 -18.4
	$v6$	74.8 +1.7	28.2 -11.2	39.0 -12.9	75.8 +6.8	47.4 -7.5	64.9 +0.2	52.5 -5.2	49.9 -5.9
	$v7$	44.9 -29.9	50.6 +22.4	27.5 -11.4	76.3 +0.4	29.8 -17.6	71.8 +6.9	40.0 -12.6	55.1 +5.2
	$v8$	28.9 -15.9	56.9 +6.2	17.8 -9.7	79.0 +2.8	18.5 -11.2	74.2 +2.5	30.4 -9.5	54.9 -0.2
GPT-5.4	$v0$	49.9	–	49.4	–	50.0	–	49.7	–
	$v1$	47.3 -2.7	97.9	44.6 -4.7	100.0	45.5 -4.5	99.9	45.8 -3.8	98.7
	$v2$	58.1 +10.8	78.3 -19.5	42.9 -1.8	98.8 -1.2	48.9 +3.4	92.1 -7.9	45.6 -0.2	94.9 -3.8
	$v3$	69.0 +10.9	55.8 -22.6	40.2 -2.6	95.7 -3.1	50.7 +1.8	80.4 -11.7	43.4 -2.2	91.3 -3.6
	$v4$	52.4 -16.6	76.5 +20.7	32.6 -7.7	99.7 +4.0	37.2 -13.4	95.4 +15.0	36.2 -7.3	95.8 +4.5
	$v5$	64.6 +12.2	52.9 -23.6	47.4 +14.8	74.0 -25.7	49.2 +11.9	72.1 -23.2	46.9 +10.7	73.6 -22.1
	$v6$	65.9 +1.3	41.4 -11.5	35.3 -12.1	79.6 +5.6	41.3 -7.9	72.9 +0.7	36.8 -10.0	74.6 +0.9
	$v7$	39.1 -26.8	58.8 +17.4	22.4 -12.9	81.3 +1.6	24.6 -16.7	79.0 +6.1	26.0 -10.8	76.9 +2.3
	$v8$	27.0 -12.1	61.6 +2.8	11.8 -10.5	84.5 +3.2	14.3 -10.2	80.5 +1.6	16.5 -9.5	77.5 +0.6
Zero-shot									
AdaLoc (Zhang et al., 2024b)	$v0$	46.0	–	48.6	–	46.4	–	49.2	–
	$v1$	51.9 +5.9	79.3	48.0 -0.6	94.7	50.2 +3.8	85.7	47.3 -1.8	96.4
	$v2$	51.0 -0.8	80.3 +1.0	45.1 -2.9	95.3 +0.5	49.1 -1.1	86.0 +0.3	44.0 -3.3	96.9 +0.4
	$v3$	49.5 -1.5	77.8 -2.6	40.3 -4.9	95.0 -0.3	45.3 -3.8	86.2 +0.2	38.8 -5.2	97.2 +0.3
	$v4$	46.4 -3.1	78.5 +0.7	35.3 -5.0	96.5 +1.5	41.9 -3.3	86.7 +0.5	34.9 -3.9	97.3 +0.1
	$v5$	40.1 -6.3	82.0 +3.6	30.7 -4.6	96.2 -0.3	38.1 -3.9	86.2 -0.5	29.6 -5.3	97.4 +0.1
	$v6$	34.0 -6.1	79.9 -2.1	22.5 -8.2	96.1 -0.1	29.9 -8.1	86.4 +0.2	19.7 -9.9	97.3 -0.1
	$v7$	26.4 -7.7	75.8 -4.2	11.5 -11.0	95.7 -0.4	20.8 -9.2	84.5 -1.9	11.9 -7.8	96.3 -1.0
	$v8$	22.4 -4.0	70.2 -5.5	4.7 -6.7	95.0 -0.7	14.4 -6.4	83.1 -1.4	5.3 -6.6	94.4 -1.9
GenAI-Sentence (Teja et al., 2025)	$v0$	34.1	–	17.1	–	23.9	–	20.0	–
	$v1$	45.4 +11.3	42.4	31.0 +13.9	18.3	37.0 +13.1	28.8	33.9 +13.9	19.2
	$v2$	49.6 +4.2	38.1 -4.3	36.7 +5.7	19.8 +1.6	41.6 +4.6	31.0 +2.2	38.6 +4.7	18.2 -1.0
	$v3$	54.2 +4.6	31.8 -6.2	43.1 +6.5	18.6 -1.3	46.2 +4.7	31.6 +0.6	44.1 +5.5	14.0 -4.3
	$v4$	54.4 +0.3	39.3 +7.4	43.5 +0.4	17.6 -1.0	47.1 +0.8	31.1 -0.5	47.2 +3.1	19.9 +6.0
	$v5$	56.3 +1.8	30.1 -9.1	49.1 +5.5	22.0 +4.4	50.0 +2.9	33.1 +2.0	50.3 +3.1	11.4 -8.5
	$v6$	55.4 -0.8	25.8 -4.3	50.6 +1.5	20.8 -1.2	49.8 -0.1	32.5 -0.6	52.5 +2.2	9.4 -2.0
	$v7$	49.4 -6.0	27.7 +1.9	48.5 -2.0	21.2 +0.4	45.5 -4.3	32.6 +0.2	51.8 -0.6	10.2 +0.7
	$v8$	43.3 -6.1	23.6 -4.1	44.5 -4.1	19.9 -1.3	40.7 -4.8	31.2 -1.4	47.9 -3.9	8.0 -2.1
GL-CLiC (Adi et al., 2025)	$v0$	40.3	–	33.3	–	29.7	–	42.6	–
	$v1$	51.6 +11.2	56.5	44.8 +11.5	46.8	42.0 +12.2	37.7	52.3 +9.7	67.1
	$v2$	50.3 -1.2	66.3 +9.8	45.6 +0.8	54.8 +8.1	44.6 +2.6	44.2 +6.5	50.2 -2.1	73.3 +6.2
	$v3$	50.0 -0.4	65.7 -0.5	50.5 +4.9	49.0 -5.8	48.6 +4.0	43.4 -0.8	49.2 -1.0	72.2 -1.1
	$v4$	46.8 -3.1	69.8 +4.1	41.7 -8.7	67.3 +18.3	47.7 -0.9	49.6 +6.2	43.4 -5.8	78.6 +6.4
	$v5$	43.8 -3.1	70.6 +0.9	47.5 +5.7	53.9 -13.4	47.2 -0.4	49.6 -0.0	42.3 -1.1	75.2 -3.3
	$v6$	37.2 -6.6	72.3 +1.6	45.1 -2.3	52.3 -1.6	43.9 -3.4	50.1 +0.4	34.6 -7.7	75.5 +0.3
	$v7$	29.1 -8.1	71.0 -1.3	36.7 -8.4	54.2 +2.0	39.0 -4.8	48.6 -1.5	28.7 -5.8	73.6 -1.9
	$v8$	25.0 -4.1	66.5 -4.4	32.9 -3.8	50.9 -3.3	35.2 -3.8	45.4 -3.2	22.6 -6.1	70.7 -2.9
SeqXGPT (Wang et al., 2023)	$v0$	49.7	–	50.0	–	49.9	–	50.0	–
	$v1$	46.1 -3.6	98.9	45.4 -4.6	100.0	46.1 -3.8	99.2	44.9 -5.1	100.0
	$v2$	42.6 -3.5	99.7 +0.8	42.5 -3.0	100.0 +0.0	42.6 -3.5	99.7 +0.5	41.8 -3.0	99.9 -0.1
	$v3$	36.8 -5.7	99.9 +0.2	37.2 -5.3	99.9 -0.1	37.2 -5.4	99.6 -0.1	36.8 -5.0	99.8 -0.1
	$v4$	33.0 -3.9	99.8 -0.1	33.4 -3.8	100.0 +0.1	33.1 -4.1	99.8 +0.1	32.7 -4.0	99.9 +0.1
	$v5$	27.7 -5.3	99.9 +0.0	28.4 -4.9	100.0 -0.0	27.9 -5.1	99.8 +0.0	27.5 -5.2	99.9 -0.0
	$v6$	18.9 -8.8	99.9 +0.1	20.1 -8.4	100.0 +0.0	19.1 -8.9	99.9 +0.1	17.2 -10.3	100.0 +0.1
	$v7$	7.6 -11.3	99.8 -0.2	9.3 -10.8	100.0 +0.0	7.7 -11.3	99.9 +0.0	8.4 -8.8	100.0 +0.0
	$v8$	0.4 -7.2	99.6 -0.2	2.4 -6.8	100.0 -0.0	0.1 -7.6	99.8 -0.1	0.2 -8.2	99.8 -0.2

Table 11: Extended sentence-level Main split results for LLM-as-detectors and zero-shot sentence-level detectors across revision versions and domains, aggregated over the three primary generators. We report Macro-F1 and FNR. Colored deltas indicate changes relative to the previous version; no delta is shown for v_0.

Detector	Ver.	Essays		Reports		News		Abstracts	
		Acc	F1-AI	Acc	F1-AI	Acc	F1-AI	Acc	F1-AI
Fine-grained Zero-shot (Token + Span)									
GigaCheck (Tolstykh et al., 2024)	$v0$	98.7	0.0	98.5	0.0	99.5	0.0	97.5	0.0
	$v1$	82.6 -16.1	7.6 +7.6	81.8 -16.8	2.6 +2.6	82.9 -16.6	1.5 +1.5	79.6 -17.9	6.8 +6.8
	$v2$	74.2 -8.4	19.0 +11.4	71.9 -9.9	3.3 +0.6	72.6 -10.4	3.0 +1.5	69.4 -10.1	12.1 +5.3
	$v3$	66.1 -8.1	39.0 +20.0	56.6 -15.3	5.7 +2.5	57.2 -15.3	8.2 +5.2	58.5 -11.0	22.4 +10.3
	$v4$	58.9 -7.2	26.6 -12.4	52.6 -4.0	8.7 +3.0	51.9 -5.3	6.8 -1.4	53.3 -5.2	21.4 -1.0
	$v5$	54.6 -4.3	44.8 +18.3	38.6 -14.1	11.5 +2.7	38.7 -13.3	12.1 +5.3	45.9 -7.4	29.9 +8.5
	$v6$	56.7 +2.1	62.3 +17.4	28.9 -9.7	18.3 +6.8	30.1 -8.6	20.7 +8.7	43.9 -2.0	46.5 +16.6
	$v7$	41.6 -15.1	53.8 -8.4	17.0 -11.9	17.7 -0.6	19.0 -11.1	20.0 -0.7	32.8 -11.1	40.5 -5.9
	$v8$	51.1 +9.4	66.7 +12.8	11.2 -5.8	18.1 +0.4	15.7 -3.3	23.9 +3.8	30.9 -1.9	45.2 +4.7
DAMASHA (Teja et al., 2026)	$v0$	89.1	0.0	82.9	0.0	98.5	0.0	80.6	0.0
	$v1$	76.6 -12.5	22.3 +22.3	71.5 -11.4	19.0 +19.0	82.2 -16.3	4.2 +4.2	69.4 -11.2	23.4 +23.4
	$v2$	71.7 -4.9	31.8 +9.5	65.8 -5.8	23.1 +4.1	73.3 -9.0	7.6 +3.4	64.6 -4.7	30.9 +7.5
	$v3$	65.5 -6.2	46.7 +14.9	56.5 -9.3	31.7 +8.6	59.5 -13.7	12.6 +5.0	60.3 -4.3	40.8 +9.9
	$v4$	60.5 -5.0	37.9 -8.8	53.0 -3.4	27.3 -4.4	53.6 -5.9	6.7 -5.9	56.0 -4.3	36.7 -4.1
	$v5$	54.9 -5.6	51.7 +13.8	44.9 -8.1	35.0 +7.7	40.0 -13.6	13.3 +6.5	51.1 -4.9	46.4 +9.7
	$v6$	53.9 -0.9	61.1 +9.4	38.4 -6.5	40.0 +5.0	28.1 -12.0	15.9 +2.7	46.2 -4.9	51.7 +5.4
	$v7$	41.9 -12.1	55.1 -6.1	26.8 -11.6	35.5 -4.5	13.4 -14.7	11.4 -4.6	38.1 -8.1	49.4 -2.3
	$v8$	41.3 -0.6	58.1 +3.0	24.6 -2.2	39.2 +3.7	6.3 -7.1	11.5 +0.2	37.2 -1.0	53.6 +4.2

Table 12: Fine-grained Main split results across revision versions and domains, aggregated over the three primary generators. DAMASHA is evaluated at the token level and GigaCheck at the span level. We report accuracy and F1-AI. Colored deltas indicate changes relative to the previous version; no delta is shown for v_0.

Detector	Ver.	Essays		Reports		News		Abstracts	
		Macro-F1	FNR	Macro-F1	FNR	Macro-F1	FNR	Macro-F1	FNR
Fine-grained Zero-shot (Token + Span)									
GigaCheck (Tolstykh et al., 2024)	$v0$	49.7	–	49.6	–	49.9	–	49.4	–
	$v1$	49.0 -0.7	95.7	46.3 -3.4	98.6	46.1 -3.8	99.2	47.7 -1.7	96.0
	$v2$	51.8 +2.8	87.9 -7.8	43.4 -2.9	98.3 -0.3	43.5 -2.6	98.4 -0.8	46.8 -0.9	92.5 -3.5
	$v3$	57.5 +5.7	71.3 -16.7	38.8 -4.6	96.9 -1.3	40.1 -3.4	95.2 -3.2	46.8 +0.1	84.7 -7.7
	$v4$	48.9 -8.6	82.8 +11.5	38.4 -0.4	95.2 -1.8	37.2 -2.9	96.2 +1.0	44.0 -2.8	86.1 +1.4
	$v5$	51.6 +2.7	65.6 -17.2	32.0 -6.4	93.4 -1.7	32.1 -5.0	92.9 -3.3	41.6 -2.4	78.7 -7.4
	$v6$	53.6 +2.0	50.5 -15.1	26.8 -5.1	88.5 -4.9	28.1 -4.1	86.7 -6.3	40.6 -1.0	64.7 -14.0
	$v7$	35.2 -18.4	61.3 +10.8	16.3 -10.6	89.5 +1.0	17.7 -10.4	87.5 +0.8	29.1 -11.6	72.1 +7.4
	$v8$	34.3 -0.9	48.9 -12.4	9.6 -6.7	89.2 -0.3	12.7 -5.0	84.8 -2.7	23.2 -5.9	69.4 -2.7
DAMASHA (Teja et al., 2026)	$v0$	47.1	–	45.3	–	49.6	–	44.6	–
	$v1$	54.3 +7.1	79.4	50.9 +5.6	80.4	47.2 -2.4	97.7	52.1 +7.5	73.5
	$v2$	57.0 +2.7	73.4 -5.9	50.6 -0.3	80.9 +0.4	46.0 -1.2	95.7 -1.9	53.5 +1.4	70.8 -2.7
	$v3$	60.5 +3.6	61.3 -12.1	49.8 -0.7	76.1 -4.8	43.1 -2.8	92.7 -3.1	55.3 +1.8	65.4 -5.4
	$v4$	54.5 -6.1	72.1 +10.9	46.3 -3.5	81.0 +4.9	37.9 -5.2	96.4 +3.7	51.4 -3.9	71.4 +6.0
	$v5$	54.3 -0.2	60.5 -11.7	43.4 -2.9	76.4 -4.6	33.5 -4.4	92.4 -4.0	50.1 -1.3	64.9 -6.5
	$v6$	51.8 -2.5	52.3 -8.2	38.1 -5.3	73.3 -3.1	26.3 -7.2	90.8 -1.6	44.5 -5.6	62.0 -2.9
	$v7$	36.0 -15.8	61.0 +8.8	25.0 -13.1	77.7 +4.4	13.2 -13.1	93.8 +3.0	33.7 -10.8	65.7 +3.7
	$v8$	29.1 -6.9	58.7 -2.3	19.6 -5.4	75.4 -2.3	5.8 -7.4	93.7 -0.1	26.8 -6.9	62.8 -2.8

Table 13: Extended fine-grained Main split results across revision versions and domains, aggregated over the three primary generators. DAMASHA is evaluated at the token level and GigaCheck at the span level. We report Macro-F1 and FNR. Colored deltas indicate changes relative to the previous version; no delta is shown for v_0.

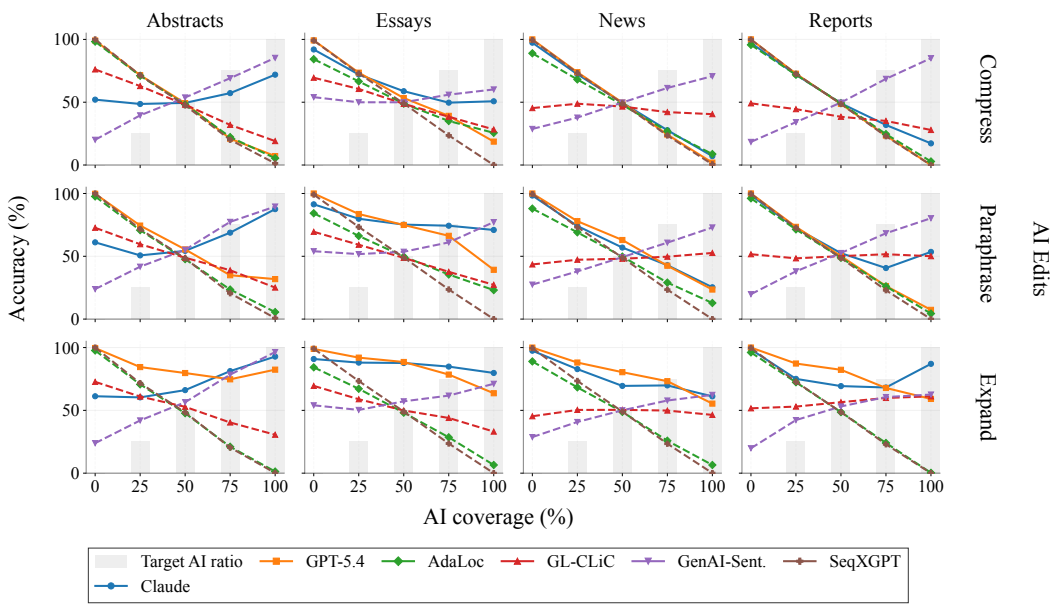

Figure 12: Sentence-level accuracy under coverage-controlled edit operations.

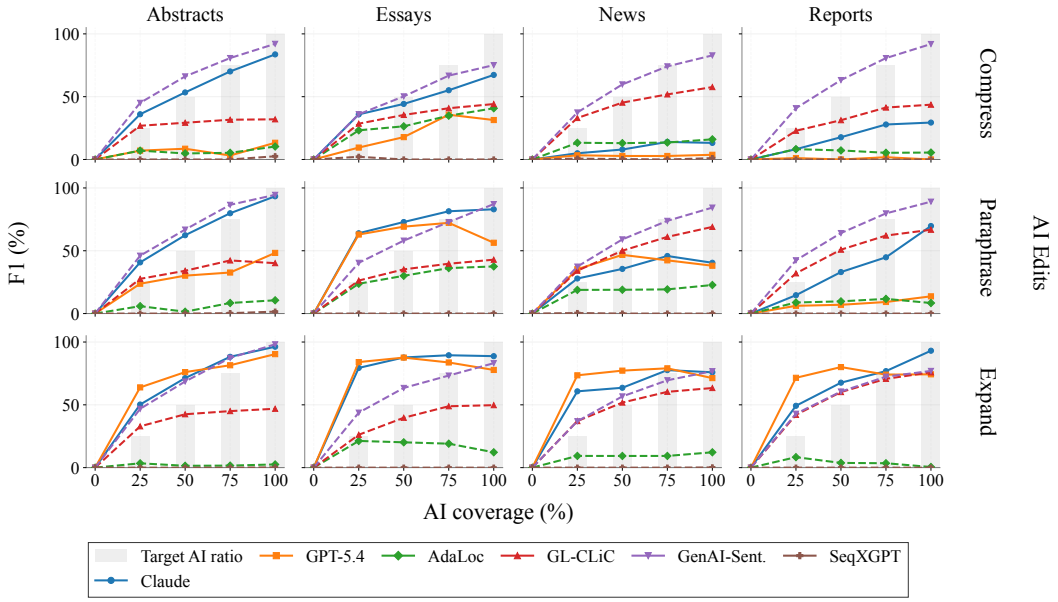

Figure 13: Sentence-level F1-AI under coverage-controlled edit operations.

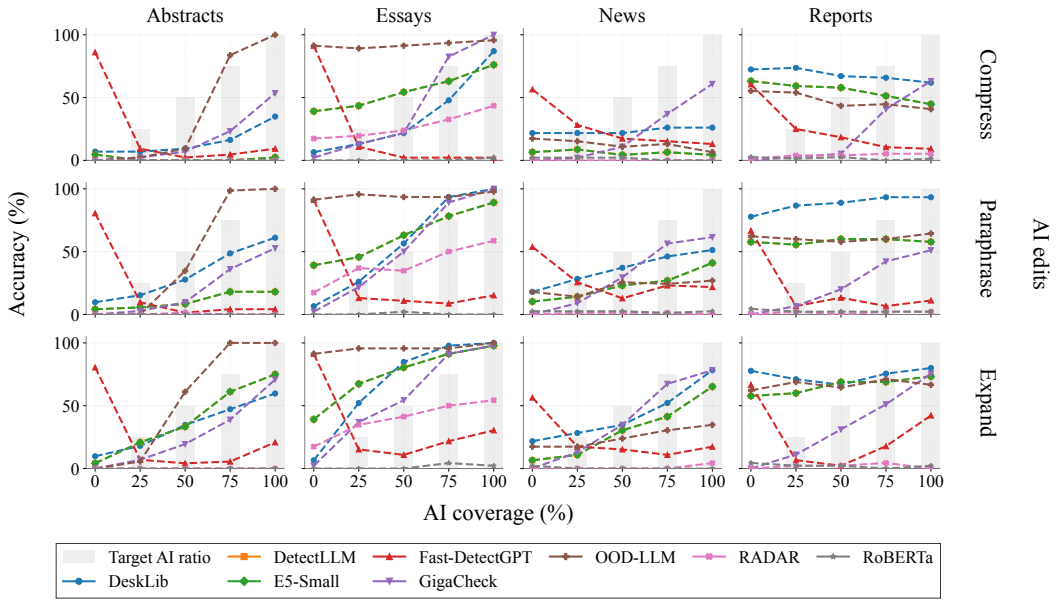

Figure 14: Document-level accuracy under coverage-controlled edit operations.

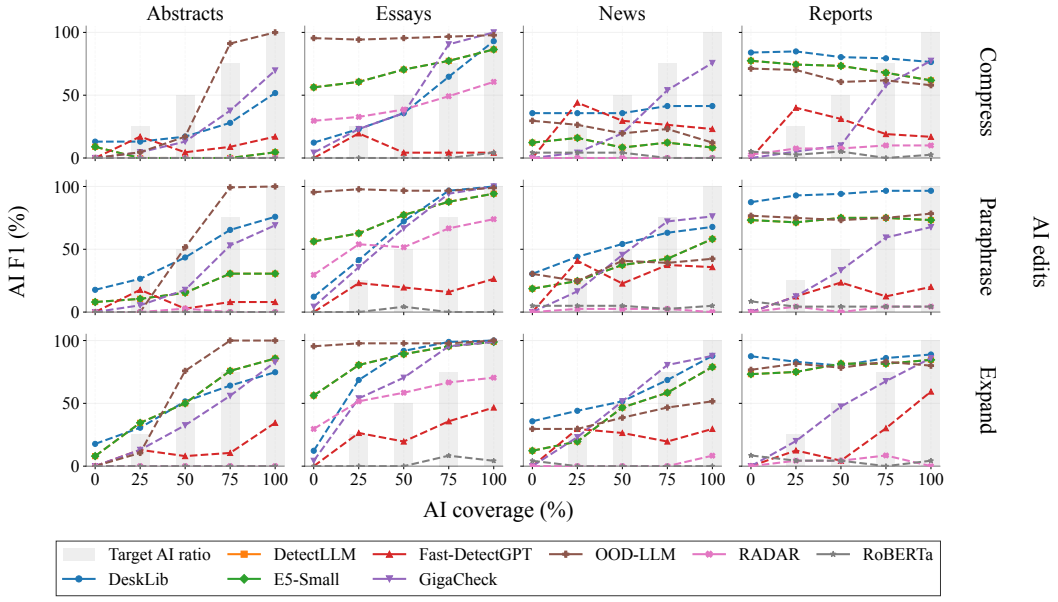

Figure 15: Document-level F1-AI under coverage-controlled edit operations.

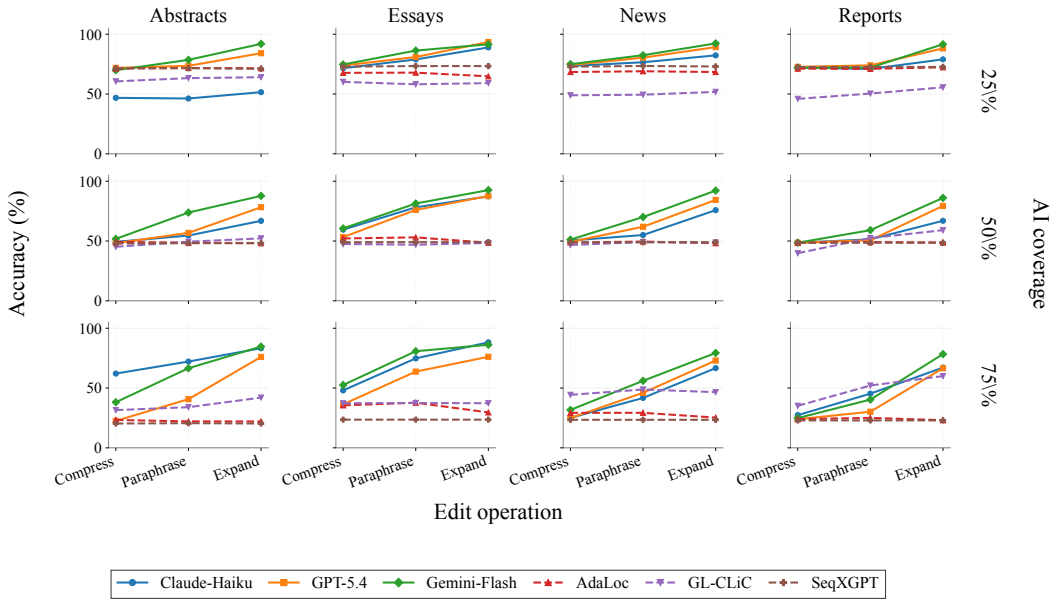

Figure 16: Sentence-level accuracy at fixed AI coverage while varying edit operation.

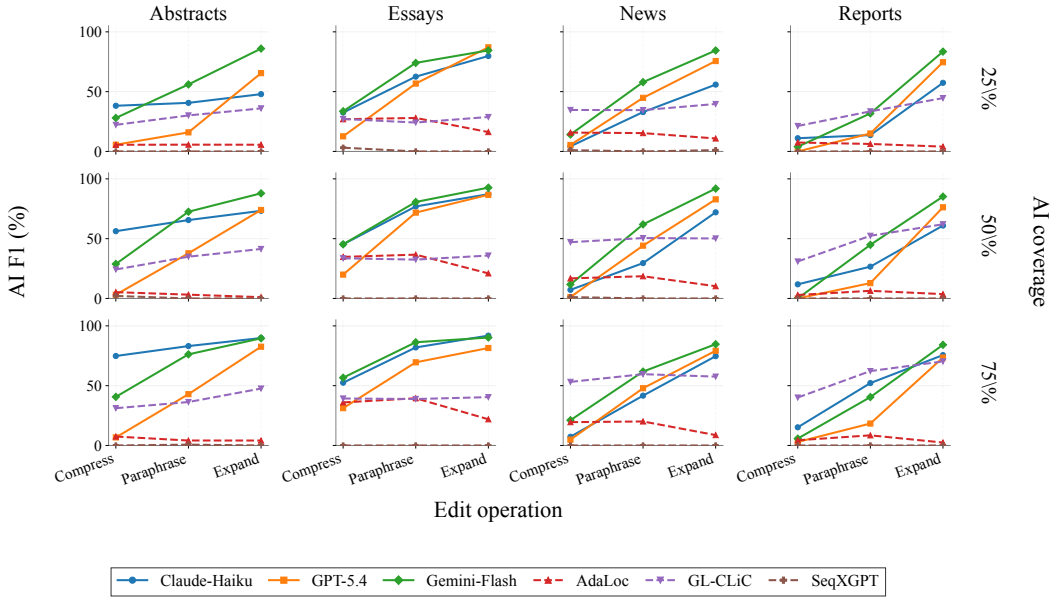

Figure 17: Sentence-level F1-AI at fixed AI coverage while varying edit operation.

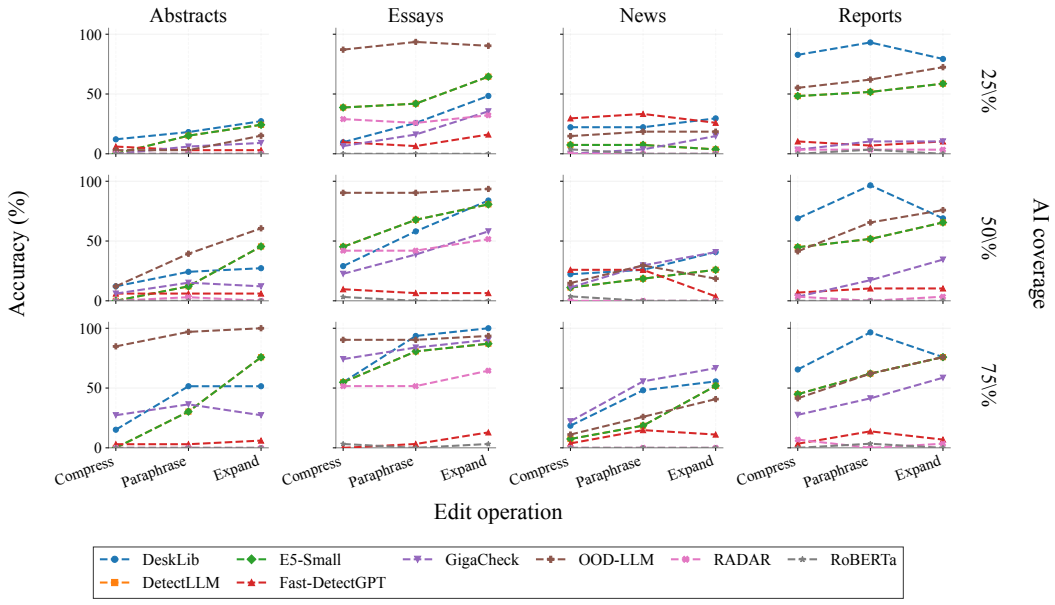

Figure 18: Document-level accuracy at fixed AI coverage while varying edit operation.

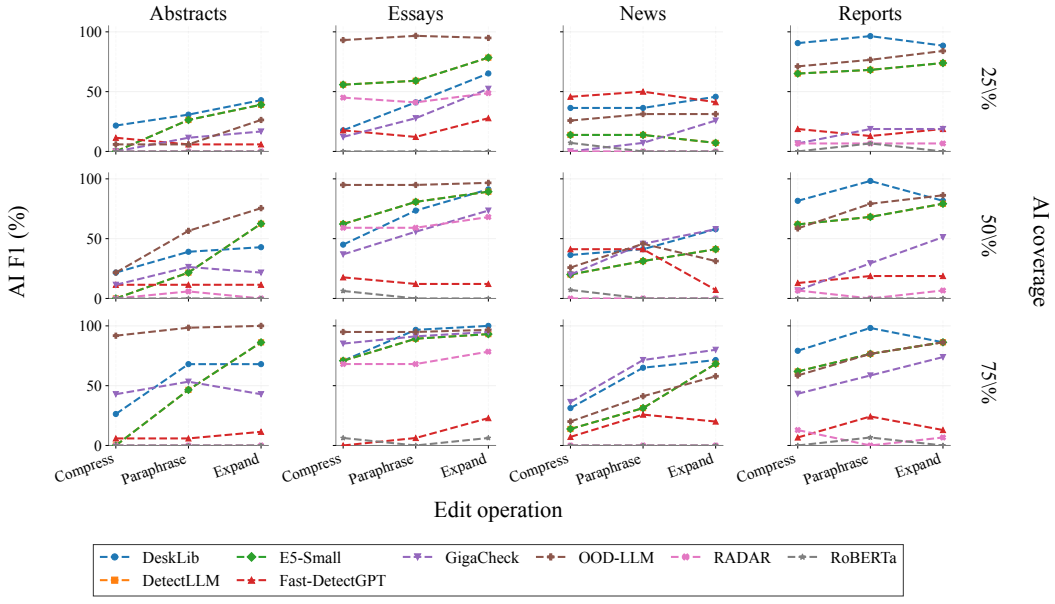

Figure 19: Document-level F1-AI at fixed AI coverage while varying edit operation.

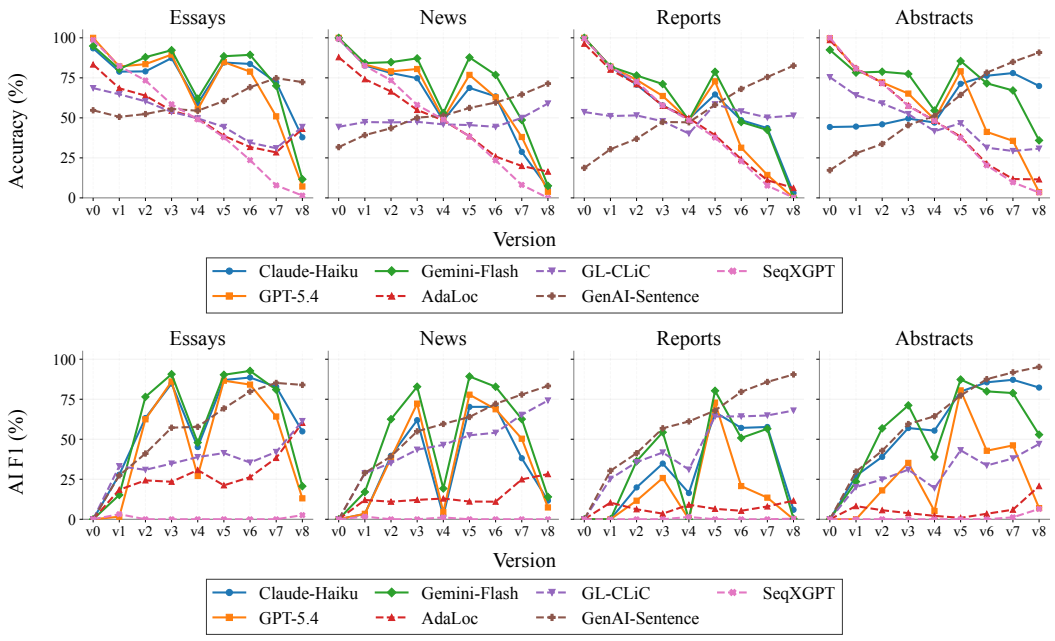

Figure 20: Sentence-level performance under independent edits from the source document. Top: accuracy; bottom: F1-AI. Each point corresponds to a version (v_0–v_8). Compared to the cumulative trajectory, trends are smoother and the drop around v_4 is less pronounced, suggesting that the non-monotonic behavior is more strongly associated with edit type than with accumulated edits.

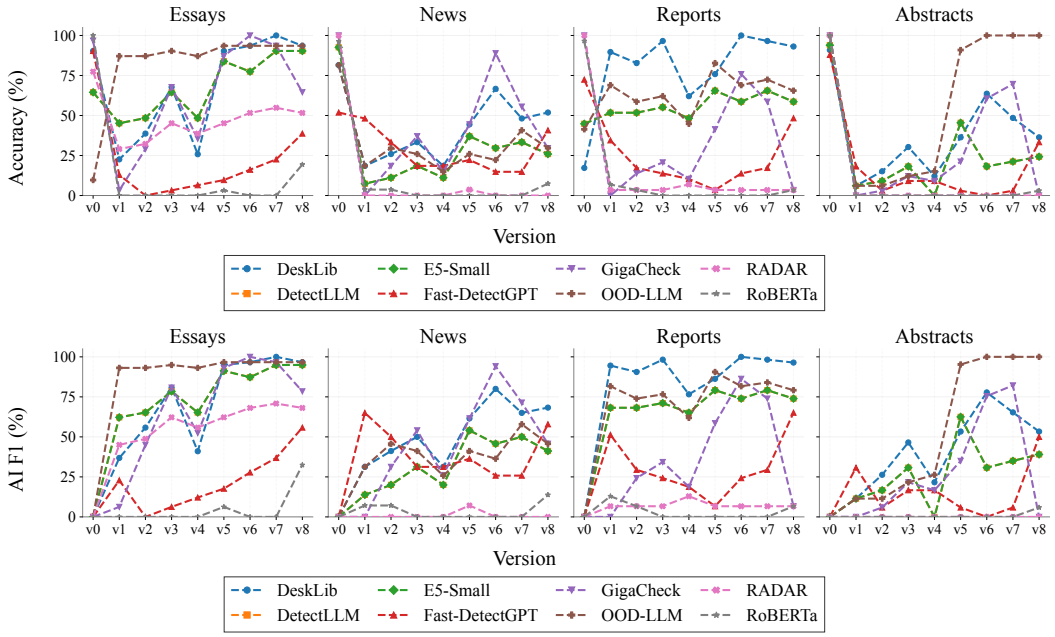

Figure 21: Document-level performance under independent edits from the source document. Top: accuracy; bottom: F1-AI. Each point corresponds to a version (v_0–v_8). Compared to the cumulative trajectory, trends are smoother and the drop around v_4 is less pronounced, suggesting that the non-monotonic behavior is more strongly associated with edit type than with accumulated edits.

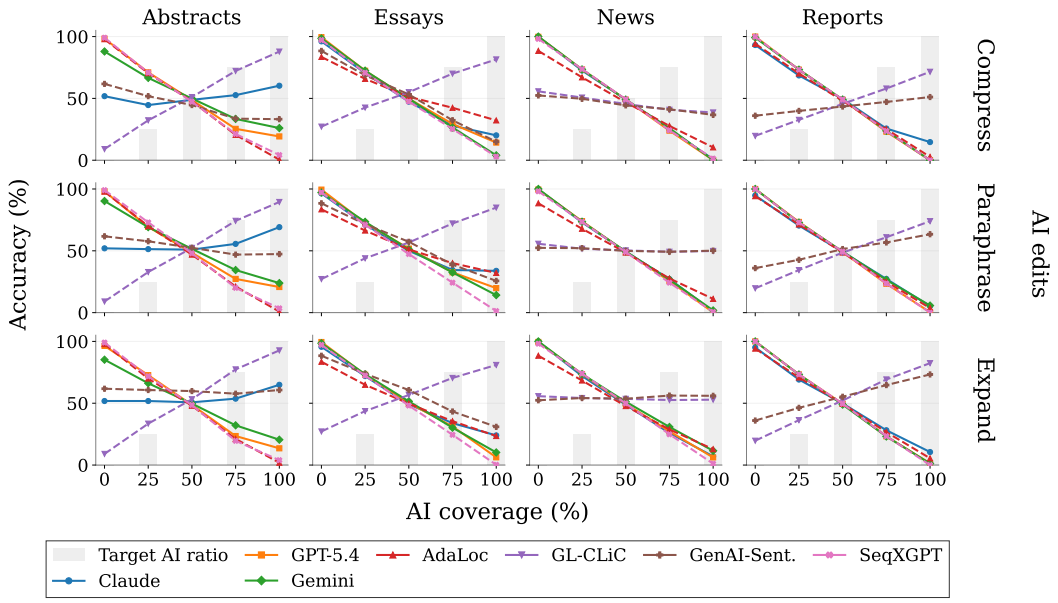

Figure 22: Sentence-level accuracy under coverage-controlled edit operations using *GPT-5.4-nano*.

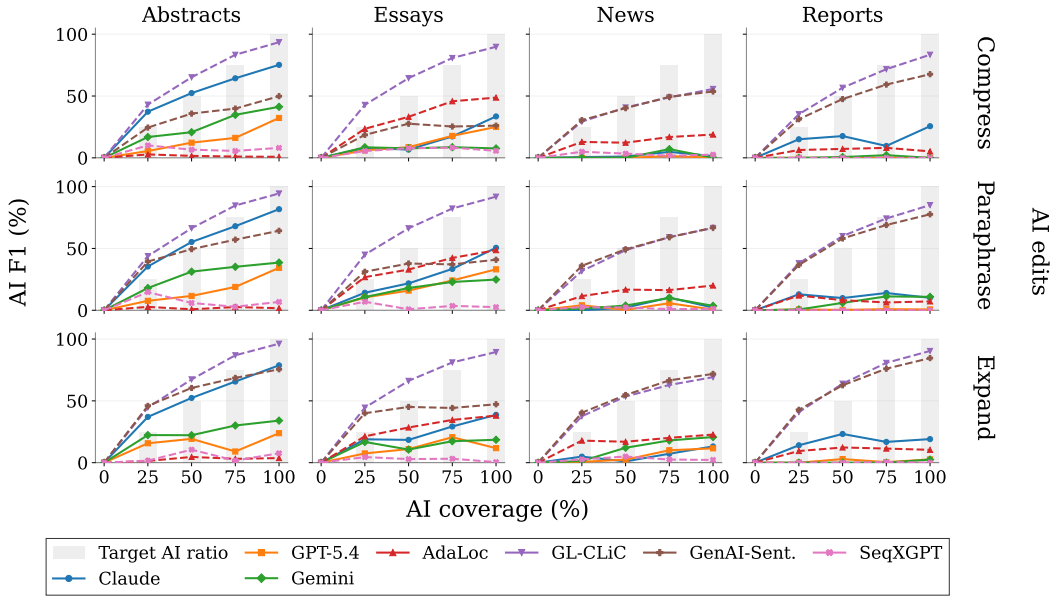

Figure 23: Sentence-level F1-AI under coverage-controlled edit operations using *GPT-5.4-nano*.

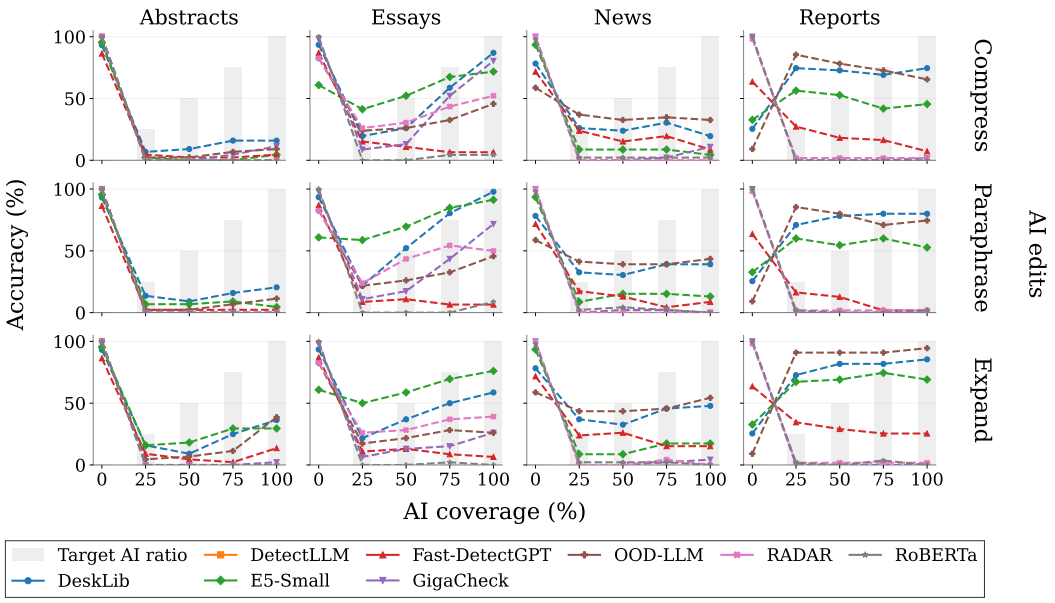

Figure 24: Document-level accuracy under coverage-controlled edit operations using *GPT-5.4-nano*.

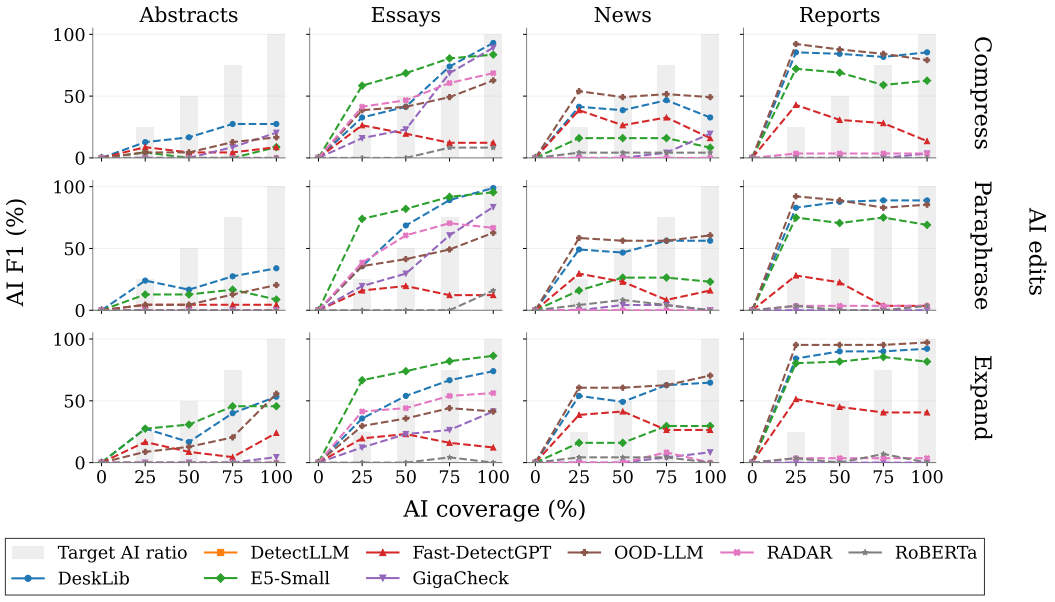

Figure 25: Document-level F1-AI under coverage-controlled edit operations using *GPT-5.4-nano*.

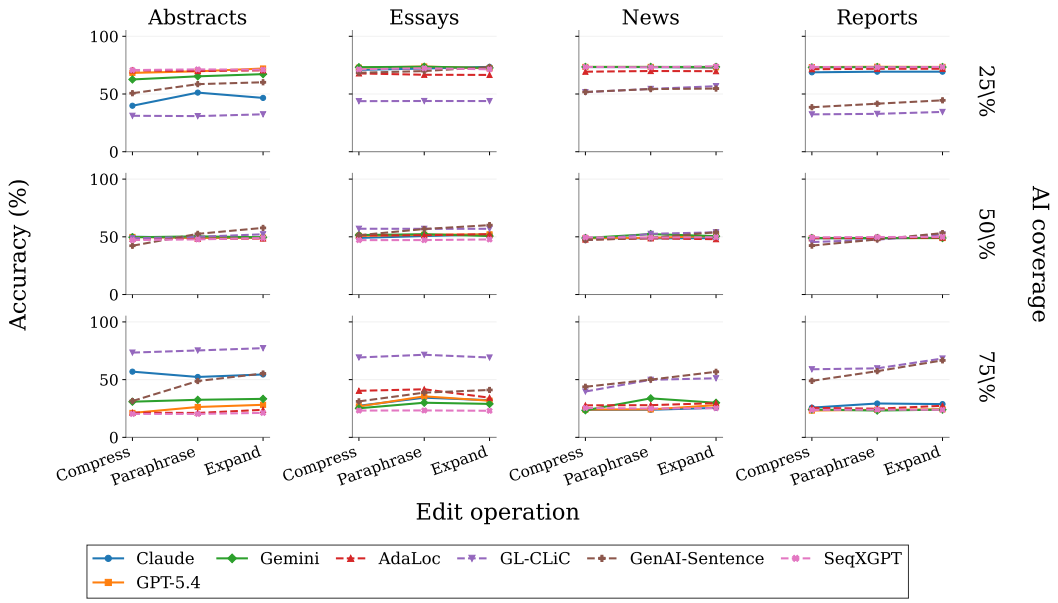

Figure 26: Sentence-level accuracy at fixed AI coverage while varying edit operation using *GPT-5.4-nano*.

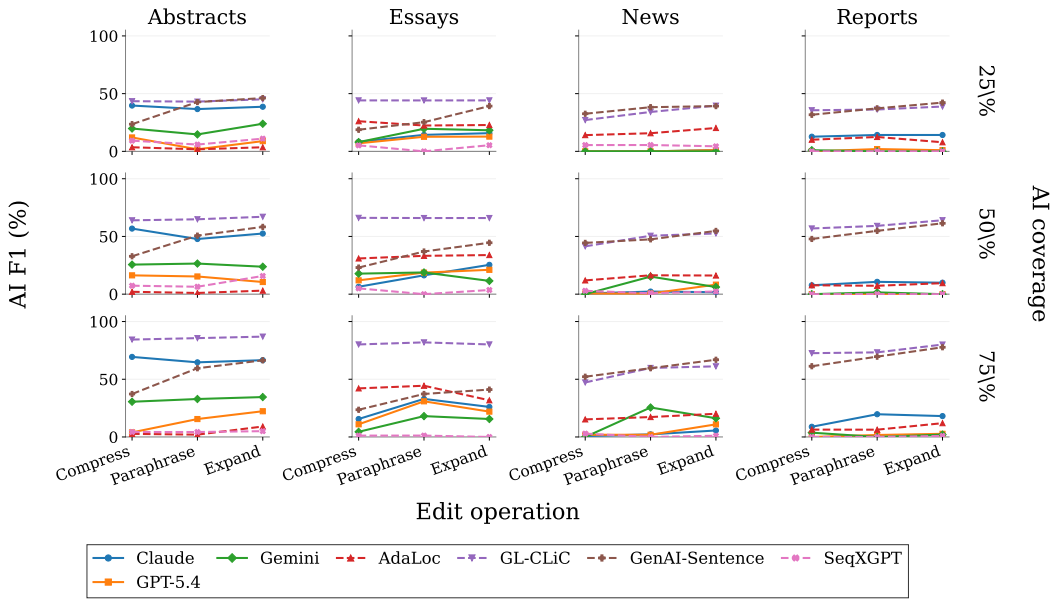

Figure 27: Sentence-level F1-AI at fixed AI coverage while varying edit operation using *GPT-5.4-nano*.

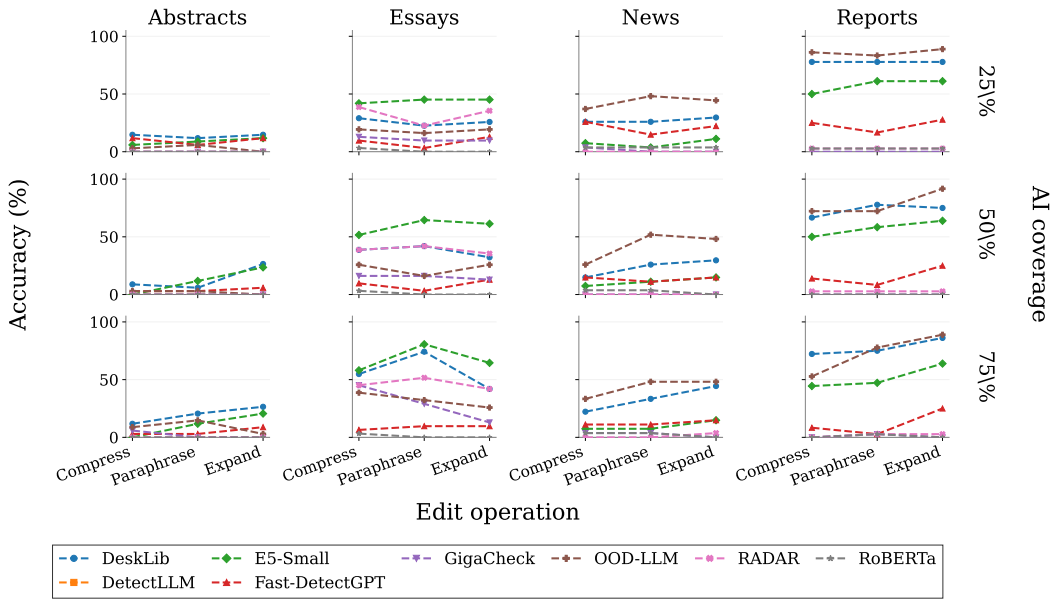

Figure 28: Document-level accuracy at fixed AI coverage while varying edit operation using *GPT-5.4-nano*.

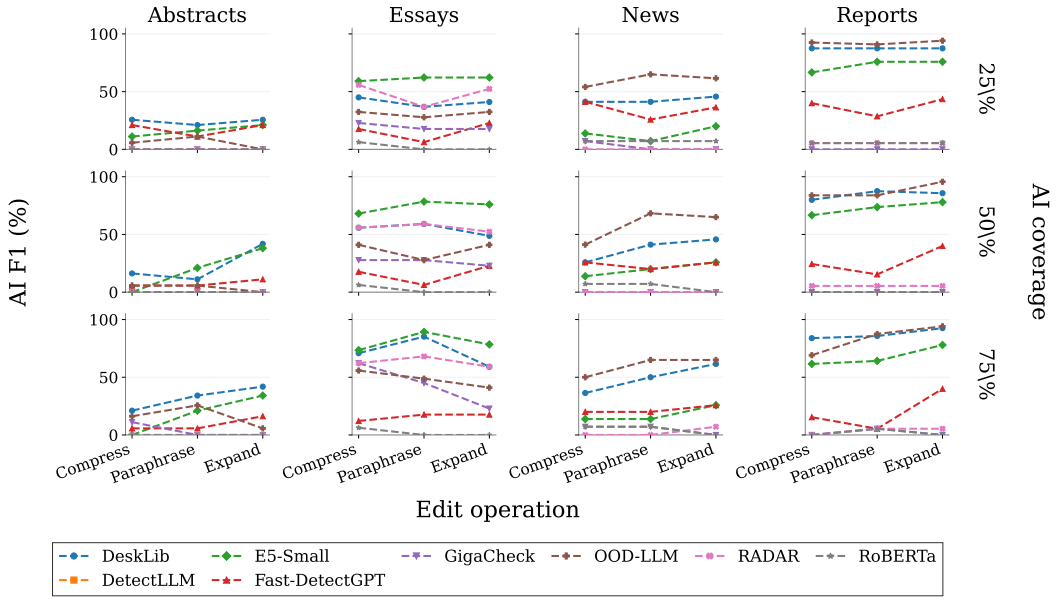

Figure 29: Document-level F1-AI at fixed AI coverage while varying edit operation using *GPT-5.4-nano*.

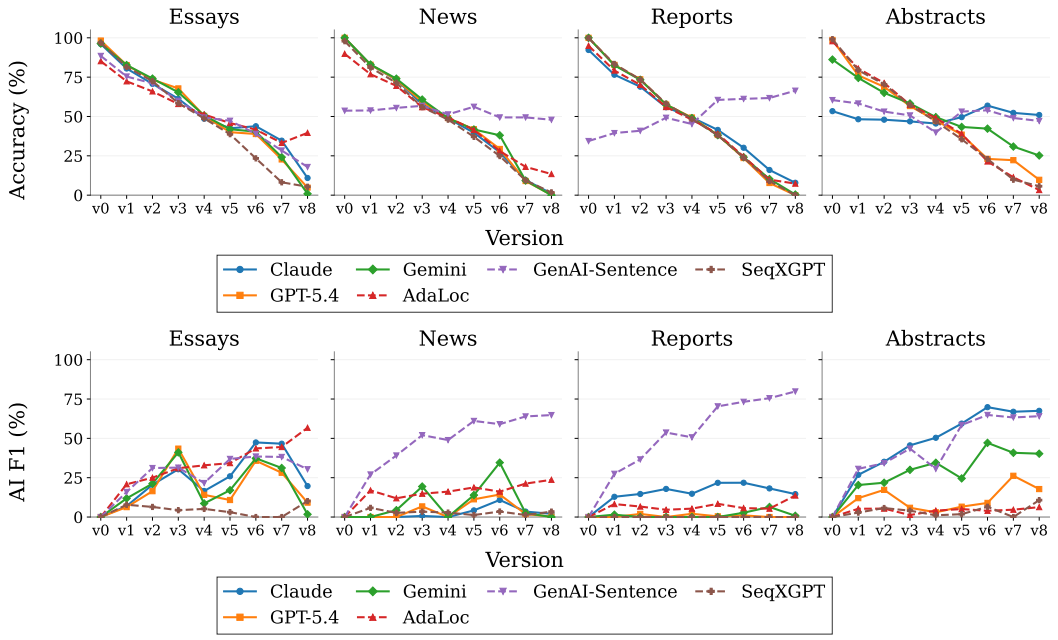

Figure 30: Sentence-level performance under independent edits from the source document using *GPT-5.4-nano*. Top: accuracy; bottom: F1-AI. Each point corresponds to a version (`v0`–`v8`).

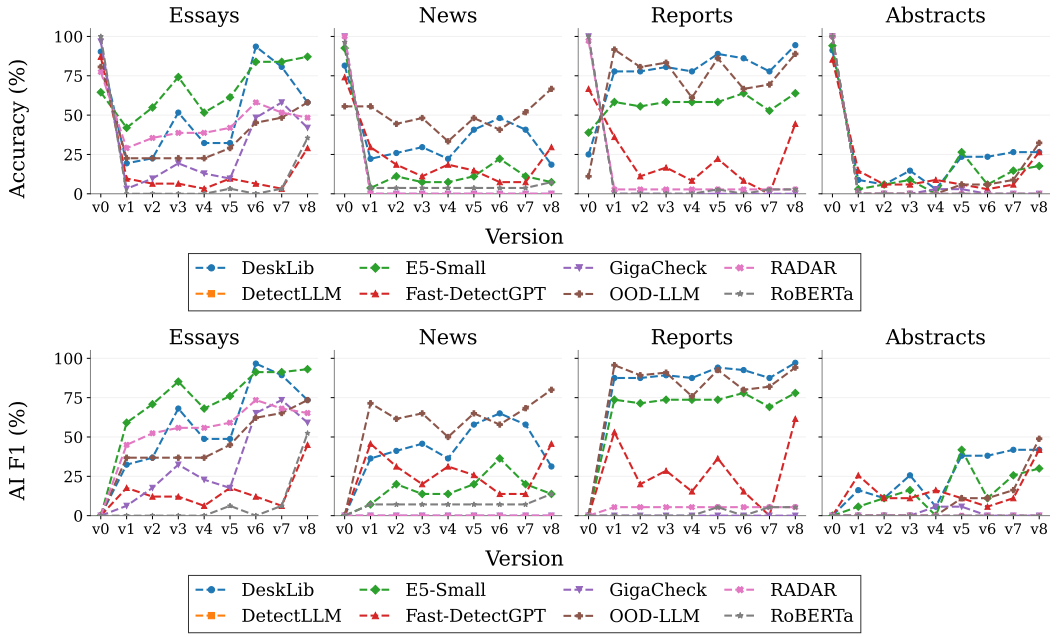

Figure 31: Document-level performance under independent edits from the source document using *GPT-5.4-nano*. Top: accuracy; bottom: F1-AI. Each point corresponds to a version (`v0`–`v8`).