DON'T DISCARD, REPAIR: A REPAIR-FIRST PARADIGM FOR RECOVERING TRAINING UTILITY FROM IMPERFECT PRE-TRAINING CORPORA

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) pre-training increasingly relies on large-scale corpora, yet existing data curation pipelines predominantly follow a filter-and-discard paradigm: documents failing quality criteria are removed rather than assessed for recoverability. We show that this paradigm conflates structural corruption (e.g., encoding, parsing, and layout errors) with low semantic quality, leading to the discard of potentially useful training data—a limitation that is especially consequential in data-scarce settings where such data may be unique and irreplaceable.

We introduce REPAIR-FIRST, a data curation paradigm for recovering training utility from imperfect pre-training corpora through reversible repair before irreversible filtering. REPAIR-FIRST makes three contributions: (1) it establishes structural integrity as a missing dimension of LLM pre-training data quality and introduces repair-before-filtering as a new curation paradigm; (2) it develops a reversible, mixed-content-aware repair framework whose bounded, logged edits restore structurally corrupted corpora without generative rewriting; and (3) it systematically demonstrates that structural repair improves LLM pre-training utility and exposes limitations of existing quality assessment methods.

A controlled perturbation benchmark and fixed-budget experiments with Qwen3-1.7B across multiple Nemotron-CC quality regimes reveal three findings: (1) structural corruption constitutes a distinct failure mode in LLM data curation, with existing quality estimators exhibiting systematic blind spots, especially for layout-level anomalies; (2) reversible repair recovers measurable training utility from imperfect corpora; and (3) REPAIR-FIRST improves corpus utilization across diverse quality regimes while complementing existing filtering pipelines.

Our findings suggest that future LLM data curation should not only evaluate whether data is high quality, but also assess whether apparently low-quality data contains recoverable training value before irreversible deletion.

1 INTRODUCTION

Web-scale pre-training corpora are built by filtering crawls: a cascade of largely binary, mostly document-level decisions (URL, language, heuristic rules, deduplication, model-based quality) keeps, deletes or buckets documents, at most deleting lines or duplicate paragraphs (Wenzek et al., 2020; Penedo et al., 2024; Li et al., 2024; Su et al., 2025). The cuts of this FILTER-AND-DISCARD paradigm are deep—DCLM keeps only the top 10% of the documents that survive its heuristic filters and deduplication (Li et al., 2024)—and irreversible. Yet discarded text is not worthless: when Nguyen et al. (2025) rewrite web documents with a 70B model, about 82% of the rewritten documents in their best training mix derive from originals that a standard quality filter would discard.

Consider a technical page whose prose is hard-wrapped at a fixed column, whose apostrophes arrive as mojibake, and which embeds a Python program. A filter that judges its surface sees noise; a cleaner that collapses whitespace makes the prose readable and destroys the program's indentation. Repairing such a page is not a string-processing chore but three coupled decisions—*where* an edit may act, *which* correction is plausible given the corruption that produced the text, and *when* a repair may be committed—each of which needs a model. We argue that much of what filters discard is

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.

of this kind (§5, §6.2): *structurally damaged* rather than semantically poor. Such defects sit in the physical representation of a document, leave machine-detectable diagnostic signals and are in principle reversible; filtering conflates such *structural corruption* with low *semantic quality*. To our knowledge no mainstream open pipeline performs non-generative, reversible repair of structural defects before filtering, and generative rewriting (Maini et al., 2024; Nguyen et al., 2025; Yu & Xiong, 2026; Maini et al., 2025; Su et al., 2025; Karimi Mahabadi et al., 2026) regenerates the text, so the source text itself is not among its outputs even where faithfulness is rewarded, as in RePro.

When data are scarce—unique, irreplaceable or expensive to recreate—keeping rather than discarding needs no experiment: a discarded document is lost (§2). Whether *repairing* what is kept adds training value does need evidence. We measure it directly, as the difference between training on the raw and on the repaired text of the same documents at a fixed budget, on Common Crawl's 2013–2015 snapshots, which we treat as intermediate between the scarce and the plentiful regime: 2013 opens both Common Crawl's WARC-era series and Nemotron-CC's coverage, whose quality tiers let us condition on a state-of-the-art filter, yet every 2013–2015 tier exceeds our token budget (§5). **RQ1** (a distinct failure mode; H1): do existing quality estimators detect structural corruption, or do they have systematic blind spots? (§4) **RQ2** (training utility; H2): at a fixed token budget, does pre-training on repaired documents improve the model, and which repair layer is responsible? (§6.1) **RQ3** (corpus utilization; H3): how much of each quality regime does repair reach, and how do its effects vary from high- to low-quality data? (§6.2–7)

Contributions. (1) *Formulation.* We add *repair* to the actions of curation, model structural damage as a corruption channel whose recoverable part can be inverted from local evidence, derive an abstention rule that separates repair from generative rewriting, and state constraints and three testable hypotheses (§2). (2) *Measurement.* The quality estimators used for selection are largely insensitive to, or inverted by, many structural corruptions, layout above all: the good/bad KenLM ensemble prefers the corrupted text on 60.9–70.5% of pairs for four layout defects. Two consequences follow: such scorers cannot validate repair, and damage survives selection into every tier (§4). (3) *Principles and algorithms.* Two principles, evidence ordering and containment, derive a ten-stage topology and a mixed-content router with exact restoration (Proposition 1); recoverable boundary and linguistic edits and a validator chosen by sensitivity to the defects repair fixes complete the pipeline (§3, Table 1). Code, mathematics and structured data are protected and restored verbatim; only prose is edited. (4) *Evidence.* In token-matched pre-training of a 1.7B model from scratch on three Nemotron-CC tiers, two of which a top-10% classifier filter would discard entirely, the full pipeline wins 31 robust comparisons against 14 and improves LAMBADA by 3.9–5.3 pp and web-domain BPB by 4.4–6.6% on every tier; its effects on the six improved web and mixed-source corpora agree across the three disjoint tiers to within 0.34 points, and its downstream gains concentrate on tasks that read multi-sentence passages (§6–7, Fig. 1).

Five findings. (1) Selection scorers are largely blind to layout damage (H1), so it survives into every quality tier, HQ included: repair touches 72.6% of HQ documents (§4, §6.2). (2) The full pipeline is net-positive on every tier: 31 robust wins against 14 losses, LAMBADA $+3.9$ to $+5.3$ pp, web-domain BPB -4.4 to -6.6% (§6.1). (3) Its effects are consistent across three disjoint corpora: the six improved web and mixed-source domain shifts agree within 0.34 points ($\rho = 0.93$–0.99). (4) Gains concentrate where a model must read connected discourse: seven passage-conditioned tasks carry 8 of the 12 robust downstream wins and 1 of the 8 losses (§7). (5) The effect is organized by the target distribution, not the input tier (98% vs. 0.3% of the variation), so a quality label does not settle whether repair pays.

2 Problem Formulation

Curation with a repair action. A standard pipeline realizes $C_{\text{filtered}} = F(C_{\text{raw}})$ with $F = f_{\text{dedup}} \circ f_{\text{quality}} \circ f_{\text{heuristic}} \circ f_{\text{langid}} \circ f_{\text{url}}$, a composition of keep/discard decisions on a corpus whose discards are irreversible: each document's action set is {keep, discard}. Repair-First adds a third action, *repair*, and two corpus policies:

$$C_{\text{rich}} = F\big(R(C_{\text{raw}})\big), \qquad C_{\text{scarce}} = R(C_{\text{raw}}). \tag{1}$$

In the rich mode, repair improves the candidates the filter judges; in the scarce mode no quality-based discard is applied at curation time, and every document is kept, as its accepted repair or its original, with its audit record.

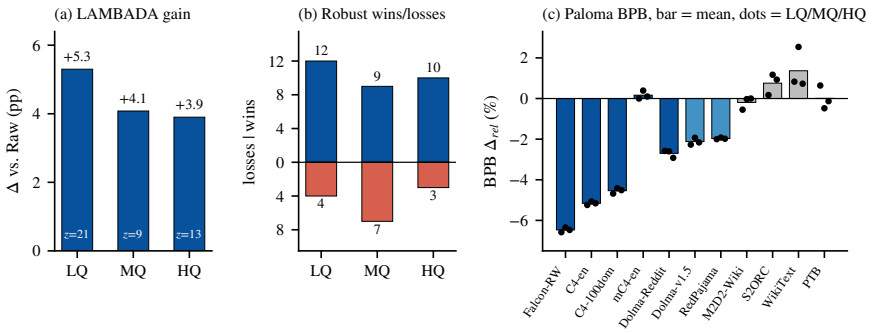


Figure 1: Effect of the full repair pipeline (V2) on fixed-budget pre-training (34.4B tokens) by Nemotron-CC input tier. (a) LAMBADA gain over Raw; z in units of the seed-calibrated noise band (§5). (b) Robust wins (up) and losses (down) over 38 evaluations, $|z| \geq 2$. (c) Paloma relative BPB change per corpus: bar = mean over tiers, dots = the three tiers; web (dark), mixed-source (mid) and curated (grey) corpora. The web and mixed-source shifts recur almost unchanged on three disjoint corpora; the curated ones vary more.

Corruption, recoverability and abstention. We model an observed document as $y = K(x)$: the author's text x passed through a *corruption channel* K (mis-decoding, hard wrapping, word splitting, inserted markup and boilerplate). Semantic quality is a property of x, structural corruption a property of K, and FILTER-AND-DISCARD judges the first through the second; quality criteria studied so far, such as writing style, expertise, facts and educational value (Wettig et al., 2024), all concern x. A corruption is *recoverable* on a span of y when local evidence identifies its inverse, i.e. when the posterior over the clean span concentrates on one candidate: a C1-control cluster identifies a Windows-1252 mis-decoding, a line-final hyphen whose joined form is a word identifies a split word. This is the noisy-channel view of correction (Brill & Moore, 2000) with one addition: where the posterior is not concentrated, the right action is to *abstain*. Repair returns an identified inverse or nothing; a generative rewriter returns a sample from a broad posterior, and where such a system leaves a document unchanged, the choice is a model's judgement rather than the absence of identifying evidence.

Three constraints. We impose the formulation on the repair operator as three constraints checkable by construction. **(C1) Reversibility.** The original is retained, every edit is logged with stage, operation, affected segment and pre/post SHA-256, and protected content is restored byte for byte as saved at extraction (Proposition 1). **(C2) Bounded, non-generative edits**, which operationalize the abstention rule: R is a composition of edits $e = (s, \rho)$, admissible only if (i) s lies in a prose segment, save the logged stage-1–2 repairs that precede routing (*scope*), (ii) a diagnostic predicate fires on s (*evidence*), (iii) ρ deletes a diagnosed artifact or comes from a candidate set bounded by edit cost or by a rule with an explicit precondition (*locality*: never a sample from a model), and (iv) a validator accepts the result or the edit is rolled back (*abstention*). **(C3) Temporal consistency.** Repair must not modernize historical text; stages that consult present-day resources satisfy this only partially (App. L). Semantic preservation is what C1–C2 are designed to secure and make auditable; it is not measured.

Endpoints and hypotheses. Our primary endpoint is training utility: with $U_B(D)$ the downstream performance after training on B tokens drawn from D, we measure the *repair contrast* $U_B(R(T)) - U_B(T)$ on Nemotron-CC tiers T with the tier assignment held fixed, the quantity that matters in the scarce mode and one that separates repair from selection; whether repair flips a filter's decision is a separate, filter-specific quantity (§7). The formulation yields three hypotheses. **H1** (measurement): scorers built to measure semantic quality are insensitive to, or inverted by, many structural corruptions, layout above all, so they can neither remove them nor validate their repair, and such corruption survives selection into every tier (§4). **H2** (utility): removing recoverable corruption raises training utility at a fixed budget, even within tiers a state-of-the-art filter has assigned (§6.1). **H3** (mechanism, exploratory): gains concentrate where corruption interrupts what a model learns from, multi-sentence context, and their size on a target depends on how closely the target resembles repaired text (§6.2–7).

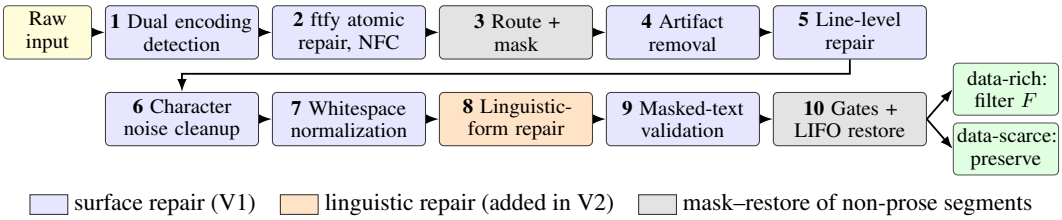


Figure 2: The ten-stage REPAIR-FIRST pipeline. Stages 1–2 recover bytes and characters; stage 3 routes code, mathematics and structured data out of the prose stream and replaces them by sentinels; stages 4–8 repair prose in the masked state; stage 9 validates the masked document (pass, or reject and roll back); stage 10 restores the masked segments in LIFO order with SHA-256 checks—verbatim in the runs of this paper (§3.2). The output enters one of the two modes of equation 1.

Table 1: Algorithmic and modeling components of REPAIR-FIRST, grouped by the decision each one models: evidence ordering (Principle 1) governs the topology and the boundary edits, containment (Principle 2) the routing, and the abstention rule of §2 the linguistic stage and the validator (App. B–E). Each observation supports the design choice on its row; none is a training ablation of that component.

Component	Algorithmic or modeling idea	Supporting observation (scope)
Where may an edit act?		
Stage topology (§3.1)	Order from four ordering constraints; two-detector encoding vote with a precedence rule	Preferred detector wins all 70 real disagreements (blind LLM judge)
Mixed-content router (§3.2)	Container-first priority DFS; 42-feature XGBoost; cost-asymmetric table cascade; inside-out extraction, LIFO restore	Exact restoration (Prop. 1); damage matrix (Table 6)
Which correction is plausible?		
Boundary repair (§3.2)	Lexical/structural cascade (Eq. 2): word split, wrap and real boundary get different edits	5.7–15.5% of logged operations (Fig. 6)
Typed protection (§3.2)	Three-signal code-language fusion with keyword/negative anchors; parser-certified structured data; mask, restore byte for byte	Runs in every experiment; 98.3% language ID on 1,000 unhinted blocks
Linguistic-form repair (§3.3)	Isolate/repair/re-insert workflow keeping vocabulary decisions away from suspect fragments; gated acceptance	V2 vs. V1: net verdict $+10/+5/+10$; lower BPB on 31 of 33 Paloma pairs
When may a repair be committed?		
Validator ensemble (§4)	Three-way sensitivity benchmark; exhaustive search over $2^{13}-1$ scorer subsets; per-defect ownership	KenLM inverted on 4 layout defects; best CPU subset 0.896 (in-sample)

3 THE REPAIR-FIRST FRAMEWORK: PRINCIPLES AND ALGORITHMS

Figure 2 shows the ten stages (V1 runs all but stage 8, V2 all ten; App. B). Two principles derive the design from §2, and Table 1 lists the algorithms that implement them. Stages use mature open tools where they exist; the contribution is how they are ordered, routed, specialized, validated and evaluated.

3.1 EVIDENCE ORDERING: A TOPOLOGY DERIVED FROM TOOL INVARIANTS

Principle 1 (evidence ordering). *An operation that destroys the evidence a later stage needs must run after that stage.* Three of the four ordering constraints of the pipeline instantiate it. (a) *Encoding first*: C1 control characters (U+0080–U+009F) and U+FFFD are the evidence of mis-decoding (Speer, 2019), so stripping them first makes mojibake unrepairable. (c) *Artifacts before lines*: BBCode and multiply escaped entities distort the line-length evidence by which stage 5 recognizes hard wraps. (d) *Validate before restore*: the validator's evidence is a homogeneous prose context, available only while non-prose is masked. The fourth, (b) *mask before layout normalization*, instantiates Principle 2. The constraints fix a partial order among the stages they involve; stage 1's encoding vote, in which charset-normalizer wins disagreements (supported by blind LLM adjudication of the 70 real disagreements among 7,000 web pages), is in App. B.1.

3.2 CONTAINMENT: MIXED-CONTENT ROUTING WITH EXACT RESTORATION

Principle 2 (containment). *An edit may act only inside the content type whose conventions license it; everything else is removed from its reach and restored exactly.* Containment is necessary because no common normalization is safe for all content types (Table 6): tab expansion, line merging and whitespace folding are fatal to code, hyphen re-joining breaks package names, NFC comes closest but preserves canonical equivalence rather than bytes, and NFKC destroys mathematical semantics (Beeton et al., 2017). It requires finding code, mathematics and structured data before any prose edit. The router anchors on the *container format* of a block, outer-first, in a three-level cascade: a `<head>` pre-scan that enables `$`-delimited mathematics only when the page declares it; a container-priority DFS whose table cascade defaults to *data table* when inconclusive, since a dissolved table cannot be recovered; and an XGBoost classifier (Chen & Guestrin, 2016) over 42 features for ambiguous `<div>`/`<span>` containers (App. C). Code identification fuses highlighter hints, Magika (Fratantonio et al., 2025) and Tree-sitter error ratios with keyword anchors (98.3% language-identification accuracy on 1,000 unhinted blocks), and a parser certifies structured data. Detected segments are extracted inside-out, replaced by sentinels and restored last-in-first-out with SHA-256 checks.

Proposition 1 (Exact restoration)**.** *Suppose the detected segments form a non-crossing hierarchy, sentinels are unique and absent from the input, prose repair preserves the identity, multiplicity and order of sentinels, and stored payloads are not modified. Then LIFO restoration reproduces every untouched protected segment, including its nested descendants, byte for byte.*

The proof is by induction over the reverse extraction sequence (App. C.1); it covers the representation saved at extraction, after the logged stage-1–2 decoding and normalization. The framework also specifies minimum-edit repair lanes for typed content (App. C); they were not exercised in our runs, so no gain is attributed to typed *repair*.

3.3 RECOVERABLE PROSE EDITS: LINE BOUNDARIES AND LINGUISTIC FORM

Boundary-aware line repair (stage 5). A newline can split a word, wrap a sentence at a fixed column or mark a real boundary; stage 5 infers this latent role and edits only when evidence identifies it: lexicon-checked dehyphenation ("state-\nof-the-art" keeps its hyphen), a join when fixed-column evidence holds and no blank line, list marker or heading vetoes it, and no change otherwise (equation 2, App. B.3).

Linguistic-form repair (stage 8) applies the abstention rule to spelling, grammar and syntax. To avoid "correcting" rare names and identifiers it isolates suspect, dictionary-unparsable fragments before any vocabulary decision, runs two local grammar-and-syntax passes over them, substitutes a candidate only when a ranking of candidate quality, edit cost and grammaticality clears a threshold, re-inserts it and checks the full sentence; no component calls a generative model (App. B.4). V2−V1 measures the layer as a whole (§6.1). Stage 9 then scores the masked, repaired document with the validator of §4, and a rejected document is rolled back to its original (Algorithm 2).

4 RQ1: STRUCTURAL CORRUPTION AND THE BLIND SPOTS OF QUALITY ESTIMATORS

A three-way sensitivity benchmark. Stage 9 must answer a question standard quality scorers were not designed for: *is the repaired document structurally better than the original?* We built a benchmark of 30 structural anomalies in four families (character distribution, 8; layout, 9; control characters, 7; artifacts, 6; App. D), injected each into the same base texts and had every candidate scorer score both versions. A response is *sensitive-and-correct* if the score moves significantly toward lower quality, *perverse* if it moves toward higher quality (the scorer prefers the corrupted text) and *insensitive* otherwise; a scorer's score on a defect is $S = 100\,P_{\text{correct}} - 100\,P_{\text{perverse}} \in [-100, 100]$.

The good/bad KenLM ensemble inverts on layout defects. Dual KenLM—KenLM (Heafield, 2011) models trained on high- and low-quality reference text, combined after z-score standardization (Kim et al., 2025), extending CCNet's single-model filter (Wenzek et al., 2020)—detects character-distribution and artifact defects well (mojibake 98.4% sensitive-and-correct, HTML residue 92.2%,

navigation-bar boilerplate 98.3%), but rates the corrupted text as *better* for the layout defects that line repair exists to fix: intra-word newlines (61.6% perverse), stray inter-word newlines (70.5%), blank-line runs (65.1%) and chaotic tabs/spaces (60.9%), i.e. S from -22 to -41 (App. E). A plausible, untraced mechanism lies in the n-gram statistics: splitting a rare word such as "baseball" into the common fragments "base" and "ball" around a newline (a frequent symbol in line-formatted Wikipedia training text) lowers perplexity, and whitespace rearrangements barely move it; a validator with this profile would accept layout corruption and reject its repair.

Other estimators are largely insensitive to structure. We evaluated 13 candidate scorers from five families (fastText, Transformer, ByT5, rule-based, and `ftfy` reverse-wrapped as a detector; Table 12). The educational-value classifiers behind Nemotron-CC reach sensitivity totals of only 14.78 and 20.10, against 75.01 for our typed heuristic scorer: they measure the other axis. DCLM's fastText classifier (30.18) responds to macro-level anomalies (curly quotes, PDF artifacts, boilerplate) but hardly to micro-structural ones: structural corruption is a failure mode that current estimators miss or reward (RQ1).

Consequences: selection and validation are different measurement problems (H1). A scorer can respond to a corruption in three ways: *sensitive*, so a filter built on it discards documents for a repairable defect; *insensitive*, so a filter lets damaged documents through and a validator cannot tell a repair from damage; or *inverted*, so a validator rejects correct repairs. A validator must therefore be chosen for sensitivity to the defects repair fixes, not for its value as a selector. And because two of the three classifiers that define Nemotron-CC's tiers are largely insensitive to structure and the third responds mainly to macro-level anomalies, H1 predicts that structural damage survives selection into every tier, the top one included: repair touches 72.6% of HQ documents (§6.2).

Validator design as combinatorial optimization. No single scorer covers all defect families, so we scored every non-empty subset of the 13 tools ($2^{13} - 1$ ensembles) on the benchmark with a composite criterion, under the constraint that the deployed ensemble runs on CPU (Table 14). The best CPU-only subset—DCLM-fastText, LanguageTool, the typed heuristic scorer, `ftfy`-diff, NLTK and `unicodedata`—scores 0.896 (the best overall, with a GPU classifier, 0.913), and was re-measured end-to-end at 0.896 on the same benchmark. The score is not monotone in the subset (adding `textstat` lowers it to 0.894), and tools weak alone earn their place through defects on which they are strongest (`unicodedata`: C1 controls, lone surrogates). Each assigned defect (29 of the 30) is scored exclusively by its pre-assigned best tool, so every per-defect score is attributable to one tool (App. E); Dual KenLM averages $+43.7$ over the 30 defects and is inverted on four.

5 Experimental Setup

Data: 2013–2015 Common Crawl and Nemotron-CC tiers. Common Crawl's WARC-era archive starts with its Summer 2013 crawl, CC-MAIN-2013-20; the three earlier crawls (2008–2012) use the legacy ARC format (Vaughan & Ortiz Suarez, 2024; Common Crawl, 2026). Nemotron-CC starts at the same snapshot (Su et al., 2025), so 2013 is the first year of both the modern archive and a state-of-the-art tiered baseline; extending the window to 2015 makes every tier large enough ($\gg$34.4B tokens) for precise down-sampling. We take the documents that Nemotron-CC v1 assigns to its high (HQ), medium (MQ) and low (LQ) partitions in several 2013–2015 snapshots (the tier is encoded in the release's partition paths; App. F) and, because the released text has already been normalized by NeMo Curator, retrieve each document's *original* text from the Common Crawl record identified by the WARC record ID and URL that v1 preserves. Because a document's tier is set by the maximum of three classifiers' percentile buckets (High = 19, Medium = 12–17, Low = 0–6; Su et al., 2025), every MQ and LQ document lies outside each classifier's top $\approx 10\%$: a top-10% filter on any of them, the policy DCLM applies with its own classifier, discards both tiers, more than half of whose documents receive a repair (§6.2). Nemotron-CC itself rephrases its Low tier with an LLM; repair is the non-generative alternative.

Treatments and budget. **Raw** is the retrieved source text of a tier without repair; **V1** applies the surface stages (all stages except the linguistic-form stage 8) and is up-sampled to the Raw token count, because surface repair shortens text; **V2** adds stage 8, which can lengthen text again, so V2 pools exceed Raw and need no up-sampling. Every run trains for 16,406 iterations at a global batch

of 512 sequences of 4,096 tokens (34.4B tokens), 20 tokens per parameter for the 1,720,574,976-parameter model (Hoffmann et al., 2022). Groups are matched at document granularity (Table 16); V1's up-sampling is under 1%, far from the multi-epoch regime in which repeated data loses value (Muennighoff et al., 2023).

Model, training and evaluation. We pre-train the Qwen3-1.7B-Base architecture from scratch (random initialization; no checkpoint is loaded) with its tokenizer (Yang et al., 2025) in BF16 with AdamW, a warmup–stable–decay schedule (500 + 13,506 + 2,400 cosine iterations) and peak learning rate 3×10^{-4} on eight A100 GPUs (App. F). This is the standard proxy scale of curation research (FineWeb's ablations: 1.71B parameters on ≈28B tokens, Penedo et al., 2024; DCLM's 1B-1x: 1.4B on 28.8B, Li et al., 2024). Each model is evaluated on 27 downstream tasks in five families (language modelling, science/knowledge, commonsense, logic/NLI, reading and open QA; MMLU 5-shot) and on bits per byte (BPB) for 11 Paloma corpora (Magnusson et al., 2024): 38 evaluations (App. G; SQuAD v2's recorded scale is undecoded).

Statistical protocol. The Raw baseline of every tier was pre-trained with three seeds (1234, 1337, 42). We separate σ_{pretrain} (across seeds), σ_{eval} (repeated evaluations of a run) and σ_{binom} (finite test sets, multiple choice only); $\sigma_{\text{tot}} = \sqrt{\sigma^2_{\text{pretrain}} + \sigma^2_{\text{eval}}}$, with σ_{binom} used only when both vanish (App. H). We report $\Delta = m_{\text{repair}} - m_{\text{Raw}}$ and $z = \Delta/\sigma_{\text{tot}}$ and call $|z| \geq 2$ a *robust* win or loss: a descriptive label ("beyond twice the Raw noise band"), not a hypothesis test, since each repaired configuration is a single run (seed 1234, the training framework's default and one of the Raw seeds) and treatment-side variance is unmeasured. Under stricter cutoffs and a binomial floor on σ_{tot} (App. H) V2 stays net-positive on every tier, but the assumptions those cutoffs need (equal treatment variance, independence) only further treatment seeds can replace. The Raw/V1/V2 contrast evaluates each recipe as deployed (including token re-matching); V2−V1 is the observed difference between the two recipes, attributed to stage 8 as a layer only under that caveat (App. L).

6 Results

6.1 RQ2: Does repair recover training utility?

Repair wins on every tier. Across the three tiers the full pipeline wins 31 robust comparisons and loses 14 (10/3, 9/7 and 12/4 on HQ, MQ and LQ; Table 2, Figure 3), and stays net-positive under every stricter counting rule (Table 21). Without any significance filter it improves 68 of 111 tier–evaluation pairs and worsens 35; its downstream gains sum to 91.9 pp against 26.7 pp of losses and its Paloma improvements to 70.0% against 7.5%; its robust downstream wins average +2.7 pp, its losses −1.2 pp; ignoring changes under 1 pp and excluding CB and LAMBADA still leaves 21 improvements against 10 declines (Table 33); and its task mean stays positive without CB (+0.48 pp) or without CB and LAMBADA (+0.32; Table 32).

The effects are consistent across disjoint corpora. The three tiers contain different documents, so each is a separate test of the same repair under one protocol. On the six web and mixed-source Paloma corpora that improve, the three tiers agree to within 0.34 points (Figure 1c, Table 31), and the ordering of all eleven corpora is nearly identical across tier pairs (Spearman $\rho = 0.93$–0.99): a two-way decomposition of the 33 domain effects places 98.2% of their variation between evaluation corpora and 0.3% between tiers (Table 32). The Raw seeds' standard deviation on these corpora is 0.1–0.4%, 5 to 65 times smaller than the shifts, so if repaired runs vary across seeds as the Raw runs do, seed variation cannot produce shifts of this size (the tiers vary the data, not the seed). LAMBADA (Paperno et al., 2016) improves by 3.9–5.3 pp on every tier (9.0–14.7% relative; $z = 8.8$–20.6); seven tasks improve on all three tiers and only WinoGrande worsens on all three. Rankings of individual tasks do not replicate ($\rho \leq 0.19$); most tasks sit near chance at this scale (§7).

Repair depth adds value. The surface stages precede the linguistic stage and supply its input, decoded, masked and line-joined prose (Figure 2): alone they already deliver most of the LAMBADA gain (+2.4/+4.2/+2.9 pp); adding the linguistic stage improves Paloma fit over them on 31 of the 33 tier–corpus pairs, keeps the downstream gains and raises the robust net count by +10/+5/+10 (App. H).

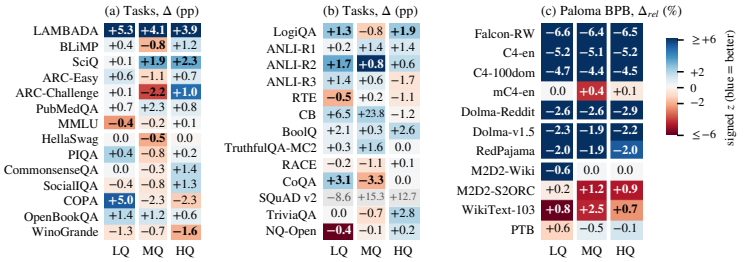


Figure 3: All 38 evaluations under full repair (V2) on each tier. Cell text: change vs. Raw (tasks in pp; Paloma in relative BPB, negative is better). Colour: signed z (blue = better, clipped at ± 6); bold: $|z| \geq 2$. SQuAD v2 (grey) is recorded on an undecoded scale. Paloma rows: web/social crawls (top five), mixed-source (Dolma-v1.5, RedPajama) and curated corpora. Surface-only repair is added in Figure 4; full numbers in App. H.

Table 2: The full repair pipeline (V2) against Raw. Robust W/L: evaluations with $|z| \geq 2$ in favour of / against repair (38 per tier); better/worse: all 37 scored evaluations regardless of significance (ties omitted); gain/loss: summed positive and negative changes; passage-conditioned tasks: the 7 tasks whose items include a multi-sentence passage (§7). Surface-only repair (V1) and per-family Paloma means are in App. H.

	Robust ($\lvert z\rvert \geq 2$)		All evaluations	26 tasks (pp)	11 Paloma (%)	7 passage tasks
Tier	W / L	net	better / worse	gain / loss	gain / loss	better / worse, W / L
HQ	10 / 3	+7	25 / 8	23.1 / 7.9	23.3 / 1.8	6 / 0, 3 / 0
MQ	9 / 7	+2	19 / 18	38.2 / 15.6	22.8 / 4.1	4 / 3, 2 / 1
LQ	12 / 4	+8	24 / 9	30.6 / 3.2	23.9 / 1.6	6 / 1, 3 / 0
All	**31 / 14**	+17	**68 / 35**	91.9 / 26.7	70.0 / 7.5	16 / 4, 8 / 1

6.2 RQ3: Does repair improve corpus utilization across quality regimes?

Repair touches most of every tier, as H1 predicts. Reading *utilization* as repair incidence and utility across tiers: coverage under V2 is 72.63%, 57.82% and 56.52% on HQ, MQ and LQ (Table 20), and the quality score rises for 95.0–97.4% of sampled documents (descriptive; App. L).

The lowest tier gains most. LQ, which every one of Nemotron-CC's three classifiers ranks in its bottom ≈35%, has the most robust wins (12 against 4), the largest LAMBADA gain (+5.3 pp, 14.7% relative) and downstream gains 9.6 times its losses; its repairs are dominated by grammar and syntax edits (37% of operations, against 12% on HQ). MQ is the most mixed tier downstream, although its domain-level shifts match the other tiers. Repair thus complements filtering in every regime (RQ3).

7 Analysis: Why Repair Helps, Where It Does Not, and What to Expect

We test H3 and the account behind it: repair changes the *form* of text, not its content (full analysis in App. K).

Repair helps where text must be read as connected discourse. Grouping tasks by a dataset property, whether an item contains a multi-sentence passage, separates the effects sharply (Table 30): the seven passage-conditioned tasks improve in 16 of 21 task–tier pairs (mean +1.3 pp) and carry 8 of the 12 robust downstream wins but only 1 of the 8 losses, although they are 7 of the 26 tasks. Our reading: hard wraps, split words and boilerplate break the continuity of Raw documents; line-level repair and artifact removal (15–31% of logged operations) restore it.

Repair changes form, not content. Closed-book knowledge and short-context commonsense tasks barely move (pooled +0.26 and −0.09 pp), and six of the eight robust downstream losses fall on them, four on tasks near chance or floor at 1.7B (Table 28) and two (BLiMP, HellaSwag) on MQ.

Bounded edits add no facts (C2), though mending a damaged name can make an existing fact easier to learn; at this scale such gains are rare (one robust win, ARC-Challenge on HQ).

The effect is organized by the target distribution, not by the input tier. Every web and mixed-source corpus except mC4-en improves by 2.0–6.5% on average, two curated corpora (WikiText-103, M2D2-S2ORC) lose 0.8–1.4%, and M2D2-Wikipedia and PTB do not move, so a web-versus-curated split is too coarse; 98% of the variation across the 33 domain effects lies between corpora and 0.3% between tiers. We hypothesize that compatibility between repaired text and target organizes it (forecast iii). An HQ label does not make repair irrelevant, and an LQ label does not make it futile.

What to expect. Testable forecasts (App. K): (i) near-chance passage tasks (BoolQ, RACE, LogiQA) should gain once models can use their passages, as BoolQ already does on every tier; (ii) knowledge tasks should gain only where facts are present but obscured; (iii) targets that measurably resemble the repaired text (e.g., in line and whitespace statistics) should gain, with a stable ordering across sources; (iv) gains should track *recoverable* damage, not damage alone.

How to use REPAIR-FIRST. For scarce data, a document has two values, as training material and as a record of what someone wrote; generated replacements forfeit the second, and when generated text replaces rather than accompanies human text, rare content can be lost (Shumailov et al., 2024). Scarcity raises the cost of a wrong edit as well as of a discard, hence conservative repair with rollback; training on the damaged originals is our Raw condition, which the full pipeline beats on every tier. For plentiful data, repair complements selection, acting on form, which the selection classifiers barely see (§4), while they act on content; since MQ's downstream effects are mixed (3 wins, 4 losses), a practitioner targeting downstream accuracy can repair only HQ and LQ (a mixed corpus we have not trained on) or measure first. To assess repair cheaply, track LAMBADA and web-domain BPB ($z \geq 8.8$ and ≥ 30), in line with the finding that likelihood-based proxy metrics make small experiments more predictive (Magnusson et al., 2025). Repair runs on CPU and, by C3, avoids a rewriter's modern phrasing. The limits of this evidence (one proxy scale, single treatment runs, layer-level attribution, unmeasured tier re-assignment and semantic preservation) are discussed in App. L.

8 RELATED WORK

Filter-based corpora—C4 (Raffel et al., 2020), CCNet (Wenzek et al., 2020), RefinedWeb (Penedo et al., 2023), FineWeb (Penedo et al., 2024), DCLM (Li et al., 2024), Dolma (Soldaini et al., 2024) and Nemotron-CC (Su et al., 2025)—select documents with heuristic, deduplication and model-based filters, refined by educational classifiers, diversity-aware selection (He et al., 2025) or stricter filtering (Emami Gohari et al., 2026). Where they edit a document they delete material (RefinedWeb's line-wise corrections), and FineWeb2 runs `ftfy` and a table-artifact fixer only *after* filtering (Penedo et al., 2025); none, to our knowledge, applies logged, reversible repair of structural defects before the filtering decision. Generative rewriting (Maini et al., 2024; Nguyen et al., 2025; Yu & Xiong, 2026; Maini et al., 2025), with faithfulness rewarded in RePro, LLM-executed cleaning (Mi et al., 2026) and model-written calls to a predefined edit executor (Zhou et al., 2025) run models over the corpus; repair can clean their input, and we claim no superiority over them. Beyond selection, quality has been decomposed into criteria of the text itself, such as writing style, expertise, facts and educational value (Wettig et al., 2024), all properties of x in our notation; structural integrity is a property of the channel K. Noisy-channel correction is classical (Brill & Moore, 2000); we add abstention as a curation outcome and evaluation by pre-training utility.

9 CONCLUSION

We framed curation as a choice among keeping, discarding and repairing, and modeled structural damage as a partially invertible corruption channel. Selection scorers are largely blind to that channel, so damage survives into every tier; repair that inverts only what evidence identifies, and abstains elsewhere, recovers training utility on every tier, including the two a strict filter would discard, with effects organized by the target distribution. Structural integrity deserves a place as a curation axis of its own, beside semantic quality.

ETHICS STATEMENT

All training data derive from publicly available Common Crawl snapshots (2013–2015) as tiered in the public Nemotron-CC v1 release; we collect no new web data. The prose stages do not generate content, and every modification is logged and traceable to the source bytes by hash. The pipeline performs no detection, de-identification or filtering of personally identifiable information or adult content: these are semantic properties of a document, not structural defects, so the decision is left to the users of repaired corpora, who must make it before any deployment. Recovering documents that FILTER-AND-DISCARD would remove can also recover harmful or biased content, and checkpoints trained on repaired data may differ in risk from the Raw baselines in either direction; no comparative safety evaluation was performed. Repair can also change names, quantities, minority language forms or historical spellings; logged, reversible edits make such changes auditable but do not rule them out. The pipeline targets English text, and linguistic repair may normalize dialectal or deliberately non-standard text, so we recommend auditing repaired corpora for such homogenization. Stage 4 removes boilerplate such as cookie-consent banners; no content is removed because of its topic. No LLM edits any text in this work: code, mathematics and structured data are restored verbatim, and LLMs serve only as judges (AI use statement).

REPRODUCIBILITY STATEMENT

The stage tools and configurations are in App. B; the routing cascade, table classifier, restoration proof and code gates in App. C; the 30-defect benchmark and scoring protocol in App. D–E; the data construction, corpus statistics, hyperparameters and training dynamics in App. F; the evaluation suite in App. G; the noise-calibration protocol in §5; and the complete per-task results with σ_{tot} per arm in App. H (components in the released data files). The repaired corpora of every tier and configuration (surface and linguistic) are available anonymously at `https://huggingface.co/datasets/anonymous-md/ProseOnlyRepair_X` with $X \in \{$`LQ`, `MQ`, `MQ1`, `HQ`, `linguistic_LQ`, `linguistic_MQ`, `linguistic_MQ1`, `linguistic_HQ`$\}$ (the MQ tier is split into `MQ` and `MQ1` by the per-dataset size limit). The `Codes` repository holds a reference implementation of the ten-stage specification, and the scripts that regenerate every table and figure of this paper from the recorded results are released alongside the corpora; it is not the code that produced the latest released corpora, as we are still sorting out and cleaning the latest codes for releasing. The codes, commit and configuration behind each reported corpus will be archived and pinned for the camera-ready. Not released: model checkpoints, the fitted router parameters and the judge prompts, which we will add to the same repository for the camera-ready version.

AI USE STATEMENT

We used large language model assistants to assist in preparing this manuscript from an author-written Non-English thesis: translation into English, editing and proof-reading, refining writing, organization and structuring, and literature search, every reference found this way having been checked by the authors at its source. Gemini models were additionally used *as experimental instruments* in three predefined roles: (i) as a blind adjudicator of 70 encoding-detector disagreements (App. B); (ii) as an external reference for the Pearson column of the 30-defect scorer benchmark (App. E); and (iii) as the quality scorer for the LQ quality-shift analysis (§6.2). The judge was Gemini 3.1 Pro Preview (Google, 2026); for judgments whose quality it did not affect, in the authors' assessment, we switched to the faster Gemini 2.5 and 2.0 Flash and Flash-Lite variants. The framework's specification permits code blocks shorter than 50 characters to be sent, with their context, to an LLM-based rescue module (§3.2); that module was not run for any corpus in this work because code blocks are protected rather than repaired in the scope of this work. And no LLM edited any training text. Apart from these uses, we did not use generative AI tools to generate research ideas, formulate or refine hypotheses, design or modify the experimental methodology, run or simulate experiments, produce experimental data/results, or author citations that were not independently verified. All AI-assisted text, tables, figures and code were checked against our records and revised or validated by the authors. The research ideas, experimental designs, experimental data and reported numerical results are the authors' own, and the authors take full responsibility for the content of this work, including all text, claims, analyses and artifacts produced with the assistance of AI tools.

REFERENCES

Yannis Assael, Thea Sommerschield, Alison Cooley, Brendan Shillingford, John Pavlopoulos, Priyanka Suresh, Bailey Herms, Justin Grayston, Benjamin Maynard, Nicholas Dietrich, Robbe Wulgaert, Jonathan Prag, Alex Mullen, and Shakir Mohamed. Contextualizing ancient texts with generative neural networks. *Nature*, 645:141–147, 2025. doi: 10.1038/s41586-025-09292-5.

Barbara Beeton, Asmus Freytag, and Murray Sargent, III. Unicode technical report #25: Unicode support for mathematics. The Unicode Consortium, Revision 15, 2017. https://www.unicode.org/reports/tr25/.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Eric Brill and Robert C. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pp. 286–293, Hong Kong, 2000. Association for Computational Linguistics. doi: 10.3115/1075218.1075255.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. ACM, 2016. doi: 10.1145/2939672.2939785.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Common Crawl. Crawl collection index. https://index.commoncrawl.org/collinfo.json, 2026. Lists the 2008–2009, 2009–2010 and 2012 ARC-file crawls and all later crawls; accessed 25 September 2026.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges: Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment*, volume 3944 of *Lecture Notes in Computer Science*, pp. 177–190. Springer, 2006.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. The CommitmentBank: Investigating projection in naturally occurring discourse. In *Proceedings of Sinn und Bedeutung 23*, pp. 107–124, 2019.

Hajar Emami Gohari, Swanand Ravindra Kadhe, Syed Yousaf Shah, Constantin Adam, Abdulhamid Adebayo, Praneet Adusumilli, et al. GneissWeb: Preparing high quality data for LLMs at scale. In *International Conference on Learning Representations*, 2026.

Yanick Fratantonio, Luca Invernizzi, Loua Farah, Kurt Thomas, Marina Zhang, Ange Albertini, et al. Magika: AI-powered content-type detection. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering*, pp. 2638–2649, 2025.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Henry Sleight, John Hughes, Tomasz Korbak, Rajashree Agrawal, Dhruv Pai, Andrey Gromov, Daniel A. Roberts, Diyi Yang, David L. Donoho, and Sanmi Koyejo. Is model collapse inevitable? Breaking the curse of recursion by accumulating real and synthetic data. In *First Conference on Language Modeling (COLM)*, 2024. arXiv:2404.01413.

Google. Gemini 3.1 Pro Preview: model card and API documentation. https://ai.google.dev/gemini-api/docs/models/gemini-3.1-pro-preview, 2026. Model code `gemini-3.1-pro-preview`; accessed 25 September 2026.

Hongyi He, Xiao Liu, Zhenghao Lin, Mingni Tang, Yi Cheng, Jintao Wang, et al. Learning from the best, differently: A diversity-driven rethinking on data selection. *arXiv preprint arXiv:2510.18909*, 2025.

Kenneth Heafield. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pp. 187–197, 2011.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, et al. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems 35*, pp. 30016–30030, 2022.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength natural language processing in Python, 2020. https://doi.org/10.5281/zenodo.1212303.

Siming Huang, Tianhao Cheng, Jason Klein Liu, Weidi Xu, Jiaran Hao, Liuyihan Song, et al. OpenCoder: The open cookbook for top-tier code large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33167–33193, 2025.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2567–2577, 2019.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Rabeeh Karimi Mahabadi, Sanjeev Satheesh, Shrimai Prabhumoye, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-CC-Math: A 133 billion-token-scale high quality math pretraining dataset. In *International Conference on Learning Representations*, 2026.

Yungi Kim, Hyunsoo Ha, Sukyung Lee, Jihoo Kim, Seonghoon Yang, and Chanjun Park. Rethinking KenLM: Good and bad model ensembles for efficient text quality filtering in large web corpora. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 53–58, 2025.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, et al. Natural Questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, 2017.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6086–6096, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1612.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, et al. DataComp-LM: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems 37 (Datasets and Benchmarks Track)*, pp. 14200–14282, 2024.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, 2022.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. LogiQA: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 3622–3628, 2020.

Ren Ma, Jiantao Qiu, Chao Xu, Pei Chu, Kaiwen Liu, Pengli Ren, et al. AICC: Parse HTML finer, make models better — a 7.3T AI-ready corpus built by a model-based HTML parser. *arXiv preprint arXiv:2511.16397*, 2025.

Ian Magnusson, Akshita Bhagia, Valentin Hofmann, Luca Soldaini, Ananya Harsh Jha, Oyvind Tafjord, et al. Paloma: A benchmark for evaluating language model fit. In *Advances in Neural Information Processing Systems 37 (Datasets and Benchmarks Track)*, pp. 64338–64376, 2024.

Ian Magnusson, Nguyen Tai, Ben Bogin, David Heineman, Jena D. Hwang, Luca Soldaini, et al. DataDecide: How to predict best pretraining data with small experiments. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 42487–42502, 2025.

Pratyush Maini, Skyler Seto, Richard Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14044–14072, 2024. doi: 10.18653/v1/2024.acl-long.757.

Pratyush Maini, Vineeth Dorna, Parth Doshi, Aldo Carranza, Fan Pan, Jack Urbanek, et al. BeyondWeb: Lessons from scaling synthetic data for trillion-scale pretraining. *arXiv preprint arXiv:2508.10975*, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Tiantian Mi, Dongming Shan, Zhen Huang, Yiwei Qin, Muhang Xie, Yuxuan Qiao, et al. Data Darwinism part II: DataEvolve — AI can autonomously evolve pretraining data curation. *arXiv preprint arXiv:2603.14420*, 2026.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018.

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, et al. Scaling data-constrained language models. In *Advances in Neural Information Processing Systems 36*, pp. 50358–50376, 2023.

Thao Nguyen, Yang Li, Olga Golovneva, Luke Zettlemoyer, Sewoong Oh, Ludwig Schmidt, and Xian Li. Recycling the web: A method to enhance pre-training data quality and quantity for language models. In *Conference on Language Modeling (COLM)*, 2025.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4885–4901, 2020.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, 2016.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, et al. The RefinedWeb dataset for Falcon LLM: Outperforming curated corpora with web data only. In *Advances in Neural Information Processing Systems 36 (Datasets and Benchmarks Track)*, pp. 79155–79172, 2023.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems 37 (Datasets and Benchmarks Track)*, pp. 30811–30849, 2024.

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, et al. FineWeb2: One pipeline to scale them all — adapting pre-training data processing to every language. In *Conference on Language Modeling (COLM)*, 2025. Released pipeline: https://github.com/huggingface/fineweb-2, file `fineweb-2-pipeline.py`.

Jake Poznanski, Aman Rangapur, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, et al. olmOCR: Unlocking trillions of tokens in PDFs with vision language models. *arXiv preprint arXiv:2502.18443*, 2025.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, 2018.

Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, 2011.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. AI models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, 2024. doi: 10.1038/s41586-024-07566-y.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, et al. Dolma: An open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15725–15788, 2024.

Robyn Speer. ftfy. Zenodo, version 5.5.1, 2019. https://doi.org/10.5281/zenodo.2591652.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-CC: Transforming Common Crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2459–2475, 2025.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pp. 4149–4158, 2019.

TEI Consortium. TEI P5: Guidelines for electronic text encoding and interchange, §3.5 simple editorial changes. https://www.tei-c.org/release/doc/tei-p5-doc/en/html/CO.html, 2026. Version 4; accessed September 2026.

Thom Vaughan and Pedro Ortiz Suarez. Web archiving file formats explained. Common Crawl blog, https://commoncrawl.org/blog/web-archiving-file-formats-explained, 2024. 1 March 2024.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Position: Will we run out of data? Limits of LLM scaling based on human-generated data. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 49523–49544. PMLR, 2024. URL https://proceedings.mlr.press/v235/villalobos24a.html.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392, 2020.

Maurice Weber, Daniel Y. Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, et al. RedPajama: An open dataset for training large language models. In *Advances in Neural Information Processing Systems 37 (Datasets and Benchmarks Track)*, pp. 116462–116492, 2024.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, 2017.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 4003–4012, 2020.

Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. QuRating: Selecting high-quality data for training language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 52915–52971, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zichun Yu and Chenyan Xiong. RePro: Training language models to faithfully recycle the web for pretraining. In *International Conference on Machine Learning*, 2026. arXiv:2510.10681.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

Fan Zhou, Zengzhi Wang, Qian Liu, Junlong Li, and Pengfei Liu. Programming every example: Lifting pre-training data quality like experts at scale. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79281–79327, 2025.

A Notation and Glossary

Table 3: Notation and terminology used throughout the paper.

Symbol / term	Definition
C_{raw}	Raw crawl before filtering
C_{rich}, C_{scarce}	Output of the two deployment modes: $F(R(C_{\text{raw}}))$ and $R(C_{\text{raw}})$ (no quality-based discard)
F	Filter-and-discard operator, a composition of keep/discard decisions on a corpus (deduplication is corpus-level, the rest document-level)
R	Repair operator, applied before F
x, $y = K(x)$	The author's text; the observed document, produced by a corruption channel K (mis-decoding, hard wrapping, word splitting, inserted markup and boilerplate)
Recoverable span	A span of y whose inverse local evidence identifies, i.e. the posterior over the clean span concentrates on one candidate
Abstention	Leaving a span unchanged, or rolling a document back, when no candidate is identified; operationalized by C2 (evidence, locality, abstention)
Principles 1, 2	Evidence ordering; containment (§3)
H1, H2, H3	Measurement, utility and mechanism hypotheses (§2)
$U_B(D)$	Downstream performance after training on B tokens drawn from D
S	Per-defect sensitivity score, $100\,P_{\text{correct}} - 100\,P_{\text{perverse}}$
Raw / V1 / V2	Nemotron-CC tier data without repair / + surface repair, up-sampled to the Raw token count / + linguistic repair (stage 8)
HQ / MQ / LQ	Nemotron-CC high / medium / low quality tiers
σ_{pretrain}, σ_{eval}, σ_{binom}, σ_{tot}	Seed, evaluation-repeat, finite-sample and total noise (§5)
z	Standardized effect, $\Delta/\sigma_{\text{tot}}$; $\lvert z\rvert \geq 2$ is called a robust verdict, a descriptive label
Structural anomaly	Any non-semantic element attached to the surface of a text: structural errors (incl. artifacts) and structural redundancy
Segment	Logically contiguous span of one content type (prose, code, mathematics, structured data)
Mask–restore	Non-prose segments are extracted and masked before prose repair and put back afterwards
PUA sentinel	Private-Use-Area character standing in for a masked segment; restored in LIFO order
BPB	Bits per byte (Paloma); lower is better

B Stage Specifications

B.1 Stage 1: dual encoding detection

Bytes are decoded by a five-priority cascade (Algorithm 1): a byte-order mark determines the encoding directly; otherwise the HTTP `Content-Type` charset, then the HTML `<meta>` charset, then a vote of two detectors—chardet 7.x (statistical) and charset-normalizer (consistency probing without a statistical prior)—and finally a lenient UTF-8 decoding that maps undecodable bytes to U+FFFD, which keeps error positions visible to stage 2. When the detectors disagree, charset-normalizer wins. On 7,000 real web pages the two detectors disagreed on 147 files (2.1%); 77 of these (1.1%) were alias or subset pairs that decode the file identically (e.g. `ASCII`⊂`UTF-8`, `latin_1`/`ISO-8859-1`, `cp1252`/`Windows-1252`), and 70 (1.0%) were real disagreements. A Gemini model, shown both decodings blind and asked which better fits natural-language coherence and plausible character distribution, sided with charset-normalizer in all 70 cases; chardet's failure mode was a confident mis-assignment of UTF-8 or Windows-1252 text to rare encodings such as `utf-7` or `ISO-2022-JP`. For English text a fast path precedes the detector vote—strict UTF-8, then Windows-1252 if the bytes look like Windows-1252 and the result is acceptable. Encoding detection and language identification are kept separate: asked to identify the language of system documentation such as man pages, charset-normalizer largely returns `Unknown`.

Table 4: The ten stages and their core operations. Most stages use lightweight CPU tools; the stage-9 validator ensemble runs on CPU alone.

#	Stage	Core operation (tools)
1	Dual encoding detection	BOM → HTTP header → `<meta>` → detector vote (chardet 7.x + charset-normalizer; charset-normalizer wins disagreements) → lenient UTF-8 decoding; English fast path (Alg. 1)
2	ftfy atomic repair	convergence loop over seven `ftfy` settings: mojibake, C1 controls, ligatures, character width, line breaks, control characters; **NFC, never NFKC**
3	Routing + masking	three-level DOM cascade or plain-text fallback; PUA sentinels; SHA-256 of masked segments
4	Structural-artifact removal	BBCode and forum markup, deprecated HTML4 tags, multiply escaped entities, Usenet quote prefixes, navigation breadcrumbs, cookie-consent banners, PDF headers/footers
5	Line-level repair	hyphenated-wrap re-join (three conditions + dictionary check) and fixed-column unwrap (Textunwrap; spaCy for complex boundaries)
6	Character-noise cleanup	NFC re-confirmation; NBSP/ideographic space → space, zero-width characters and BOM removed; native PUA characters removed except sentinels (textacy)
7	Whitespace normalization	tabs → spaces; trailing whitespace removed; blank-line runs collapsed to at most one blank line; leading/trailing blank lines removed
8	Linguistic repair	extract suspect fragments → two grammar-repair passes on the fragments → gated fragment repair → re-insert → grammar-and-syntax pass; no generative LLM
9	Masked-text validation	six-tool ensemble on the masked text (§4); pass, or reject ⇒ rollback
10	Gates + restoration	type-specific quality gates (specified for code, App. C); LIFO restoration; SHA-256 verification

Algorithm 1 Stage 1 decoding cascade

procedure DECODE($bytes$, $http_hdr$, $html_meta$)
  if $bytes$ starts with a BOM **then**
    return decode by BOM ▷ P1
  end if
  if $http_hdr$.charset is valid **then**
    return decode by HTTP header ▷ P2
  end if
  if $html_meta$.charset is valid **then**
    return decode by `<meta>` ▷ P3
  end if
  if $bytes$ decode strictly as UTF-8 **then**
    return decode as UTF-8 ▷ P4-fast
  end if
  if $bytes$ look like Windows-1252 and quality is acceptable **then**
    return decode as Windows-1252 ▷ P4-fast
  end if
  $c_1 \leftarrow$ CHARDET($bytes$); $c_2 \leftarrow$ CHARSET_NORMALIZER($bytes$)
  if $c_1 = c_2 \neq \varnothing$ **then**
    return decode by c_1 ▷ P4, agreement
  end if
  if $c_2 \neq \varnothing$ **then**
    return decode by c_2 ▷ P4, charset-normalizer wins
  end if
  return decode as UTF-8, undecodable bytes → U+FFFD ▷ P5, always succeeds
end procedure

B.2 STAGE 2: FTFY ATOMIC REPAIR

`ftfy` (Speer, 2019) runs as a convergence loop, re-applying its fixers until the output is stable, with seven settings: `fix_encoding` (mojibake such as `cafÃ©`→café; ftfy's documentation reports that a pre-release of version 4.0, evaluated on 30.88M tweets, produced a sin-

gle false positive, which was subsequently fixed[1]), `fix_c1_controls` (Windows-1252 typographic characters—dashes, ellipses, smart quotes—mis-decoded as Latin-1 C1 controls are mapped back), `fix_latin_ligatures` (ligatures such as the single "fi" character are split into letters), `fix_character_width` (full-width Latin characters to half-width; enabled for English corpora, while ftfy's documentation advises disabling it for CJK text), `fix_line_breaks` (CR, CRLF and the Unicode line and paragraph separators to LF, so that a paragraph boundary is always the same token), `remove_control_chars`, and `normalization="NFC"`, which must follow the C1-control repair (NFC does not touch C1 bytes). NFC composes canonically equivalent sequences (a change of bytes, not of canonical meaning; in stage 2 it is applied to the whole decoded document before masking, and the restoration guarantee of Proposition 1 therefore covers the representation saved at extraction, after stages 1–2); NFKC additionally applies compatibility mappings ($\mathbb{R} \rightarrow$R, $x^2 \rightarrow$x2, $H_2O \rightarrow$H2O) and, in the words of the Unicode Consortium, "would destroy the semantics of many mathematical expressions, and should never be used with mathematical texts" (Beeton et al., 2017). The protection holds only if the downstream tokenizer does not re-normalize with NFKC: SentencePiece normalizes its input with NFKC by default (Kudo & Richardson, 2018), whereas the tokenizer used by Qwen3 applies NFC, forming a consistent protection chain with the pipeline.

B.3 Stages 4–7

Stage 4 removes structural artifacts: BBCode and phpBB/vBulletin forum markup, deprecated HTML4 tags (`<font>`, `<center>`, `<marquee>`), multiply escaped entities, Usenet quote prefixes, navigation breadcrumbs, cookie-consent banners and PDF extraction residue (leftover page headers, footers and line numbers). Cookie banners are handled in two layers: (i) at extraction time, DOM-level identifiers of consent-management platforms (e.g. OneTrust, Cookiebot, TrustArc, Quantcast Choice; `aria-label="Cookie Consent"`, `role="dialog"` with cookie-policy descriptors) drawn from the EasyList Cookie list and the IAB TCF registry; (ii) for unmarked banners, NeMo Curator's cross-document repeated-paragraph filter (a paragraph repeated exactly in $\geq$10 documents of one site or $\geq$100 corpus-wide) conjoined with a cookie vocabulary of 30+ terms; both conditions must hold. **Stage 5** treats two defects separately because their generative processes differ. A *hyphenated wrap* is re-joined only when the line ends in an ASCII hyphen or soft hyphen, the character before it is a letter, and the next line starts with a letter; for compounds such as "state-\nof-the-art" the join is accepted only if the result is a dictionary word, otherwise the hyphen is kept. A *fixed-column hard wrap* (RFCs, plain-text e-mail, Usenet, early Internet archives) is never merged across a blank line or into a following list item, enumeration or all-caps heading, and line length serves as the statistical diagnostic: in fixed-width text most non-final lines cluster near the column width (e.g. 70–74 characters), whereas paragraph-final lines vary. Textunwrap, which is rule-based, performed best among the candidate tools and is the primary tool; a spaCy model (Honnibal et al., 2020) based on dependency structure and context is added as a supplementary check for complex boundaries and reduces residual title merges. Stage 5 sits between stages 4 and 6 so that line-length statistics are not perturbed by BBCode tags or full-width spaces. **Stage 6** (textacy) re-confirms NFC after the string surgery of stages 4–5; converts non-breaking spaces (U+00A0), ideographic spaces (U+3000) and other space variants to ASCII spaces and removes zero-width space, non-joiner and joiner, the byte-order mark and the word joiner; and removes native private-use characters (U+E000–U+F8FF, e.g. from early web-font schemes or CMS internal markers) while sparing the sentinels injected by stage 3. **Stage 7** replaces remaining tabs by equal-width space sequences, strips trailing whitespace, collapses runs of three or more blank lines so that at most one blank line remains as a paragraph separator, and removes leading and trailing blank lines.

The stage-5 decision rule. For neighbouring prose lines ℓ_i, ℓ_{i+1} the line-level repair of §3.2 is

$$r(\ell_i, \ell_{i+1}) = \begin{cases} \mathrm{dehyphenate}(\ell_i, \ell_{i+1}) & \text{line-final hyphen, letters on both sides, joined form in lexicon,} \\ \mathrm{join}(\ell_i, \ell_{i+1}) & \text{fixed-column line-length evidence, no boundary veto,} \\ (\ell_i, \ell_{i+1}) & \text{otherwise (blank line, list marker, heading, unresolved),} \end{cases} \tag{2}$$

evaluated only inside the prose skeleton. The first case removes a hyphen only when the joined form is in the lexicon, a conservative heuristic with known false positives ("re-creation" and "recreation" are both words); the second never merges across a blank line, into a list item or into a heading;

[1] https://ftfy.readthedocs.io/en/v5.5.1/

the third retains the original whenever the evidence is unresolved, so the rule errs toward leaving a boundary in place.

B.4 Stage 8: linguistic repair (grammar and syntax)

Spelling errors (non-word and real-word), agreement and tense errors are repaired by edit distance, morphological rules, dictionary matching and local n-gram statistics; syntactic repair addresses dropped punctuation, sentence boundaries misplaced by line truncation and split clauses. Candidate substitution is restricted to fragments flagged as suspect, because applying it to all text would "correct" rare proper nouns, technical terms and identifiers. A one-pass screen (visual confusions, merges and splits, spacing errors, UTF-8 residue, garbage tokens) routes documents with no suspect spans, or whose document-level suspect score is below a threshold, directly to grammar and syntax repair. For the others, the stage runs five steps: (1) *extract* suspect, dictionary-unparsable fragments so they do not disturb vocabulary-based spelling decisions; (2) *two grammar-repair passes* over the extracted fragments (local word-form correction and preliminary local syntax recovery, repeated exactly twice), so that step 3 sees cleaner fragments; (3) *fragment repair*: candidates c for a fragment $\tilde{f}$ (the fragment after step 2) are ranked by a character-confusion score, by edit cost, by linguistic fit in the document context, by word frequency and by a non-garbage score; the top-ranked candidate is substituted only when its score clears an acceptance threshold τ_{apply}, otherwise the fragment keeps the result of step 2; (4) *re-insertion*; (5) a final grammar-and-syntax pass over the full-sentence context that corrects incoherence introduced by re-insertion and checks dependency structure. No component of stage 8 calls a generative LLM, and the V2−V1 contrast measures stage 8 as a layer. The training set, frequency-to-cost mapping, candidate limits and dictionaries of step 3 are configuration of the experiment code and are not reported here.

B.5 Stages 9–10: validation and restoration

Stage 9 is described in §4 and App. E. Stage 10 restores the masked segments in LIFO order of masking, so nested placeholders resolve naturally, and only documents whose restored segments pass the SHA-256 integrity check are output as final repaired documents. The specification adds a type-specific gate for repaired segments (for code, App. C): a repaired segment is restored only if it passes the gate, and otherwise the original is restored if it passes the language-agnostic layer. In the pre-training experiments of this paper this branch is never taken: code, mathematics and structured data are extracted, protected and put back *unchanged*, so the gate reduces to the identity check, and these operations are not counted as repair operations.

B.6 The repair transaction and the two deployment modes

Algorithm 2 summarizes the per-document transaction. A failed repair rolls back to the original document; Table 5 states how the two deployment policies of equation 1 use the outcome.

Table 5: How the transaction outcome is used by the two deployment policies of equation 1 (a policy specification; our pre-training study evaluates the Raw/V1/V2 repair contrast within tiers, §2).

Transaction outcome	Data-rich mode, $F(R(C))$	Data-scarce mode, $R(C)$
Repair accepted	repaired document is judged by the filter F	repaired document kept, with its audit record
Repair rejected or integrity failure	original document is judged by F	original document kept

C Hierarchical Mixed-Content Routing

The damage matrix. Table 6 records, from sampled and manually audited real documents, how common text-normalization operations—applied uniformly to whole documents by conventional pipelines—affect code, mathematics and structured data; it motivates the mask-before-normalize constraint of §3.1.

Algorithm 2 The REPAIR-FIRST transaction for one document d (V1 omits stage 8). Stack H holds one tuple (sentinel π, segment g, type τ, SHA-256 h) per masked segment.

procedure REPAIR(d)
  $x \leftarrow$ DECODE(d) ▷ stage 1, Algorithm 1
  $x \leftarrow$ FTFYFIX(x) ▷ stage 2: convergence loop, NFC
  $(s, H) \leftarrow$ ROUTEANDMASK(x) ▷ stage 3: skeleton s, stack H
  for $j \in \{4, 5, 6, 7, 8\}$ **do** ▷ prose repair in the masked state
    $s \leftarrow r_j(s)$; log (stage, operation, span, pre/post SHA-256)
  end for
  if VALIDATE(s) = *reject* **then**
    return d ▷ stage 9: roll back to the original
  end if
  while $H \neq \varnothing$ **do** ▷ stage 10: last in, first out
    $(\pi, g, \tau, h) \leftarrow$ POP(H)
    $g' \leftarrow$ GATEORROLLBACK$_\tau(g)$ ▷ repaired g if it passes, else g (App. C)
    verify SHA256(g) = h; replace sentinel π in s by g'; log the hash of g' if it differs
  end while
  return s if every integrity check passed, else d
end procedure

Table 6: Damage matrix: severity of common text-normalization operations on non-prose content when applied without masking (from sampled and audited documents). Fatal = severe and irreversible.

Operation	Code	Math	Struct. data	Typical failure
NFC normalization	safe*	safe*	safe*	*canonical equivalence, not byte identity: a decomposed literal changes length
Tab → spaces	fatal	low	high	Makefile recipes require literal tabs; TSV columns merge
Line-break merge	fatal	medium	high	`x=1 y=2 z=x+y` → `x=1y=2z=x+y`; `align` environments merged; table rows flattened
Whitespace compress	fatal	medium	high	Python indentation lost; fixed-width column alignment lost
Curly → straight quotes	medium	low	low	lossy in some code contexts (e.g. rich-text comments); loses typographic signal
HTML entity decode	high	low	medium	`<div>` in a tutorial becomes a live tag
Hyphen-wrap re-join	high	low	medium	`scikit-\nlearn` → `scikitlearn`; `--max-\nretries` → `--maxretries`; URLs broken

Design principles. (1) Container format over content semantics: the outermost markup or delimiter system decides the route, because each repairer's parser is built for one container. (2) One route repairs only the outermost container; sub-blocks that belong to another route (e.g. a LaTeX `array` inside an HTML `<td>`) are thrown back to the router as new blocks, so nesting of any depth is handled by recursion. (3) DOM first, plain-text fallback. (4) Outer-first priority: strong outer signals (`<pre>`, `<table>`, `<math>`, fenced code, display math) are tested before weak inner ones (inline code, inline math), so that once an outer structure matches, its subtree is locked and no "formula" or "table" is mis-detected inside content that is already code.

Level 1: metadata pre-scan. `<head>` is scanned for JSON-LD, `<meta>` and Microdata, and the presence and configuration of MathJax/KaTeX are recorded. Inline `$...$` mathematics is enabled only when the page's configuration explicitly declares it; otherwise it is disabled by default, following expert consensus, to avoid prices such as "$2.50" and shell variables.

Level 2: single-traversal DOM classification. Table 7 lists the priority chain. Strong-boundary tags (`pre`, `code`, `math`, `table`, `ol`, `ul`, `dl`, `form`, `figure`, `figcaption`, `blockquote`, `script`, `style`) stop the descent in principle. Before the stop is applied,

every strong-boundary tag receives a light subtree scan: if it contains tags of the structured-data, mathematics or code classes, the stop is cancelled and the DFS continues. Structural containers (lists, key–value lists, data tables) additionally throw embedded code, mathematics or deeper structured data back to the router as independent blocks.

Table 7: Level-2 DOM classification priority chain (highest first).

#	Signal	Details	Class
1	JSON-LD / Microdata	`<script type="application/ld+json">`, Microdata containers	structured
2	MathML / renderers	`<math>`; `MathJax`, `mjx-`, `katex`, Wikipedia `mwe-math`; `<img>` with TeX in `alt`/`title`/`data-latex`	math
3	Code blocks	`<pre><code>`; highlighter classes (`language-*`, `lang-*`, `highlight`, `codehilite`, `sourceCode`, `prettyprint`, `brush:`, `hljs`, `chroma`, `rouge`); legacy `<xmp>`/`<listing>`/`<plaintext>`/`<tt>`	code
4	Inline code	`<code>` outside `<pre>`, `<kbd>`, `<samp>`, `<var>`	code
5	Data tables	`<table>` classified as data table by the cascade of Table 8	structured
6	Lists	`<ul>`, `<ol>`, `<dl>` after subtree scan	structured

Table 8: Layout-table vs. data-table cascade applied to every `<table>`, highest confidence first. Layout tables are dissolved and their cells re-routed; data tables enter the structured-data route.

Tier	Signal	Verdict
1 (ARIA)	`role="presentation"` or `role="none"`; HTML 4.01 `datatable="0"`	layout
1 (ARIA)	`role="table"`	data
2 (semantic)	any of `<caption>`, `<thead>`, `<tfoot>`, `<colgroup>`, `<th>`; `summary` attribute; any `<td>` with `headers`/`scope`/`abbr`	data
3 (structural)	nested `<table>`; contains `<form>`/`<embed>`/`<applet>`/`<iframe>`/`<object>`; $\leq$1 row or $\leq$1 column	layout
4 (legacy visual)	`cellspacing="0"` with border absent or 0; percentage `width` without border	layout
5 (content)	>50% of cells contain `<img>` (gallery)	likely layout (conditional)
default	no conclusive signal	data (asymmetric loss: a mis-kept layout table adds removable noise; a dissolved data table loses structure irreversibly)

Level 3: residual classifier. (Training set, label definitions, split and hyperparameters of this classifier are not reported here and are not in the public reference implementation and will be released with the pinned experiment code (Reproducibility statement); no routing confusion matrix is available, so the router is presented as a design choice rather than a validated model.) Ambiguous `<div>`/`<span>` containers are scored by an XGBoost classifier over 42 features in nine groups: DOM structure (tag one-hot, depth, sibling/child counts, parent tag, neighbouring tags, class presence; 8), inline style (monospace, centering; 2), content patterns (LaTeX delimiters, programming keywords, JSON openers, brackets; 4), character-class ratios (letters, digits, symbols, whitespace; 4), code statistics (keyword density, bracket balance, semicolon-line ratio, indentation consistency; 4), math statistics (math Unicode density, delimiter count, Greek letters; 3), table features (`<th>`, `<thead>`, rows, columns; 4), block signals (link density, longest run, repeated-character ratio; 3) and a context window over neighbouring siblings (10).

Plain-text fallback. When no DOM is available: (1) *boundary forcing* injects blank lines around known delimiters (code fences, residual `<pre>`/`<code>`, display-math delimiters, LaTeX environments, Markdown headings and rules, table rows, REPL prompts, indentation transitions) and

re-collapses runs; (2) *inline extraction with masking* physically extracts strongly delimited spans in priority order (fenced code, inline code, display math, inline math, residual tags), stores their SHA-256 and replaces them by placeholders, so that "mixed-type lines" cease to exist; (3) *blank-line segmentation* splits on `\n[\t\xa0]*\n+` (tab or non-breaking space between newlines); (4) *block classification* uses four aggregated line features (LaTeX density, code-keyword density, natural-language density, structural consistency) in a two-layer scheme: heuristic rules for clear cases, a light classifier for the rest.

LIFO restoration. Masked segments are replaced by sentinels—PUA characters or readable placeholders such as `__MATH_001__`—that travel with their container; Proposition 1 requires every prose stage, including the stage-8 grammar tools, to leave them intact, which we did not test separately; restoration proceeds in last-in-first-out order, so placeholders nested inside other placeholders resolve naturally (Proposition 1, proved in App. C.1). Extraction is inside-out: inner *content* entities are extracted before the *container* entities around them, so that a container is recognized without impurities. The order matters for genuine nested children: an explicit MathML or renderer node inside a `<pre><code>` block is a real child (literal `$`-delimited text inside code is never scanned), and if the code were extracted first the child would be swallowed with it and lose its own route; extracting the child first leaves its sentinel inside the code segment, and LIFO restoration puts it back.

Mathematics. In the DOM, MathML `<math>`, MathJax script types (`math/tex`, `math/mml`), renderer class whitelists (MathJax, `mjx-`, KaTeX, Wikipedia's `mwe-math-element`) and image formulas whose `alt`, `title`, `aria-label` or `data-latex` attributes carry TeX are strong signals. In plain text, LaTeX environments are atomic units, display math (`$$`, `\[`) and `\(` inline math are unambiguous, and `$`-delimited inline math is enabled only when (i) L1 finds `['$','$']` in the page's MathJax `inlineMath` configuration, (ii) Pandoc's anti-false-positive rules hold (no space after the opening or before the closing `$`, no digit after the closing one) and (iii) the span contains a known LaTeX command such as `\frac`, `\sum` or a sub/superscript. Math delimiters are never scanned inside `pre`, `code`, `script`, `style`, `textarea` or annotation elements (the `skipTags` convention of MathJax and KaTeX), which protects shell and Perl variables and JavaScript template strings. Ambiguous blocks pass a lightweight fastText math classifier and then, in series, OpenWebMath's MathScore (fast and high-recall, but prone to false positives on number-dense text such as financial tables) and the FineMath classifier (more precise, more expensive), so the cascade combines the recall of the first with the precision of the second.

Structured data. Acceptance is parser-based. A JSON block must be consumed by `json.JSONDecoder.raw_decode` as one complete value; a JSON literal inside a JavaScript or Python assignment is routed to code, because its container is code. YAML (ruamel.yaml), TOML (tomlkit) and INI (configparser) blocks must load with their headers or key–value structure; CSV/TSV blocks must show a consistent delimiter (csv.Sniffer, with CleverCSV as a fallback) over at least three rows with equal field counts; XML must parse with lxml (a recovered parse of malformed markup is accepted as structured data but is a certificate of structure, not of the original's validity); and runs of at least three `Key: Value` or `Key=Value` lines are treated as key–value data.

Code segments. Beyond DOM and delimiter signals, unhinted code (early web pages, PDF text) is detected by a three-tier cascade: a fastText code/natural-language classifier on fixed windows; a language-identification vote across Magika (Fratantonio et al., 2025), Guesslang and go-enry, fused into a per-language confidence vector; and Pygments' lexer guess as a weak confirmation. Language identification for the repairer fuses highlighter class hints, Magika (a 1 MB model covering over a hundred content types, with an average F1 of about 99% on common programming languages, chosen over extension-based tools because crawled snippets have no filename) and Tree-sitter error ratios, preceded by fast paths (shebang, server-side delimiters, modelines) and followed by keyword anchors (a language must show its signature keywords) and negative anchors (patterns that veto a language). A label is accepted when more than two of four sources (Magika, keyword anchors, Tree-sitter parsing, HTML hint) agree; otherwise it is marked low-confidence. An error analysis identifies six systematic confusions of statistical identifiers on very short blocks (Table 9); they are inherent to identifiers that see neither a full syntax tree nor enough context, and they motivate the specification's rule of sending blocks shorter than 50 characters, with their surrounding context, to an LLM rescue module

(this step was not run for the corpora of this paper). On 1,000 unhinted blocks the fused identifier reaches 98.3% accuracy; the 17 errors are inline-code fragments. This is a component evaluation of the identifier, not a training ablation: in the pre-training runs, identified code segments were protected and restored verbatim (§3.2).

Table 9: Systematic confusions of statistical language identifiers on very short code blocks, and their causes (from our error analysis). They motivate keyword and negative anchors and the LLM rescue of blocks shorter than 50 characters.

Short block (example)	Labelled as	Cause
single-line C macro (`#define BUFSIZE 1024`)	Bash	no braces, statements or type keywords; a `#`-prefixed line resembles a shell comment or assignment
tiny Java/Scala fragment (`class A {}`)	Forth	too few character trigrams; short lower-case tokens resemble Forth words
short Clipper line (`? "Hello", USE Customer`)	Bash	shell-like single-command syntax; Clipper is nearly absent from modern corpora
assembly or BIOS-interrupt lines (`mov ax, 0x0E`)	Python, Bash	short mnemonics and hex literals resemble assignments and arithmetic
constant-definition table (`MAX_LEN 100`)	awk	untyped key–value lines match awk's assignment patterns
all-comment block (`- ...`)	Haskell	comment-dense Haskell training data; runs of hyphens

Code quality gates. A language-agnostic layer computes 29 features designed with reference to OpenCoder's programming-language-independent filtering rules (Huang et al., 2025) and reorganized after probing the distribution of crawled code blocks, restricted to what is computable on a snippet without a file boundary, AST or repository context (size, line and word statistics, blank- and long-line fractions, unigram entropy, character-class fractions, hex-word and replacement-character fractions, long-token fraction, unique-word and single-word-line ratios, duplicate 10-gram and top 3-gram fractions, duplicate-line ratio, and flags for auto-generated, encoded/binary, XML-start, corrupted-encoding and licence-only blocks). A language-specific layer runs mature linters (pylint/ruff, PMD/Checkstyle/SpotBugs, ESLint, cppcheck/clang-tidy, golangci-lint, Roslyn analyzers, PHPStan/Psalm, RuboCop, and equivalents) for languages that satisfy two criteria: consistently high ranks in several of five independent popularity sources (TIOBE, the Stack Overflow survey, GitHub Octoverse, IEEE Spectrum, RedMonk)—R is included for its established role in statistics, econometrics and finance despite lower overall popularity—and a mature tool ecosystem (linters, a Tree-sitter grammar and public quality-labelled data). The supported languages are Python, Java, JavaScript/TypeScript, C/C++, C#, Go, PHP, Ruby, Rust, Shell, SQL and R; languages without such an ecosystem (e.g. Swift, Kotlin, Scala, Haskell) fall back to the language-agnostic layer. The gate returns pass/reject per block; a rejected repaired block is replaced by the original if the original passes the language-agnostic gate. Because the runs of this paper apply no typed repair, the gate is not exercised in the reported experiments; it is part of the framework specification and of the experiment code to be pinned for the camera-ready.

C.1 Proof of Proposition 1

Represent the detected segments as a finite forest whose edges are genuine containment; by assumption it is non-crossing. Extraction proceeds inside-out, so a segment is extracted only after all segments nested in it, and its stored payload may contain the sentinels of those children. Consider the reverse extraction sequence. The last extracted segment is outermost; its sentinel occurs in the skeleton exactly once and in its original position, because prose repair preserves the identity, multiplicity and order of sentinels and sentinels are unique and absent from the input (an assumption the pipeline supports by removing native private-use characters in stage 6 while sparing the sentinels it tracks). Replacing it by its stored, unmodified payload therefore reproduces the original bytes of that segment, including the sentinels of its children at their original positions. By induction over the reverse sequence, when a segment is reached its parent has already been restored, so its sentinel is present exactly once at the stored position and its replacement is exact. After the last step every untouched protected segment and all its descendants are restored byte for byte; the stored SHA-256 hashes provide an independent integrity check. □

The proposition concerns protected content after a correct detection decision. It does not cover a missed code span, a wrong parser boundary or an accepted type-specific repair of a payload (whose restored bytes are then those of the accepted repair), and it says nothing about the prose, which repair changes by design.

D THE 30-DEFECT SENSITIVITY BENCHMARK

The benchmark covers the anomaly types that the ten stages are designed to detect and repair; several injected variants (NFD text, curly quotes, ligatures, full-width characters, non-breaking spaces, private-use characters) are not intrinsically errors and count as anomalies only in the benchmark's English-prose setting, where they are artifacts of conversion (a limitation: the suite has no benign-transformation negative controls), each a pattern with a measurable incidence in real web corpora (Table 10). Each defect is injected into the same base texts and every candidate scorer scores both versions. A response is *sensitive-and-correct* if the score moves significantly toward lower quality after injection, *perverse* if it moves toward higher quality, and *insensitive* if it barely changes; the per-defect score is $100\,P_{\text{correct}} - 100\,P_{\text{perverse}} + 0 \cdot P_{\text{insensitive}}$, which ranges from -100 to 100. The perverse class is kept separate because it is the most serious failure: the scorer judges the corrupted text to be of *higher* quality.

Table 10: The 30 structural anomalies of the benchmark, by family, with typical forms.

Family	Defects	Typical forms
Character distribution (8)	mojibake; multiple encoding errors; mojibake with U+FFFD; inconsistent UTF-8 mojibake; undecomposed ligatures; NFD-form text; full-width characters; curly quotes	an apostrophe mis-decoded as â€™; clusters of U+FFFD; ligatures such as "fi" as one character; base letters separated from combining marks; full-width letters and digits; mixed curly and straight quotes
Layout (9)	hyphenated line breaks; intra-word line breaks; stray inter-word line breaks; blank-line runs; chaotic tabs/spaces; 0xA0 as space; mixed line endings; soft hyphens; form feeds	words split across lines with or without a hyphen; runs of blank lines; mixed `\r\n`, `\n`, `\r`; residual U+00AD; isolated U+000C
Control characters (7)	C1 control characters; lone surrogates; basic control characters; residual control characters; zero-width characters; BOM; private-use characters	U+0080–U+009F; unpaired high/low surrogates; C0 controls other than tab, newline and carriage return; zero-width space/non-joiner/joiner
Structural artifacts (6)	PDF extraction artifacts; HTML residue; encoding-artifact residue; boilerplate text; BBCode tags; navigation-bar boilerplate	leftover headers, footers and line numbers; tags and entities such as `<div>`; `&` that one cleaning pass cannot reduce; `[b]`, `[url]`; "Home | Products | Services | About us"

E SCORER BENCHMARK DETAILS

Dual KenLM. We instantiate the good/bad ensemble of Kim et al. (2025), in which the two models' perplexities are z-score standardized and combined into a relative quality score, with a "good" KenLM trained on high-quality text (Wikipedia) and a "bad" one trained on noisy web text such as OSCAR; Kim et al. (2025) themselves train on S2ORC and textbook data versus social-media, spam and toxic text. Beyond the "baseball" example of §4, whitespace defects (blank-line runs, chaotic tabs and spaces) are only re-arrangements of characters whose n-gram patterns are abundant and varied in the training text, so perturbing them does not raise perplexity. Used as the stage-9 validator, Dual KenLM would invert the validation logic for exactly the layout anomalies the pipeline targets.

Ensemble selection. The document-level pass/reject rule that combines the per-defect scores at stage 9 is part of the experiment code and is not reported here. The composite criterion is implemented in the experiment code, not in the public reference implementation (Reproducibility statement); it is not an unpenalized average of per-defect maxima (adding a tool can lower it, below), so the aggregates are reported as calibration scores on the benchmark, without a closed-form optimum

Table 11: Dual-KenLM sensitivity profile on representative defects: share of pairs scored sensitive-and-correct vs. perverse, and the net directional score ($P_{\text{correct}} - P_{\text{perverse}}$, in points). The four inverted rows are layout defects that stages 5–7 exist to repair.

Defect	correct (%)	perverse (%)	net
correct direction			
Mojibake	98.38	1.62	+97
Mojibake, heavy	99.82	0.18	+100
Hyphenated line breaks	87.69	12.31	+75
HTML residue	92.19	7.81	+84
Encoding-artifact residue	97.74	2.26	+95
Navigation-bar boilerplate	98.26	1.74	+97
inverted			
Intra-word line breaks	38.43	61.57	**−23**
Stray inter-word line breaks	29.47	70.53	**−41**
Blank-line runs	34.94	65.06	**−30**
Chaotic tabs/spaces	39.13	60.87	**−22**

Table 12: The 13 candidate scorers on the 30-defect benchmark: sensitivity total and the Pearson correlation between each tool's per-pair score difference and Gemini-as-Judge's quality ordering of the same pair. GPU-dependent tools are marked †. For reference, Dual KenLM's 30-defect mean score is +43.7. The last two rows give the composite score of the best combinations (all combinations of the 13 tools were scored exhaustively).

Rank	Scorer	Family	Sensitivity	Pearson
1	Typed heuristic scorer (ours)	rule-based	75.01	0.617
2	ftfy-diff	repair-difference	60.94	0.545
3	yelpfeas-byt5†	byte-level T5	60.86	0.553
4	NLTK	rule-based	56.94	0.457
5	sdadas-byt5†	byte-level T5	49.59	0.510
6	RoBERTa-CoLA†	Transformer	49.52	0.501
7	LanguageTool	rule-based	30.54	0.298
8	DCLM fastText	fastText	30.18	0.360
9	unicodedata	rule-based	30.11	0.260
10	textstat	rule-based	22.56	0.198
11	FineWeb Mixtral-Edu†	Transformer	20.10	0.266
12	FineWeb Nemotron-4-Edu†	Transformer	14.78	0.219
13	Dolma fastText	fastText	13.28	0.235
	Best CPU-only combination (6 tools, deployed)		composite **0.896**	
	Best overall (5 tools, incl. Mixtral-Edu)		composite 0.913	

claim or a held-out validation. All $2^{13} - 1$ non-empty combinations of the 13 tools were scored exhaustively with a composite criterion; Table 14 lists the ten best. The best overall combination (0.913) includes the GPU-dependent Mixtral-Edu classifier, and the best CPU-only combination (0.896) was implemented and re-measured end-to-end at 0.896, matching the prediction. The criterion is not monotone in the subset: adding `textstat` to the deployed six tools lowers it to 0.894, so a larger ensemble is not automatically better and the search rewards complementarity rather than size. Tools that score low on their own, such as `unicodedata` and LanguageTool, earn their place because they are uniquely strong on specific defects (C1 controls, BOM and lone surrogates; full-width characters).

The LLM judge is an external reference. Gemini-as-Judge serves as an external reference for the Pearson column: the reported coefficients measure agreement between each tool's per-pair score difference and the judge's quality ordering of the same pair.

Table 13: Defect assignment in the six-tool ensemble: each listed defect is scored exclusively by its pre-assigned best tool. The assignment covers 29 of the 30 defects; blank-line runs do not appear in it.

Tool	Defects owned
Typed heuristic scorer	mojibake, hyphenated line breaks, intra-word line breaks, chaotic tabs/spaces, inconsistent UTF-8 mojibake, encoding-artifact residue, zero-width characters, soft hyphens
NLTK	multiple encoding errors, mojibake with U+FFFD, undecomposed ligatures, mixed line endings, NFD-form text, private-use characters, form feeds
unicodedata	C1 control characters, lone surrogates, residual control characters, BOM
DCLM fastText	curly quotes, PDF extraction artifacts, boilerplate text, navigation-bar boilerplate
ftfy-diff	stray inter-word line breaks, 0xA0 as space, BBCode tags
LanguageTool	full-width characters, basic control characters, HTML residue

Table 14: The ten best tool combinations found by exhaustive search (composite score on the 30-defect benchmark). THS = typed heuristic scorer; † = GPU-dependent. The deployed ensemble is the best combination that runs on CPU alone.

Rank	Combination	Composite
1	Mixtral-Edu† + DCLM fastText + THS + ftfy-diff + NLTK	0.913
2	Nemotron-4-Edu† + DCLM fastText + THS + ftfy-diff + NLTK	0.908
3	**DCLM fastText + LanguageTool + THS + ftfy-diff + NLTK + unicodedata (deployed)**	**0.896**
4	LanguageTool + THS + ftfy-diff + textstat + NLTK + unicodedata	0.895
5	DCLM fastText + Dolma fastText + THS + ftfy-diff + textstat + NLTK	0.895
6	DCLM fastText + LanguageTool + THS + ftfy-diff + textstat + NLTK + unicodedata	0.894
7	Nemotron-4-Edu† + THS + ftfy-diff + NLTK	0.893
8	Mixtral-Edu† + THS + ftfy-diff + NLTK	0.893
9	THS + ftfy-diff + NLTK + RoBERTa-CoLA†	0.892
10	Mixtral-Edu† + THS + ftfy-diff + textstat + NLTK	0.892

F Data Construction and Training Details

Why Nemotron-CC v1. Two public versions exist (Table 15). We use v1 because it is openly hosted, keeps the WARC record ID and URL of every document (so documents can be traced back to their original Common Crawl files), covers the 2013–2015 snapshots we target, and offers five quality levels as a finer starting point.

Table 15: Nemotron-CC v1 (Su et al., 2025) vs. v2/v2.1.

Dimension	v1	v2 (v2.1)
Snapshots	2013-20 to 2024-30 (99 crawls)	v1 + eight 2024–2025 crawls (v2.1: further 2025 crawls)
Tokens	6.3T (4.4T real + 1.9T synthetic)	6.6T (v2.1: 2.5T additional tokens, released to be used together with v2)
Quality levels	5 (HQ, MHQ, MQ, MLQ, LQ)	3 (HQ, MHQ, MQ)
Hosting	Common Crawl S3, open	Hugging Face, gated
Schema	text + WARC record ID + URL	content + quality + uuid

Tiers and source texts. Documents were taken from several 2013–2015 global snapshots (e.g. CC-MAIN-2013-20, -2013-48, -2014-10, -2015-22) as partitioned by the Nemotron-CC v1 release, whose file paths encode the quality partition of every record (only the release's `kind=actual` partitions; its synthetic rephrasings were not used); the HQ, MQ and LQ groups are these partitions, unchanged, and we ran no quality classifier of our own. Because the released `text` field has already

passed NeMo Curator's extraction and normalization—which may have flattened tables, removed indentation or introduced encoding artifacts that repair is meant to address—we used the WARC record ID and URL that v1 preserves to retrieve each document's original text from the corresponding Common Crawl record; Raw is that retrieved text. We keep Nemotron-CC's own tiering rather than a new quality model because our stages are designed to be embedded as modules in NeMo Curator and because every claim must remain comparable with state-of-the-art open baselines; the inherited label describes the reference record, so the experiments compare repair recipes *within* a state-of-the-art quality stratum rather than against the released Nemotron-CC text.

Table 16: Prepared training pools per group (documents, tokens and the size of the tokenized training bin, from the run manifests). Every run then consumes exactly $16{,}406 \times 512 \times 4{,}096 = 34{,}405{,}875{,}712$ tokens in one epoch; the pools exceed that schedule by 0.1–0.4% and are down-sampled at document granularity, while V1 pools are up-sampled at bin level to the Raw token count (hence identical Raw and V1 token counts, and bin sizes equal to within 0.06 GB); V2 pools, which stage 8 can lengthen relative to V1, exceed Raw and are not up-sampled. Per-document multiplicities and packing were not recorded, so identical schedules establish identical token budgets, not identical unique-document exposure.

Group	Documents	Tokens (B)	Training bin (GB)
HQ-Raw	55,207,716	34.460	137.82
HQ-V1	55,619,461	34.460	137.82
HQ-V2	56,051,756	34.483	137.93
MQ-Raw	39,576,929	34.440	137.76
MQ-V1	39,836,563	34.440	137.70
MQ-V2	40,544,150	34.489	137.95
LQ-Raw	80,414,180	34.480	137.92
LQ-V1	80,772,480	34.480	137.92
LQ-V2	80,610,629	34.532	138.13

Table 17: Training hyperparameters, identical for all runs.

Field	Value
Architecture / tokenizer	Qwen3-1.7B-Base architecture (Yang et al., 2025), pre-trained from scratch (random initialization; no checkpoint loaded); byte-level BPE tokenizer with NFC normalization
Parameters	1,720,574,976
Precision	BF16 (no FP8)
Sequence length / global batch	4,096 / 512 sequences (2,097,152 tokens)
Parallelism	TP=1, PP=1, micro-batch 4, global batch $512 = 4 \times 8 \times 16$
Optimizer	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 0.1, clip 1.0
Schedule	WSD: 500 warmup + 13,506 stable + 2,400 cosine decay; peak 3×10^{-4}, min 3×10^{-5}
Iterations / tokens / epochs	16,406 / 34.4B / 1
Validation	every 1,000 iterations (50 evaluation iterations)
Checkpoints	every 4,000 iterations and at 16,406
Hardware	one node with eight NVIDIA A100 GPUs
Seeds	Raw baseline pre-trained with seeds 1234, 1337 and 42 (noise calibration); each V1 and V2 run with the training framework's default, seed 1234

Token-budget check. $512 \times 4{,}096 \times 16{,}406 = 3.441 \times 10^{10}$ tokens, i.e. 20 tokens per parameter for the 1.72B model. Each group is matched to this budget at document granularity (Table 16), so between-group differences are attributable to the repair-and-rematching recipe rather than to the token budget.

Training dynamics. On the Raw-HQ run, training loss starts near 11.2, falls to 5–6 during warmup, decays approximately log-linearly through the stable phase and converges to 2.52–2.55 during cosine decay (final ≈2.53); validation loss tracks it and ends near 2.61, with no sign of divergence or

over-fitting. The gradient norm starts near 12.5, settles below 0.5 in the stable phase and ends near 0.08, with no explosions or spikes. The run was briefly interrupted near iteration 7,500 by scheduled cluster maintenance and resumed from a checkpoint that restored model parameters, optimizer state and learning-rate scheduler state; loss and gradient statistics are continuous across the resume. The other runs show the same trends.

G Evaluation Suite

All models are evaluated on the same benchmark suite under one unified setup. Downstream tasks (27): LAMBADA-OpenAI (Paperno et al., 2016), BLiMP (Warstadt et al., 2020); SciQ (Welbl et al., 2017), ARC-Easy and ARC-Challenge (Clark et al., 2018), PubMedQA (Jin et al., 2019), MMLU (5-shot; Hendrycks et al., 2021); HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), CommonsenseQA (Talmor et al., 2019), SocialIQA (Sap et al., 2019), COPA (Roemmele et al., 2011), OpenBookQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2021); LogiQA (Liu et al., 2020), ANLI R1/R2/R3 (Nie et al., 2020), RTE (Dagan et al., 2006), CB (de Marneffe et al., 2019), BoolQ (Clark et al., 2019), TruthfulQA-MC2 (Lin et al., 2022); RACE (Lai et al., 2017), CoQA (Reddy et al., 2019), SQuAD v2 (Rajpurkar et al., 2018), TriviaQA (Joshi et al., 2017), NQ-Open (Kwiatkowski et al., 2019; Lee et al., 2019). Paloma (Magnusson et al., 2024) bits per byte on 11 corpora: Falcon-RefinedWeb, C4-en, C4-100-domains, mC4-en, Dolma-100-subreddits, Dolma-v1.5, RedPajama, M2D2-Wikipedia, M2D2-S2ORC, WikiText-103 (Merity et al., 2017) and Penn Treebank. For the analysis we group them (our grouping) into web/social crawls (the first five), mixed-source corpora that combine web and curated sources (Dolma-v1.5, Soldaini et al., 2024; RedPajama, Weber et al., 2024), and curated corpora (M2D2-Wikipedia, M2D2-S2ORC, WikiText-103, Penn Treebank); corpus descriptions follow Magnusson et al. (2024). The grouping is introduced in this paper for the analysis; mC4-en is the exception in the web group, since, like the curated corpora, it worsens slightly under V1 on every tier (Table 23–27).

H Full Per-Task Results

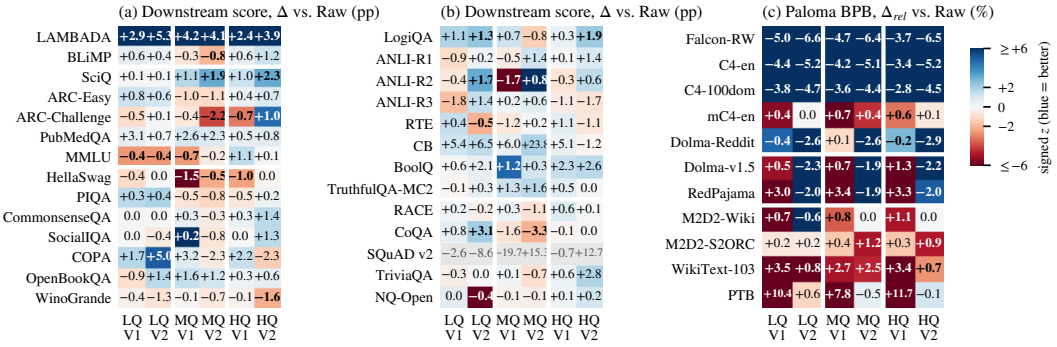


Figure 4: All 38 evaluations for the six repair×tier configurations (surface repair V1 and full repair V2 on each tier). Cell text: change vs. Raw (tasks in pp; Paloma in relative BPB, negative is better). Colour: signed z (blue = better, clipped at ± 6); bold: $|z| \geq 2$. SQuAD v2 (grey) is recorded on an undecoded scale.

Tables 22–27 report every evaluation for every configuration with the noise band, at the precision at which it was recorded (two decimals for most MQ entries). Raw means are averaged over three seeds and rounded; Δ and z are computed from unrounded values, so Δ may differ from the difference of the displayed means by up to 0.1. Table 21 repeats the verdict tally along the threshold ladder of §5 ($|z| \geq 2$, 2.31, 3.71 and 5) and lists the eight verdicts that lie between 2 and 2.31, all on V2. SQuAD v2 is recorded by the evaluation script on a scale above 100 (165–270 here) whose definition we have not recovered, so its changes are reported as recorded and their direction is not interpreted; it never reaches $|z| \geq 2$ in either direction (largest $|z| = 1.12$, MQ-V1), contributes no verdict at any threshold whichever direction applies, and is excluded from the percentage-point means, whose unit it does not share. Repair coverage (Table 20) counts documents with at least one logged text-repair operation of the classes listed there, whereas the quality-shift sample of 10,000

Table 18: Surface (V1) and full (V2) repair against Raw. Robust = $|z| \geq 2$ over 27 downstream tasks and 11 Paloma corpora (SQuAD v2 never reaches it); mean Δ over the 26 tasks scored in pp and over Paloma corpora; web = five web/social crawls, mixed = Dolma-v1.5 and RedPajama, curated = M2D2-Wikipedia, M2D2-S2ORC, WikiText-103 and PTB (our grouping; App. H).

		Downstream robust		Paloma robust			mean Δ	mean Paloma Δ_{rel} (%)			
Tier	Repair	wins	losses	wins	losses	net	tasks (pp)	all 11	web	mixed	curated
HQ	V1 surface	1	2	4	6	−3	+0.60	+1.05	−1.91	+2.31	+4.13
	V2 + linguistic	**4**	1	**6**	2	**+7**	+0.58	**−1.96**	−3.79	−2.07	+0.38
MQ	V1 surface	3	3	3	6	−3	+0.51	+0.37	−2.34	+2.03	+2.94
	V2 + linguistic	3	4	**6**	3	**+2**	+0.87	**−1.70**	−3.61	−1.93	+0.80
LQ	V1 surface	1	1	4	6	−2	+0.46	+0.47	−2.65	+1.77	+3.72
	V2 + linguistic	**5**	3	**7**	1	**+8**	+1.05	**−2.02**	−3.82	−2.14	+0.27

Table 19: Anatomy of the effect against Raw, counting every evaluation regardless of significance (ties omitted). Signal tasks: the 9 tasks whose Raw score is at least 10 pp above random choice on every tier (Table 28); W/L: robust wins/losses. Gain/loss: summed positive and negative changes.

		37 evaluations	26 tasks (pp)	9 signal tasks		11 Paloma (% BPB)
Tier	Repair	better / worse	gain / loss	better / worse	W / L	gain / loss
HQ	V1 surface	22 / 14	19.5 / 3.8	6 / 3	1 / 1	10.1 / 21.7
	V2 + linguistic	**25** / 8	23.1 / 7.9	6 / 1	2 / 0	23.3 / 1.8
MQ	V1 surface	17 / 20	22.8 / 9.6	4 / 5	1 / 1	12.5 / 16.6
	V2 + linguistic	**19** / 18	38.2 / 15.6	3 / 6	2 / 3	22.8 / 4.1
LQ	V1 surface	17 / 17	18.0 / 6.1	8 / 1	1 / 0	13.7 / 18.9
	V2 + linguistic	**24** / 9	30.6 / 3.2	8 / 0	3 / 0	23.9 / 1.6

documents was drawn from the whole V2 output, edited and unedited alike. A deterministic scorer cannot change on an unchanged document, so a share of rising scores (95.0–97.4%) above coverage (56.5–72.6%) implies that some transformation applied to most documents is not counted by the coverage counter; `ftfy` repair is counted (Table 20), so the candidates are stage-1 decoding, stage-7 whitespace normalization, or a narrower counter than the operation log. We have not recovered the counter's definition or the paired per-document records that would settle it; the two statistics are therefore reported as recorded, side by side, and the quality shift is treated as descriptive rather than as part of the evidential argument. Figure 5 shows the Paloma changes by corpus group.

Table 20: Repair coverage and quality shift per tier under V2. Repair coverage: the pipeline's per-tier repair-coverage statistic under V2 (text-repair operations only; the extraction, protection and restoration of code, mathematics and similar segments are not counted as repair operations). Operation shares: shares of the five main text-repair operation classes, as reported; the classes listed are the main ones and do not sum to 100%. Quality shift: share of 10,000 documents sampled from the whole V2 output (edited and unedited alike) whose score increased relative to Raw ($\Delta > 0$) and mean change, under DCLM fastText (HQ, MQ) and an LLM judge (LQ; mean 8.92→15.22). The share exceeds coverage, which implies that some transformation applied to most documents is not counted by the coverage counter; we have not recovered that counter's definition, so the two statistics are reported side by side, as recorded, and not combined (App. H). Retained: documents and tokens of the V2 training pool.

	Coverage & quality shift			Operation shares under V2 (%)					Retained (V2)	
Tier	coverage	$\Delta > 0$	mean Δ	ftfy	artifact	line	char.	gram./synt.	docs (M)	tokens (B)
HQ	72.63%	97.4%	+0.053	15.70	9.45	5.67	5.69	12.02	56.1	34.48
MQ	57.82%	95.0%	+0.006	26.09	15.76	15.48	7.96	26.38	40.5	34.49
LQ	56.52%	95.0%	+6.30	17.61	16.46	12.69	11.27	37.23	80.6	34.53

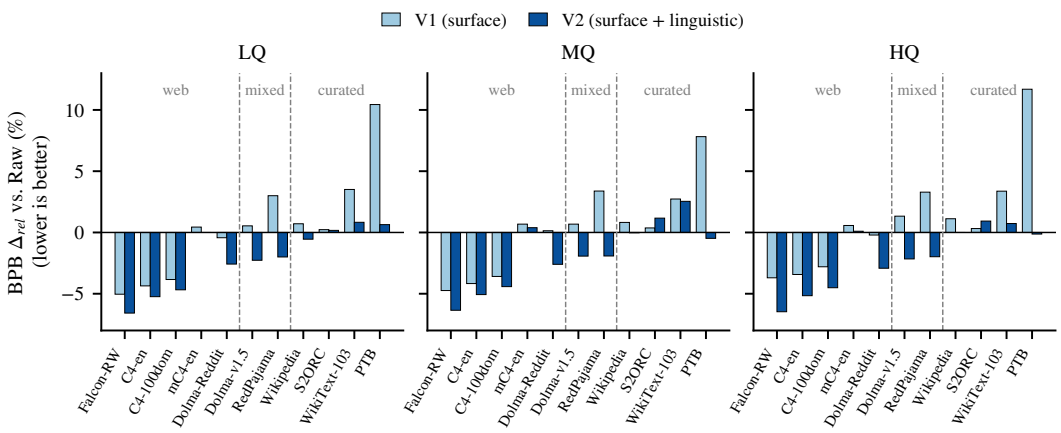


Figure 5: Paloma relative BPB change vs. Raw for V1 and V2 on each tier, grouped (our grouping) into web/social crawls, mixed-source corpora and curated corpora (dashed lines). Surface repair alone improves web fit at a cost on mixed-source and curated text; adding linguistic repair improves web and mixed-source corpora and removes most of the cost on curated corpora.

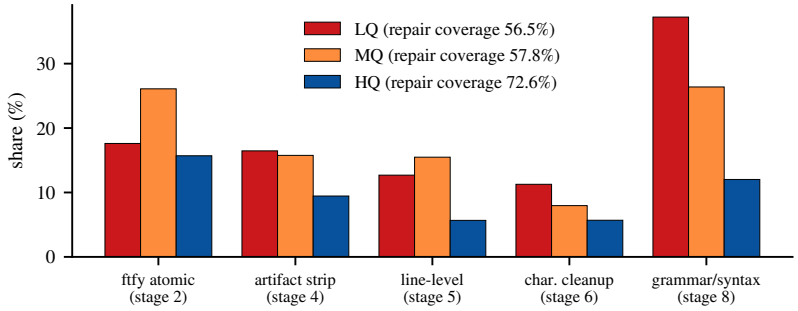


Figure 6: Shares of the main text-repair operations under V2, by stage and tier (Table 20). Grammar/syntax repair has the largest share on LQ (37.23%); on HQ `ftfy` repair leads (15.70%), followed by grammar/syntax (12.02%) and artifact removal (9.45%).

Table 21: Sensitivity of the verdict tallies to the counting rule (§5). Thresholds: $|z| \geq 2$ (the protocol), $2\sqrt{1+1/3} \approx 2.31$ (two standard errors when one treatment run is compared with a three-seed mean of equal variance), 3.71 (two-sided Bonferroni over the 38 evaluations, 3.21, scaled by the same $\sqrt{4/3}$) and 5 (above the two-degree-of-freedom t cutoff 4.30 scaled the same way, 4.97); these are descriptive sensitivity cutoffs, not calibrated tests. Floor: $|z| \geq 2$ with σ_{tot} floored at σ_{binom} for every multiple-choice task, which removes the near-degenerate denominators (e.g. SocialIQA on MQ-V1). DS: the 27 downstream tasks (SQuAD v2 never reaches $|z| \geq 2$); Pal.: the 11 Paloma corpora; W/L = wins/losses. V2 is net-positive and V1 is never net-positive on every tier under every rule. Verdicts with $2 \leq |z| < 2.31$ (all V2; no V1 or Paloma verdict lies there): HQ-V2: WinoGrande (L, -2.2), LogiQA (W, $+2.2$); MQ-V2: HellaSwag (L, -2.24), CoQA (L, -2.07), BLiMP (L, -2.0); LQ-V2: LogiQA (W, $+2.3$), CoQA (W, $+2.1$), RTE (L, -2.1).

		$\lvert z\rvert \geq 2$			$\lvert z\rvert \geq 2.31$			$\lvert z\rvert \geq 3.71$			$\lvert z\rvert \geq 5$			Floor		
Tier	Repair	DS	Pal.	net	DS	Pal.	net	DS	Pal.	net	DS	Pal.	net	DS	Pal.	net
HQ	V1	1/2	4/6	−3	1/2	4/6	−3	1/0	3/5	−1	1/0	3/4	0	1/1	4/6	−2
	V2	4/1	6/2	+7	3/0	6/2	+7	2/0	6/1	+7	1/0	6/0	+7	1/0	6/2	+5
MQ	V1	3/3	3/6	−3	3/3	3/6	−3	3/2	3/5	−1	2/2	3/4	−1	1/2	3/6	−4
	V2	3/4	6/3	+2	3/1	6/3	+5	2/1	6/3	+4	2/0	6/0	+8	1/2	6/3	+2
LQ	V1	1/1	4/6	−2	1/1	4/6	−2	1/0	4/6	−1	1/0	3/6	−2	1/0	4/6	−1
	V2	5/3	7/1	+8	3/2	7/1	+7	2/1	7/1	+7	2/1	7/1	+7	2/1	7/1	+7

Table 22: Full downstream results on the high-quality (HQ) tier: Raw (mean over three pre-training seeds) vs. V1 (surface repair) and V2 (surface + linguistic repair). σ is the noise band σ_{tot} of §5 for that arm (its components $\sigma_{\mathrm{pretrain}}$, σ_{eval} and σ_{binom} are in the supplement's `data/results.py`); $z = \Delta/\sigma$; bold: $|z| \geq 2$. SQuAD v2 values are reproduced as recorded by the evaluation script; their scale (above 100) is undecoded and its direction is not interpreted, so they are excluded from the percentage-point means and contribute no verdict.

		V1 (surface)				V2 (surface + linguistic)			
Task	Raw	σ	mean	Δ (pp)	z	σ	mean	Δ (pp)	z
Language modelling									
LAMBADA	39.6	0.311	41.9	+2.4	**+7.6**	0.311	43.5	+3.9	**+12.6**
BLiMP	81.3	0.894	81.8	+0.6	+0.6	0.894	82.5	+1.2	+1.3
Science / knowledge									
SciQ	79.0	0.764	80.0	+1.0	+1.3	0.764	81.3	+2.3	**+3.0**
ARC-Easy	59.6	0.973	60.0	+0.4	+0.4	0.973	60.4	+0.7	+0.8
ARC-Challenge	30.9	0.215	30.2	−0.7	**−3.1**	0.215	31.9	+1.0	**+4.6**
PubMedQA	54.2	1.186	54.7	+0.5	+0.5	1.186	55.0	+0.8	+0.7
MMLU	25.2	0.716	26.2	+1.1	+1.5	0.716	25.2	+0.1	+0.1
Commonsense									
HellaSwag	49.3	0.398	48.3	−1.0	**−2.4**	0.398	49.3	−0.0	−0.1
PIQA	70.2	0.649	69.7	−0.5	−0.7	0.649	70.4	+0.2	+0.3
CommonsenseQA	19.8	0.855	20.1	+0.3	+0.4	0.855	21.2	+1.4	+1.6
SocialIQA	40.5	0.795	40.6	+0.0	+0.0	0.795	41.9	+1.3	+1.6
COPA	72.3	1.528	74.5	+2.2	+1.4	1.528	70.0	−2.3	−1.5
OpenBookQA	34.6	0.918	34.9	+0.3	+0.3	0.911	35.2	+0.6	+0.6
WinoGrande	53.4	0.721	53.3	−0.1	−0.2	0.721	51.9	−1.6	**−2.2**
Logic / NLI									
LogiQA	21.1	0.900	21.4	+0.3	+0.4	0.900	23.0	+1.9	**+2.2**
ANLI-R1	31.8	1.480	31.9	+0.1	+0.1	1.480	33.2	+1.4	+0.9
ANLI-R2	33.8	0.379	33.5	−0.3	−0.8	0.379	34.4	+0.6	+1.5
ANLI-R3	34.8	2.014	33.8	−1.1	−0.5	2.014	33.1	−1.7	−0.8
RTE	49.5	1.532	50.5	+1.1	+0.7	1.532	48.4	−1.1	−0.7
CB	49.4	11.619	54.5	+5.1	+0.4	11.619	48.2	−1.2	−0.1
BoolQ	57.4	1.647	59.7	+2.3	+1.4	1.647	59.9	+2.6	+1.6
TruthfulQA-MC2	37.9	0.704	38.4	+0.5	+0.7	0.704	38.0	+0.0	+0.1
Reading / open QA									
RACE	31.3	0.829	31.9	+0.6	+0.8	0.829	31.4	+0.1	+0.1
CoQA	23.2	0.854	23.1	−0.1	−0.1	1.530	23.2	−0.0	−0.0
SQuAD v2	256.9	29.608	256.2	−0.7	−0.0	26.842	269.5	+12.7	+0.5
TriviaQA	5.6	1.525	6.2	+0.6	+0.4	1.525	8.3	+2.8	+1.8
NQ-Open	3.2	0.215	3.3	+0.1	+0.3	0.215	3.5	+0.2	+1.1
Robust wins / losses ($\lvert z\rvert \geq 2$)			1 / 2				4 / 1		
Mean Δ over 26 accuracy tasks (pp)			+0.60				+0.58		

Table 23: Full Paloma bits-per-byte (BPB) results on the high-quality (HQ) tier (lower is better). Δ is in BPB, Δ_{rel} in %; $z = \Delta/\sigma_{\text{tot}}$ with $\sigma_{\text{tot}} = \sigma_{\text{pretrain}}$ (no binomial fallback for continuous metrics). Bold: $|z| \geq 2$; negative z is an improvement.

			V1 (surface)			V2 (surface + linguistic)		
Paloma corpus	Raw BPB	σ_{tot}	BPB	Δ_{rel}	z	BPB	Δ_{rel}	z
Falcon-RefinedWeb	1.0774	0.0011	1.0376	−3.70%	**−36.4**	1.0077	−6.47%	**−63.6**
C4-en	1.0196	0.0011	0.9847	−3.43%	**−30.8**	0.9670	−5.16%	**−46.3**
C4-100-domains	0.9690	0.0015	0.9418	−2.80%	**−18.6**	0.9253	−4.51%	**−30.0**
mC4-en	1.0011	0.0018	1.0068	+0.57%	**+3.2**	1.0021	+0.10%	+0.6
Dolma-100-subreddits	1.1223	0.0009	1.1199	−0.21%	**−2.7**	1.0895	−2.92%	**−36.9**
Dolma-v1.5	0.9828	0.0020	0.9959	+1.33%	**+6.4**	0.9616	−2.16%	**−10.5**
RedPajama	1.0265	0.0040	1.0603	+3.29%	**+8.4**	1.0062	−1.98%	**−5.0**
M2D2-Wikipedia	0.8894	0.0020	0.8993	+1.12%	**+4.9**	0.8894	+0.00%	+0.0
M2D2-S2ORC	0.9160	0.0018	0.9189	+0.32%	+1.6	0.9245	+0.93%	**+4.8**
WikiText-103	0.8665	0.0023	0.8957	+3.37%	**+12.6**	0.8728	+0.73%	**+2.7**
Penn Treebank	1.0378	0.0012	1.1591	+11.69%	**+101.9**	1.0364	−0.13%	−1.1
Robust wins / losses ($\|z\| \geq 2$)				4 / 6			6 / 2	
Mean Δ_{rel} over 11 corpora				+1.05%			−1.96%	

Table 24: Full downstream results on the medium-quality (MQ) tier: Raw (mean over three pre-training seeds) vs. V1 (surface repair) and V2 (surface + linguistic repair). σ is the noise band σ_{tot} of §5 for that arm (its components σ_{pretrain}, σ_{eval} and σ_{binom} are in the supplement's `data/results.py`); $z = \Delta/\sigma$; bold: $|z| \geq 2$. SQuAD v2 values are reproduced as recorded by the evaluation script; their scale (above 100) is undecoded and its direction is not interpreted, so they are excluded from the percentage-point means and contribute no verdict.

		V1 (surface)				V2 (surface + linguistic)			
Task	Raw	σ	mean	Δ (pp)	z	σ	mean	Δ (pp)	z
Language modelling									
LAMBADA	45.23	0.465	49.40	+4.17	**+8.96**	0.465	49.31	+4.08	**+8.78**
BLiMP	82.04	0.387	81.76	−0.27	−0.71	0.387	81.26	−0.77	**−2.0**
Science / knowledge									
SciQ	72.20	0.627	73.27	+1.07	+1.7	0.624	74.10	+1.90	**+3.04**
ARC-Easy	47.42	0.947	46.40	−1.02	−1.07	0.947	46.34	−1.08	−1.14
ARC-Challenge	26.17	0.549	25.81	−0.36	−0.65	0.549	23.98	−2.19	**−3.99**
PubMedQA	57.73	2.603	60.30	+2.57	+0.99	2.603	60.00	+2.27	+0.87
MMLU	24.20	0.291	23.46	−0.74	**−2.55**	0.291	24.04	−0.16	−0.55
Commonsense									
HellaSwag	50.54	0.201	49.05	−1.49	**−7.41**	0.201	50.09	−0.45	**−2.24**
PIQA	71.62	0.636	71.13	−0.49	−0.77	0.635	70.78	−0.83	−1.31
CommonsenseQA	20.42	0.557	20.72	+0.30	+0.54	0.557	20.15	−0.27	−0.49
SocialIQA	40.69	0.021	40.86	+0.17	**+8.16**	1.110	39.87	−0.82	−0.74
COPA	70.33	4.163	73.50	+3.17	+0.76	4.163	68.00	−2.33	−0.56
OpenBookQA	31.00	1.114	32.60	+1.60	+1.44	1.114	32.20	+1.20	+1.08
WinoGrande	54.82	1.097	54.70	−0.12	−0.11	1.097	54.14	−0.68	−0.62
Logic / NLI									
LogiQA	19.87	0.582	20.58	+0.72	+1.23	0.582	19.05	−0.82	−1.41
ANLI-R1	32.57	1.137	32.05	−0.52	−0.45	1.137	34.00	+1.43	+1.26
ANLI-R2	33.37	0.115	31.70	−1.67	**−14.43**	0.115	34.20	+0.83	**+7.22**
ANLI-R3	34.69	0.747	34.88	+0.18	+0.24	0.747	35.25	+0.56	+0.74
RTE	55.35	1.988	54.15	−1.20	−0.61	1.988	55.60	+0.24	+0.12
CB	26.19	12.918	32.14	+5.95	+0.46	12.918	50.00	+23.81	+1.84
BoolQ	57.37	0.250	58.56	+1.19	**+4.76**	0.250	57.65	+0.28	+1.1
TruthfulQA-MC2	37.86	0.999	39.13	+1.27	+1.27	0.999	39.47	+1.61	+1.61
Reading / open QA									
RACE	33.59	1.199	33.92	+0.33	+0.28	1.199	32.54	−1.05	−0.88
CoQA	25.95	1.809	24.35	−1.60	−0.89	1.609	22.63	−3.33	**−2.07**
SQuAD v2	251.22	17.482	231.56	−19.65	−1.12	17.963	266.53	+15.31	+0.85
TriviaQA	3.53	0.694	3.65	+0.12	+0.17	0.694	2.83	−0.70	−1.01
NQ-Open	1.99	0.363	1.88	−0.11	−0.3	0.363	1.86	−0.14	−0.38
Robust wins / losses ($\|z\| \geq 2$)			3 / 3				3 / 4		
Mean Δ over 26 accuracy tasks (pp)			+0.51				+0.87		

Table 25: Full Paloma bits-per-byte (BPB) results on the medium-quality (MQ) tier (lower is better). Δ is in BPB, Δ_{rel} in %; $z = \Delta/\sigma_{\text{tot}}$ with $\sigma_{\text{tot}} = \sigma_{\text{pretrain}}$ (no binomial fallback for continuous metrics). Bold: $|z| \geq 2$; negative z is an improvement.

			V1 (surface)			V2 (surface + linguistic)		
Paloma corpus	Raw BPB	σ_{tot}	BPB	Δ_{rel}	z	BPB	Δ_{rel}	z
Falcon-RefinedWeb	1.0274	0.0015	0.9787	−4.74%	**−33.21**	0.9622	−6.35%	**−44.51**
C4-en	0.9766	0.0012	0.9359	−4.17%	**−33.45**	0.9271	−5.07%	**−40.64**
C4-100-domains	0.9125	0.0007	0.8797	−3.59%	**−47.58**	0.8721	−4.42%	**−58.65**
mC4-en	0.9661	0.0009	0.9727	+0.68%	**+7.74**	0.9698	+0.39%	**+4.39**
Dolma-100-subreddits	1.1102	0.0013	1.1117	+0.14%	+1.16	1.0813	−2.60%	**−21.76**
Dolma-v1.5	0.9695	0.0010	0.9761	+0.68%	**+6.6**	0.9507	−1.93%	**−18.7**
RedPajama	1.0273	0.0010	1.0620	+3.38%	**+33.8**	1.0076	−1.92%	**−19.18**
M2D2-Wikipedia	0.8740	0.0020	0.8812	+0.82%	**+3.65**	0.8737	−0.03%	−0.14
M2D2-S2ORC	0.9145	0.0022	0.9179	+0.37%	+1.57	0.9252	+1.17%	**+4.91**
WikiText-103	0.8489	0.0047	0.8720	+2.73%	**+4.93**	0.8704	+2.54%	**+4.59**
Penn Treebank	0.9813	0.0086	1.0581	+7.82%	**+8.94**	0.9765	−0.48%	−0.55
Robust wins / losses ($\|z\| \geq 2$)				3 / 6			6 / 3	
Mean Δ_{rel} over 11 corpora				+0.37%			−1.70%	

Table 26: Full downstream results on the low-quality (LQ) tier: Raw (mean over three pre-training seeds) vs. V1 (surface repair) and V2 (surface + linguistic repair). σ is the noise band σ_{tot} of §5 for that arm (its components σ_{pretrain}, σ_{eval} and σ_{binom} are in the supplement's `data/results.py`); $z = \Delta/\sigma$; bold: $|z| \geq 2$. SQuAD v2 values are reproduced as recorded by the evaluation script; their scale (above 100) is undecoded and its direction is not interpreted, so they are excluded from the percentage-point means and contribute no verdict.

		V1 (surface)				V2 (surface + linguistic)			
Task	Raw	σ	mean	Δ (pp)	z	σ	mean	Δ (pp)	z
Language modelling									
LAMBADA	35.7	0.256	38.6	+2.9	**+11.1**	0.256	41.0	+5.3	**+20.6**
BLiMP	78.4	0.885	78.9	+0.6	+0.6	0.885	78.7	+0.4	+0.4
Science / knowledge									
SciQ	65.6	2.248	65.7	+0.1	+0.1	2.248	65.7	+0.1	+0.1
ARC-Easy	35.3	0.611	36.1	+0.8	+1.3	0.611	35.9	+0.6	+1.0
ARC-Challenge	23.1	0.421	22.7	−0.5	−1.1	0.421	23.2	+0.1	+0.1
PubMedQA	49.5	5.163	52.6	+3.1	+0.6	5.163	50.2	+0.7	+0.1
MMLU	23.3	0.144	22.9	−0.4	**−2.6**	0.144	22.9	−0.4	**−2.6**
Commonsense									
HellaSwag	44.0	0.318	43.6	−0.4	−1.2	0.318	44.0	+0.0	+0.0
PIQA	69.2	0.196	69.5	+0.3	+1.6	0.196	69.6	+0.4	+1.9
CommonsenseQA	19.6	0.189	19.6	+0.0	+0.1	0.189	19.6	+0.0	+0.1
SocialIQA	38.8	0.582	38.8	+0.0	+0.1	0.581	38.4	−0.4	−0.6
COPA	68.0	1.000	69.7	+1.7	+1.7	1.000	73.0	+5.0	**+5.0**
OpenBookQA	27.6	0.872	26.7	−0.9	−1.0	0.872	29.0	+1.4	+1.6
WinoGrande	53.2	1.539	52.9	−0.4	−0.3	1.540	52.0	−1.3	−0.8
Logic / NLI									
LogiQA	21.7	0.585	22.8	+1.1	+1.8	0.582	23.0	+1.3	**+2.3**
ANLI-R1	33.2	0.819	32.3	−0.9	−1.1	0.819	33.4	+0.2	+0.2
ANLI-R2	33.4	0.551	33.0	−0.4	−0.7	0.551	35.1	+1.7	**+3.0**
ANLI-R3	35.3	0.966	33.5	−1.8	−1.8	0.966	36.7	+1.4	+1.4
RTE	52.9	0.255	53.2	+0.4	+1.4	0.255	52.4	−0.5	**−2.1**
CB	41.7	4.494	47.0	+5.4	+1.2	4.494	48.2	+6.5	+1.5
BoolQ	55.7	4.145	56.3	+0.6	+0.1	4.145	57.8	+2.1	+0.5
TruthfulQA-MC2	40.9	0.674	40.9	−0.1	−0.1	0.674	41.2	+0.3	+0.5
Reading / open QA									
RACE	29.3	0.722	29.4	+0.2	+0.2	0.722	29.1	−0.2	−0.3
CoQA	15.6	1.263	16.4	+0.8	+0.6	1.449	18.7	+3.1	**+2.1**
SQuAD v2	173.3	25.860	170.7	−2.6	−0.1	25.600	164.8	−8.6	−0.3
TriviaQA	1.4	0.753	1.1	−0.3	−0.4	0.753	1.4	+0.0	+0.0
NQ-Open	0.8	0.039	0.8	+0.0	+0.7	0.039	0.3	−0.4	**−11.3**
Robust wins / losses ($\lvert z\rvert \geq 2$)			1 / 1				5 / 3		
Mean Δ over 26 accuracy tasks (pp)			+0.46				+1.05		

Table 27: Full Paloma bits-per-byte (BPB) results on the low-quality (LQ) tier (lower is better). Δ is in BPB, Δ_{rel} in %; $z = \Delta/\sigma_{\text{tot}}$ with $\sigma_{\text{tot}} = \sigma_{\text{pretrain}}$ (no binomial fallback for continuous metrics). Bold: $|z| \geq 2$; negative z is an improvement.

			V1 (surface)			V2 (surface + linguistic)		
Paloma corpus	Raw BPB	σ_{tot}	BPB	Δ_{rel}	z	BPB	Δ_{rel}	z
Falcon-RefinedWeb	1.0887	0.0011	1.0339	−5.04%	**−51.1**	1.0171	−6.58%	**−66.8**
C4-en	1.0350	0.0009	0.9899	−4.36%	**−47.5**	0.9808	−5.24%	**−57.1**
C4-100-domains	0.9835	0.0008	0.9457	−3.84%	**−46.8**	0.9375	−4.68%	**−57.0**
mC4-en	1.0309	0.0008	1.0354	+0.44%	**+5.4**	1.0309	−0.00%	−0.0
Dolma-100-subreddits	1.1786	0.0010	1.1735	−0.43%	**−4.8**	1.1481	−2.58%	**−29.4**
Dolma-v1.5	1.1063	0.0011	1.1122	+0.54%	**+5.2**	1.0812	−2.27%	**−22.2**
RedPajama	1.1764	0.0020	1.2117	+3.00%	**+18.0**	1.1529	−2.00%	**−12.0**
M2D2-Wikipedia	0.9952	0.0005	1.0023	+0.71%	**+15.8**	0.9897	−0.55%	**−12.2**
M2D2-S2ORC	1.1722	0.0038	1.1749	+0.23%	+0.7	1.1741	+0.17%	+0.5
WikiText-103	0.9694	0.0010	1.0033	+3.51%	**+35.3**	0.9774	+0.83%	**+8.4**
Penn Treebank	1.0552	0.0053	1.1654	+10.44%	**+20.9**	1.0620	+0.64%	+1.3
Robust wins / losses ($\|z\| \geq 2$)				4 / 6			7 / 1	
Mean Δ_{rel} over 11 corpora				+0.47%			−2.02%	

Table 28: Classification of the 26 downstream tasks used in Table 19. Random choice is $1/k$ for k answer options; the Raw range is over the three tiers. A task has *signal* at this scale if its Raw score is at least 10 pp above random choice on every tier; LAMBADA and CoQA are open-ended and far above zero; TriviaQA and NQ-Open are below 6% exact match, and TruthfulQA-MC2 has no simple random baseline.

Task	random choice (%)	Raw range (%)	class
LAMBADA	–	35.7–45.2	signal
BLiMP	50.0	78.4–82.0	signal
SciQ	25.0	65.6–79.0	signal
ARC-Easy	25.0	35.3–59.6	signal
ARC-Challenge	25.0	23.1–30.9	near chance
PubMedQA	33.3	49.5–57.7	signal
MMLU	25.0	23.3–25.2	near chance
HellaSwag	25.0	44.0–50.5	signal
PIQA	50.0	69.2–71.6	signal
CommonsenseQA	20.0	19.6–20.4	near chance
SocialIQA	33.3	38.8–40.7	near chance
COPA	50.0	68.0–72.3	signal
OpenBookQA	25.0	27.6–34.6	near chance
WinoGrande	50.0	53.2–54.8	near chance
LogiQA	25.0	19.9–21.7	near chance
ANLI-R1	33.3	31.8–33.2	near chance
ANLI-R2	33.3	33.4–33.8	near chance
ANLI-R3	33.3	34.7–35.3	near chance
RTE	50.0	49.5–55.4	near chance
CB	33.3	26.2–49.4	near chance
BoolQ	50.0	55.7–57.4	near chance
TruthfulQA-MC2	–	37.9–40.9	no baseline
RACE	25.0	29.3–33.6	near chance
CoQA	–	15.6–25.9	signal
TriviaQA	–	1.4–5.6	floor
NQ-Open	–	0.8–3.2	floor

I Early-Web Anomaly Classes and the Stages That Recover Them

Nemotron-CC's snapshots begin with CC-MAIN-2013-20 (Table 15), and web text from the 1990s to the early 2010s is structurally degraded in ways that pipelines built for relatively well-formed modern pages were not designed for. The ten stages nevertheless configure a dedicated detector and repairer for each typical anomaly class of early web text (Table 29), so that every class is covered by at least one stage. Our 2013–2015 data exercise these stages to the extent that such anomalies occur in them.

J Examples of Structural Anomalies

Encoding. `fix_encoding` restores "cafÃ©" to "café"; a character encoded in UTF-8, decoded as Latin-1 and then treated as UTF-8 again yields the doubly decoded form of Table 29, which the stage-2 convergence loop unwinds step by step.

Hyphenated wraps. Typical forms are "finan-\nziellen" and "long-\nterm". Stage 5 deletes the hyphen and the line break when its three conditions hold and the concatenation is a dictionary word; for a compound such as "state-\nof-the-art", whose first hyphen belongs to the word, the concatenation is not a dictionary word and the original hyphen is kept.

Hard wraps. In a plain-text e-mail from 2000 formatted for a fixed-width terminal, the sentence "I'd volunteer to write a elisp wrapper around the common "md5sum" binary if you think it's a good idea…" is broken after `"md5sum"`, splitting one sentence across two lines; stage 5 re-joins such lines unless a blank line, list item or heading intervenes.

Table 29: Early-web structural anomalies, their typical period (indicative; these defects also occur outside the listed years) and cause, and the stage that repairs them.

Anomaly	Typical form	Period	Cause	Stage
Multiple non-Unicode encodings	legacy encodings mixed (ISO-8859-x, Shift-JIS, GB2312, Big5)	before 2005	inconsistent, unreliable declarations before Unicode	1
Latin-1 / Windows-1252 ambiguity	smart quotes and similar characters decoded as C1 controls (U+0080–U+009F)	1995–2010	overlapping byte ranges, mismatched browser/server encodings	1–2
Multiply decoded mojibake	café → cafÃ© → cafÃƒÂ©	1995–2010	CMS read–modify–save cycles	2
Mixed line endings	CR, CRLF and LF in one file	1985–1998	line-ending conventions of different operating systems	2
Markup residue	BBCode; deprecated HTML4 tags (`<font>`, `<marquee>`)	1998–2010	forum and CMS formats	4
Fixed-column hard wraps	RFCs and mailing lists wrapped at 72/80 characters; hyphenated wraps	1980s–2010	terminal width limits and conventions	5
Multiply escaped entities	`<` → `<` → `<`	1995–2010	repeated CMS escaping	2, 4

Flattened tables. In the text field of a Nemotron-CC document, a two-dimensional table of medical indicators (columns Indicator, Numerator, Denominator and Ratio) appears as a one-dimensional sequence of fields joined only by newlines: the column headers are stacked vertically and every data row is split over several lines, so the row and column relations are lost. The loss happens at extraction and cannot be undone by repairing the extracted text; such damage is an extraction-side limitation (App. L).

K In-Depth Analysis of the Repair Effect

This appendix expands §7. Every number is computed from the recorded results of App. H by `data/make_tables.py`. The task groups below are defined by a property of the benchmark items, not by the results, but the analysis is exploratory: it was carried out after the results were known, and its explanations are hypotheses that the forecasts at the end of this appendix are designed to test.

Table 30: Downstream changes of the full pipeline (V2) by task input format. Mean Δ in pp; better/worse over the tasks of the group; W/L: robust wins/losses. Groups are defined by a dataset property (whether an item contains a multi-sentence passage, a premise–hypothesis pair, a short context, or a closed-book question), not by the results; the MQ premise–hypothesis mean is carried by CB ($+23.8$ pp, not robust). Pooled: over the three tiers.

		HQ			MQ			LQ			Pooled
Group	n	mean	b/w	W/L	mean	b/w	W/L	mean	b/w	W/L	better (of $3n$)
Passage-conditioned	7	$+1.66$	6/0	3/0	$+0.48$	4/3	2/1	$+1.77$	6/1	3/0	16 of 21
Premise–hypothesis	5	-0.40	2/3	0/0	$+5.37$	5/0	1/0	$+1.86$	4/1	1/1	11 of 15
Short context	7	$+0.03$	4/2	0/1	-0.88	0/7	0/2	$+0.59$	3/2	1/0	7 of 21
Closed-book knowledge	7	$+0.77$	6/0	1/0	-0.21	2/5	0/1	$+0.23$	4/2	0/2	12 of 21

Task input format separates the effects. Table 30 groups the 26 scored tasks by what an item gives the model to read: a multi-sentence passage (LAMBADA, SciQ with its support paragraph, PubMedQA, BoolQ, RACE, CoQA, LogiQA), a premise–hypothesis pair (ANLI R1–R3, RTE, CB), a short context (BLiMP, HellaSwag, PIQA, CommonsenseQA, SocialIQA, COPA, WinoGrande) or a closed-book question (MMLU, ARC-Easy, ARC-Challenge, OpenBookQA, TruthfulQA-MC2, TriviaQA, NQ-Open). Passage-conditioned tasks improve in 16 of 21 task–tier pairs and on every

Table 31: Replication of the full pipeline's effect on each Paloma corpus across the three disjoint tiers (relative BPB change, %; negative is better). Range: largest minus smallest of the three tier values.

Corpus	group	HQ	MQ	LQ	mean	range
Falcon-RefinedWeb	web	−6.47	−6.35	−6.58	−6.47	0.23
C4-en	web	−5.16	−5.07	−5.24	−5.16	0.17
C4-100-domains	web	−4.51	−4.42	−4.68	−4.54	0.26
mC4-en	web	+0.10	+0.39	−0.00	+0.16	0.39
Dolma-100-subreddits	web	−2.92	−2.60	−2.58	−2.70	0.34
Dolma-v1.5	mixed	−2.16	−1.93	−2.27	−2.12	0.34
RedPajama	mixed	−1.98	−1.92	−2.00	−1.97	0.08
M2D2-Wikipedia	curated	+0.00	−0.03	−0.55	−0.19	0.55
M2D2-S2ORC	curated	+0.93	+1.17	+0.17	+0.76	1.00
WikiText-103	curated	+0.73	+2.54	+0.83	+1.37	1.81
Penn Treebank	curated	−0.13	−0.48	+0.64	+0.01	1.12

Table 32: Sensitivity of the full pipeline's downstream mean to the two most conspicuous gains (equal-weight mean change in pp over the tasks listed; SQuAD v2 excluded throughout), and a two-way decomposition of the 33 Paloma effects (relative BPB changes) into between-corpus, between-tier and residual variation. Both are descriptive summaries of the recorded results.

Tasks included	HQ	MQ	LQ	pooled
All 26 tasks	+0.58	+0.87	+1.05	+0.84
Without CB	+0.66	−0.05	+0.84	+0.48
Without CB and LAMBADA	+0.52	−0.22	+0.65	+0.32

Paloma effects: variation between evaluation corpora 98.2%, between input tiers 0.3%, residual 1.5%
between corpora, C4 evaluations merged: 97.9%
between corpora, without Falcon-RefinedWeb and both C4: 92.6%

tier on average ($+1.66/+0.48/+1.77$ pp); they are 27% of the tasks but carry 8 of the 12 robust downstream wins (67%) and 1 of the 8 losses. Premise–hypothesis tasks improve in 11 of 15 pairs (9 of 12 without CB). Short-context tasks improve in only 7 of 21 and closed-book tasks in 12 of 21, with means close to zero, and these two groups carry six of the eight robust losses. The finding does not rest on LAMBADA alone: on the other six passage-conditioned tasks the full pipeline improves 13 of 18 pairs, with 5 robust wins (SciQ on HQ and MQ, LogiQA on HQ and LQ, CoQA on LQ) against 1 loss (CoQA on MQ). Wins also dominate when small changes are ignored: counting only changes of at least 1 pp, the full pipeline has 26 improvements against 11 declines, and 21 against 10 without CB and LAMBADA (Table 33). Nor does the passage finding rest on SciQ's classification: SciQ is counted as passage-conditioned because its items carry a support paragraph, and moving it to the closed-book group still leaves the passage group at 13 better and 4 worse pairs with 6 robust wins against 1 loss. Under surface-only repair the passage-conditioned group already improves in 19 of 21 pairs, so both repair depths act in the same direction on it. The association is confounded by task domain, difficulty, prompt format and context length, which the grouping does not separate, and the classification should be read against the evaluation prompts actually used: SciQ's standard harness configuration includes its support paragraph, but the pinned harness revision decides whether it did here.

Signal at this scale. Only nine of the 26 tasks score at least 10 pp above random choice on every tier (Table 28). On these the full pipeline improves 6 of 7, 3 of 9 and 8 of 8 non-tied task–tier pairs on HQ, MQ and LQ, with five robust wins and no robust loss on HQ and LQ together; outside MQ, every robust downstream loss is on a near-chance or floor task. The pattern does not depend on the threshold: at 5, 10 or 15 pp above random choice, and with majority-class rates in place of $1/k$ for the unbalanced PubMedQA and BoolQ (about 55% and 62%), HQ and LQ keep robust wins and no robust loss on signal tasks, and MQ keeps two wins against three losses.

Why passage-conditioned tasks gain. In the 2013–2015 crawls, hard wraps at a fixed column (plain-text e-mail, mailing lists, forum exports), words split across lines and interrupting boilerplate

Table 33: Full-pipeline downstream changes against Raw whose magnitude reaches a threshold (improvements / declines over task–tier pairs; smaller changes omitted). Descriptive counts of the recorded changes, not significance tests.

Threshold	All 26 tasks (78 pairs)	Without CB and LAMBADA (72 pairs)
≥ 0.5 pp	34 / 18	29 / 17
≥ 1 pp	26 / 11	21 / 10
≥ 2 pp	12 / 4	7 / 4

break sentences and paragraphs of Raw documents into fragments (App. I). Stage 4 removes the interruptions, stage 5 rejoins wrapped lines and split words, and stage 8 repairs sentence boundaries left by truncation; artifact removal and line-level repair alone are 15.1%, 31.2% and 29.2% of the logged operations on HQ, MQ and LQ (Table 20). Every spurious newline, split-word fragment or boilerplate line also interposes tokens between words that belong together, lengthening the distance over which a model must relate them; repair shortens it. A model trained on the repaired text therefore sees more continuous multi-sentence context, which is what passage-conditioned tasks, and above all LAMBADA's final-word prediction from a broad discourse context, require. The hypothesis predicts that disabling stages 4–5 while keeping routing and restoration (a well-posed targeted ablation, App. L) would remove most of the passage-task gain, and that the gain would grow with the density of wrap and boilerplate damage in a source.

Candidate mechanisms and how to separate them. Four mechanisms fit parts of the record, and each predicts something the others do not (Table 34); they are hypotheses, not a completed causal analysis.

Table 34: Candidate mechanisms for the full pipeline's effects, the observations each fits, and a measurement that would distinguish it. None of the distinguishing measurements has been made.

Mechanism	Fits	Would be distinguished by
Restored contextual continuity	LAMBADA; passage-conditioned tasks, also without LAMBADA and SciQ	Gains that grow with the amount of usable context crossing repaired boundaries (full vs. truncated context on the saved checkpoints)
Fewer inconsistent surface forms	Near-identical BPB shifts on every tier	Effects on repaired patterns versus matched untouched text
Changed information density and exposure	Repair changes tokenization and pool composition	Tokens per character, document multiplicities and packing records
Alignment with the target distribution	Effects differ sharply between evaluation corpora; mC4-en and PTB break a web-versus-curated split	Independently measured similarity predicting the effect on new evaluation corpora

Alternative explanations we can bound. Two mechanisms unrelated to text quality could produce gains at a fixed token budget, and the record bounds both. *Exposure*: repaired documents are shorter, so a fixed budget could cover more documents; but the V2 pools hold only 0.2–2.4% more documents than Raw at nearly the same token count (Table 16), far too few to explain LAMBADA gains of 9–15% or BPB shifts of 2–6.6%. *Token savings*: tokens per document fall by only 0.1–2.2% from Raw to V2, so the model is not simply reading the same content in fewer tokens. What changes is how the text reads, not how much of it there is.

Why knowledge and short-context tasks do not move. Constraint C2 limits repair to bounded edits of diagnosed defects: it adds no facts and removes none. That does not make knowledge tasks immune in principle, because the information content of a corpus and the learnability of that content differ: a damaged entity name or an interrupted relation can leave a true statement present but hard to encode, and mending it can make the fact easier to learn without adding a proposition. The record shows no such gain at 1.7B (closed-book tasks: pooled $+0.26$ pp, 12 of 21 pairs better), so the appropriate expectation is conditional, gains where facts are present but structurally obscured and none where they are absent. Short single-sentence contexts, for their part, give line and paragraph

repair little to act on. Six of the eight robust downstream losses fall in these two groups: four on tasks near random choice or near zero at 1.7B (Table 28), where a change of a fraction of a point can cross $|z| = 2$ in either direction and so carries little evidence either way, and two (BLiMP, HellaSwag) on MQ, the tier whose downstream results are most mixed. WinoGrande is the only task that worsens on every tier (-0.7 to -1.6 pp, from a Raw score of 53–55% against 50% random choice). Candidate explanations for it and for the other declines, none tested here, are a shift of the training distribution away from conventions some targets reward, over-correction of legitimate rare forms (names, dialect, notation), altered local lexical cues, changed exposure through resampling and packing, and single-run training variability; an item-level audit of repaired sentences of the WinoGrande form would test the first two. We apply the same evidential standard to gains and losses: a near-chance baseline makes a small change uninformative, not harmless.

Why web-domain fit improves and two curated corpora worsen. (Counts and sums over Paloma corpora are not independent evidence: C4-en and C4-100-domains overlap, and Dolma-v1.5 and RedPajama both draw on web crawls.) Table 31 shows that the six web and mixed-source corpora that improve do so by almost the same amount on every tier (range at most 0.34 points), whereas the curated corpora move little and inconsistently, except WikiText-103 and M2D2-S2ORC, which worsen slightly on every tier. The interpretation we favour is that repair moves the training distribution toward normalized web text, which the web and mixed-source evaluation corpora share. Consistent with a formatting explanation, the two curated corpora that worsen are a benchmark distributed in tokenized form (WikiText-103) and scientific papers (S2ORC); Penn Treebank, also distributed in tokenized form, does not move on average, so formatting alone does not explain the pattern (see also "What web-domain BPB measures", App. L).

Replication across tiers, and what it says about single runs. Across tier pairs, the Spearman correlation of the full pipeline's per-corpus Paloma changes is 0.99 (HQ–MQ), 0.93 (HQ–LQ) and 0.95 (MQ–LQ). A two-way decomposition of the 33 effects places 98.2% of their variation between evaluation corpora, 0.3% between tiers and 1.5% in the residual (Table 32): the target distribution organizes the effect, the inherited tier barely does. The share stays at 97.9% when the two overlapping C4 evaluations are merged and at 92.6% when Falcon-RefinedWeb and both C4 evaluations, the largest web gains, are removed. Because the tiers are disjoint document sets, this is consistency of the domain-level effect across three different corpora under one protocol. The inherited labels also do not order downstream training value: the Raw MQ model exceeds both the Raw HQ and the Raw LQ model on 11 of the 26 tasks, and Raw HQ is best on 11, so a quality label does not settle whether repair is useful, and both HQ and LQ benefit. All treatment runs use seed 1234, so the tiers vary the data but not the initialization; initialization is bounded instead by the Raw seeds, which move the web-domain BPB values by 0.1–0.4% (relative σ_{pretrain}), 5 to 65 times smaller than the shifts. Task-level rankings do not replicate in the same way ($\rho = 0.14$, 0.04 and 0.19 over the 26 tasks; 0.76, -0.01 and -0.11 over the nine signal tasks): at this scale the reproducible downstream structure is the group-level pattern of Table 30 and the discourse-level LAMBADA gain, not the order of individual near-chance tasks.

Forecasts and how to test them. The account makes five predictions, each testable without new corpora: (F1) passage-conditioned tasks that are near chance at 1.7B (BoolQ, RACE, LogiQA) should show repair gains once models are large enough to use their passages; BoolQ already improves on every tier ($+0.3$ to $+2.6$ pp). (F2) Knowledge tasks should gain only where relevant facts are present but structurally obscured, and never where they are absent, so repair should be combined with content-oriented selection when knowledge is the target. (F3) Evaluation text preprocessed like normalized web text should gain and text with idiosyncratic conventions may lose slightly; comparing tokenized and detokenized versions of the same curated corpus would test the formatting component. (F4) Gains should track *recoverable* damage: sources with more wrap, split-word and boilerplate damage, such as early-web archives and forum or mailing-list dumps, should gain more than the 2013–2015 crawls, but only while the corruption leaves enough evidence for an admissible correction; beyond that point more damage need not mean more benefit. (F6) At larger scale, better use of context and greater robustness to corruption make competing predictions about effect size; the present evidence does not choose between them. Several of these can be tested without new pre-training: evaluating the saved checkpoints on LAMBADA with full and truncated context tests the context mechanism; scoring paired original and normalized versions of the same evaluation text

tests formatting compatibility; auditing edits on rare content tests over-correction; and measuring usable content per packed sequence tests the exposure account. The artifacts for these tests exist if retained: the paired Raw seed-1234 scores, the edit logs, the packing manifests and the checkpoints support continuous answer-likelihood evaluation, document-exposure counts and a blinded audit of rare forms, names and meaning-changing substitutions, none of which requires new pre-training; they do not replace additional treatment seeds or a trained mixed-corpus policy. (F5) Because domain-level effects replicate across corpora, the BPB profile of one small repaired run should forecast the profile of others, which makes it a cheap screening check before a full training run.

Relationship to synthetic data. Training recursively on model-generated data loses distributional tails (Shumailov et al., 2024), which motivates keeping the human-authored source accessible; it does not show that every source-conditioned rewrite is harmful, since accumulating synthetic data beside real data can avoid collapse (Gerstgrasser et al., 2024) and rephrasing beside the original can help training (Maini et al., 2024). Repair therefore sits alongside these strategies, with provenance requirements deciding which transformations are admissible; our experiments neither test recursive replacement nor show that repair prevents collapse, and a non-generative edit can also damage a rare expression, which is why the source is retained and edits are logged. Editorial scholarship has long kept original and corrected readings together (TEI Consortium, 2026), and generative restoration can itself assist historical research (Assael et al., 2025); the requirement is to distinguish source evidence from proposed reconstruction, not to forbid either. A sound scarce-data policy therefore makes three separate decisions: archive the original, decide whether to derive a corrected view, and decide which view enters a training set; rejecting a document from one training mixture need not delete it from the archive. The contribution here is the pre-training formulation, the algorithms and the evidence, not the idea of preserving the source.

Using these findings. Five practical rules follow. Deploy the full pipeline rather than the surface stages alone, since the linguistic stage improves Paloma fit over them on 31 of 33 tier–corpus pairs. Prioritize damaged and scarce sources, where the gains are largest and discarding is irreversible; for scarce, human-authored collections, repair-and-preserve is the only automated option studied here that both keeps the human-authored text and improves training on it (keeping the damaged original keeps the text but trains worse), whereas model-generated replacements are unacceptable to many of their users and lose the rare content that makes such collections valuable when models are trained on generated data (Shumailov et al., 2024). Apply repair by tier: on HQ and LQ it improves downstream tasks and domain fit, on MQ its downstream effects are mixed (3 robust wins, 4 losses) while its domain-level shifts match the other tiers, so a practitioner whose target is downstream accuracy can restrict repair to the high and low tiers, or measure on a small run first (F5); the choice is target-dependent (web-domain fit and LAMBADA-like prediction favour repairing MQ, broad task accuracy with a strict regression tolerance favours keeping it Raw, and short-context tasks call for targeted validation), and a model trained on repaired HQ and LQ with Raw MQ has not been evaluated, so its benefit cannot be read off the separately trained tier models. A more ambitious extension conditions repair on recoverability, edit confidence and the target, rather than on the tier label. Evaluate repair with measurements that have signal at the scale of the experiment, discourse-level LAMBADA and web-domain BPB at 1.7B, rather than with near-chance benchmarks. And pair repair with content-oriented selection rather than substituting one for the other: repair acts on form, which the selection classifiers barely see (§4), and selection acts on content, which repair leaves unchanged.

L Limitations, Threats to Validity and Future Work

This appendix expands the scope statement of §7. For each threat we state what the evidence establishes, what it does not, and why the design was chosen; none of these paragraphs adds an unmeasured number.

Scale, benchmarks and data source. All end-to-end evidence comes from a 1.72B-parameter model trained for one epoch on a 34.4B-token (Chinchilla-optimal) budget with 2013–2015 Common Crawl as the only input. This is the scale at which the field ranks curation decisions: FineWeb's filtering ablations use 1.71B-parameter models on ≈28B tokens, with selected results confirmed on 350B tokens (Penedo et al., 2024), and DCLM's 1B-1x track uses 1.4B parameters on 28.8B

tokens (Li et al., 2024); ReWire adopts the same DCLM scales (Nguyen et al., 2025). Our claim is therefore an existence claim at the standard proxy scale, with effect sizes of 9–21 noise bands on LAMBADA and 30–67 on web BPB, not a scaling law. Two considerations bear on larger scales without settling them: the defects repaired are properties of the corpus rather than of the model, so a larger model trained on the same crawl sees the same corruption; and the projected exhaustion of public human text between 2026 and 2032 (Villalobos et al., 2024) means that the marginal value of recovering discarded documents rises, not falls, as training sets approach the stock of available text. Whether the *gains* grow with model size and training length (7B–13B parameters, multi-epoch or beyond-Chinchilla ratios) remains open. Repeating the ablation at six sizes from 150M to 7B in the DataDecide framework (Magnusson et al., 2025), whose own suite spans 14 sizes from 4M to 1B parameters, would answer it, and code, mathematics, long-context and cross-lingual benchmarks, the suites expected to benefit most directly from structural repair, remain to be added.

Single treatment runs and the noise calibration. The Raw baseline of each tier has three seeds; each repaired configuration is a single run, so treatment-side variance and the covariance of paired runs are unmeasured, and $z = \Delta/\sigma_{\text{tot}}$ is an effect size in units of the Raw noise band rather than a calibrated test statistic. Three answers are offered instead of a claim of formal significance. First, the threshold ladder of Table 21: the single-run correction $2\sqrt{1 + 1/3} \approx 2.31$ (Table 21; it assumes equal variance and independence), a Bonferroni correction over the 38 evaluations scaled by the same factor (3.71) and a cutoff of 5, which exceeds the two-degree-of-freedom t critical value 4.30 scaled by the same $\sqrt{4/3}$ (4.97), all leave V2 net-positive and V1 never net-positive on every tier; the headline effects (LAMBADA $z = 12.6/8.8/20.6$, web BPB $|z| \geq 30$) are far beyond any of them. Second, replication across data regimes: the three tiers are built from disjoint documents, and the same pattern recurs on all three: both recipes above Raw on LAMBADA and below Raw on web BPB, with the Penn Treebank cost of V1 (7.8–11.7%) inside the noise band under V2 (-0.5 to $+0.6\%$). Third, transparency: deltas with $1 \leq |z| < 2$ are only suggestive and are never counted, and a near-degenerate band can make a negligible effect look detectable (SocialIQA on MQ-V1: $\Delta = +0.17$ pp, $z = +8.16$, because the seed term is zero and the tiny evaluation term, 0.021, prevents the binomial fallback). Each treatment run uses seed 1234, and hence the same initialization as one of the three Raw runs; the differences reported here are against the three-seed Raw mean. Given a fixed compute budget we chose three data regimes over three seeds of one regime; the seed axis is the natural next experiment.

Attribution to layers, not stages. The V1/V2 contrast attributes the difference to stage 8 as a layer (its grammar and syntax sub-steps are not ablated separately); the routing, validation and restoration modules are supported by their component evidence (Table 1), not by separate pre-training ablations. We regard the layer as the right unit for a training ablation for three reasons. The two layers are the deployment choices a practitioner actually has (surface repair is cheap and widely approximated by existing pipelines; the linguistic layer is the novel, heavier one), and finding that the cheap layer alone is net-negative is itself the informative ablation result. Stage-level leave-one-out ablations are uninformative for stages with prerequisites under the ordering constraints of §3.1: removing stage 3 makes stage 7 destructive for code, removing stage 1 makes stage 2 unrepairable, so such a run measures the damage of a broken topology rather than the value of a stage. Targeted swaps that respect the topology (disabling line repair while keeping routing and restoration, exchanging boundary policies, replacing the validator behind its interface) are well posed and are the right follow-up; they are unmeasured here for lack of compute, since each is another 1.7B run. During development, each stage was instead checked at the text level: by its own validation gate and rollback, by the sensitivity benchmark that shows which defects each tool detects (App. D–E), by the per-stage operation shares (Fig. 6) and by human inspection of random before/after samples. Those checks were used during development to confirm intended edits and catch visible damage; they do not quantify per-stage improvement, and the pre-training study establishes only that the resulting recipes change training utility. Neither establishes the training contribution of each stage, which we do not claim.

Tier re-assignment and the deployment modes. A reader may ask the natural question the title raises: how many documents that a filter discards would it accept after repair? We do not answer it for Nemotron-CC: its three classifiers are public, but a document's tier is the maximum of their percentile buckets over the full 99-snapshot corpus, and we did not reconstruct those percentile thresholds; thresholds recomputed on our 2013–2015 subset would define different tiers. Scoring

repaired documents with the three released classifiers against thresholds estimated from the released tiers is a feasible follow-up. We therefore measure what the filter is a proxy for. Training utility at a fixed tier is the deployment quantity in the scarce mode, where no filter follows repair, and a stricter test than tier promotion in the rich mode: a document that trains a better model is worth keeping whether or not a particular classifier would promote it, whereas a document that a scorer promotes is not thereby shown to be useful—RQ1 shows that the scorers in use are insensitive to, or inverted on, the layout defects that repair fixes, so a count of promotions would under-count genuine repairs and reward cosmetic ones. Holding the inherited tier fixed also separates repair from selection: if repair promoted documents across tiers, a comparison of repaired LQ against Raw HQ would confound the two. What the record does show about direction is the quality shift of Table 20: under DCLM fastText, the classifier family in Nemotron-CC's ensemble, 95.0–97.4% of sampled documents score higher after repair, so repair moves documents toward acceptance; how many cross a given threshold depends on that threshold. A filter-upgrade experiment with a fully specified open filter is a worthwhile separate study.

Processing cost, and why not generative rewriting. We make no cost comparison with generative rewriting or with heuristic filtering, and we do not claim a measured throughput: the design constraints of the framework rank quality above cost and speed and require a streaming, single-pass architecture with a target of hundreds to thousands of documents per CPU core per second on the main path (design constraints of the framework), but the realized throughput was not instrumented per stage. The operational record is that a single CPU server processed all three tiers, ≈177M documents under each of V1 and V2 (Table 16), in a few days, with the validator chosen under an explicit CPU-only constraint (§4) and no GPU inference over the corpus. We avoid generative rewriting for two reasons that are independent of cost. First, accessibility: a CPU-only pipeline is usable by groups that have no GPU cluster for corpus-scale inference. Second, temporal consistency: the archives that motivate repair-and-preserve are historical, and a rewriter trained on recent text injects modern phrasing and priors into old documents (a 1998 post rendered in 2024 idiom), whereas bounded, deterministic byte transformations cannot paraphrase. Stage 8 satisfies this only partially: its dictionary, grammar and frequency resources encode present-day conventions and may leak them into historical text; temporal consistency is therefore independent of semantic preservation (an edit can preserve meaning and still modernize form), and we list it as a known limitation. Once the experiment code is pinned (Reproducibility statement), the cost measurement can be made by anyone who wishes to compare.

Closest prior work. The nearest methods are not unconstrained rewriters. ProX (Zhou et al., 2025) has a model write, per document, a program of calls to a predefined executor (keep or drop the document, remove lines, normalize strings) that is then run deterministically; RePro (Yu & Xiong, 2026) trains a rephraser with quality and faithfulness rewards and trains on organic and recycled text together. REPAIR-FIRST differs in what decides an edit (deterministic diagnostics and bounded candidate sets, with a learned router and scorers but no per-document model-written program), in abstention by rollback, in the reversibility log, and in evaluating fixed-budget utility within inherited quality tiers; several of its operations (line removal, string normalization) overlap with ProX's executor. No head-to-head comparison was run, and we claim no superiority over either method.

Protection accuracy is not measured end to end. The 98.3% figure is language-identification accuracy on 1,000 code blocks that had already been detected; the recall of the detectors on code, mathematics and structured data in the wild, and hence the end-to-end rate at which typed content is protected rather than edited as prose, were not measured, and Proposition 1 guarantees only what its assumptions cover (a non-crossing hierarchy, unique sentinels, unmodified payloads). A detection-recall study on an annotated page sample would close this gap.

Semantic preservation of edited prose. Exact restoration (Proposition 1) covers protected non-prose payloads. For prose, the admissible-edit contract of §2—scope, evidence, locality and abstention—bounds every change to a diagnosed span and a candidate set close to the original, logs it and makes it reversible; this is the difference from generative rewriting, whose output space is unbounded. Bounding is not preservation: dictionaries, lexicons and grammaticality models can normalize legitimate rare forms, names, dialect and notation, so "no generative model" does not by itself establish that meaning or historical wording is preserved, and scarcity raises the cost of

exactly such errors. The blinded audit below is the test. It is a constraint enforced by construction and auditable, not a proof of semantic equivalence: a dictionary-valid correction can still change a name, a quantity or a dialect form, which is why stage 8 confines candidate substitution to fragments already flagged as suspect (a real word is replaced inside such a fragment only when context, frequency and the threshold agree), handles real-word spelling errors through local n-gram and grammatical evidence rather than dictionary membership, and gates every candidate (App. B.4). Table 35 lists the failure mechanisms the specification anticipates and the conservative response each stage takes. A blinded semantic audit of accepted edits, stratified by operator, tier and validation outcome, is the direct test and is future work.

Table 35: Failure mechanisms anticipated by the specification and the conservative response of the corresponding stage. These are design constraints, not measured frequencies.

Mechanism	Potential harm	Conservative response
Compound dehyphenation	Removes a lexical hyphen or damages a URL	Join only if the joined form is in the lexicon; otherwise keep the hyphen (Eq. 2)
Boilerplate false positive	Deletes a quoted policy or an explanatory notice	Prefer DOM/container evidence; require both repetition and vocabulary conditions; log for rollback
Missed code span	Prose whitespace repair alters indentation or literals	Strong containers first; uncertain code stays generic code and is protected
Over-eager math detection	Treats prices or shell variables as formulas	Three independent conditions; never scan inside code-like tags
Lexical over-correction	Replaces a proper name, quantity or valid rare word	Correct only dictionary-unparsable fragments; rank by confusion-weighted distance and context; threshold τ_{apply}
Validator self-confirmation	Rewards removal of its own trigger while meaning changes	Defect-specific ownership; document-level pass/reject; task-level evaluation
Sentinel collision	Duplicates, deletes or misplaces a protected payload	Native PUA characters (early web fonts, CMS markers) are removed in stage 6, sparing the tracked sentinels; this happens after allocation, so a pre-existing character equal to a sentinel is not excluded, and sentinel multiplicity is assumed, not checked; SHA-256 per payload

The quality-shift judges. The quality shifts of Table 20 are descriptive and rely on scorers: DCLM fastText on HQ and MQ, an LLM judge on LQ. DCLM fastText is also one of the stage-9 validator's tools, so its shift partly reflects the acceptance rule itself. Both scorers reward the removal of layout defects, which is the property the pipeline targets, but an LLM judge in particular is sensitive to formatting and cannot separate cosmetic from substantive improvement; without human-annotated ground truth the LQ shift (8.92→15.22) should be read as agreement with the pipeline's objective, not as evidence of utility. Utility rests on the training runs, which involve no judge.

What web-domain BPB measures. Part of the web-domain BPB gain may reflect closer formatting to C4 and Falcon-RefinedWeb, whose own pipelines normalize line breaks and whitespace, rather than better modelling of content; Penn Treebank and WikiText-103, with idiosyncratic formatting, move the other way under V1. We therefore present web BPB as web-domain fit, report the curated and mixed-source slices beside it, and rest the downstream claims on the task scores.

The MQ band. MQ is the least stable tier, with volatile task-level results. Modelling quality as a continuous distribution rather than a single tier, with conditional repair strategies or dynamic routing, may reduce this volatility.

No visual layout. All diagnostics operate on extracted text. Column reading-order errors, misaligned table cells and sliced formulas that arise before extraction are outside the pipeline's reach, because the layout signal is lost at its input; they are an extraction-side problem (cf. Ma et al., 2025; Poznanski et al., 2025).

Composition with generative rewriting. The pipeline is designed as a cleaning pre-pass for rewriters such as ReWire (Nguyen et al., 2025) and RePro (Yu & Xiong, 2026), but we have not run the combination end-to-end; whether the joint gain exceeds the sum of the individual gains remains to be tested.

Scope. No benchmark decontamination audit was performed on the training pools; early crawl dates reduce but do not rule out overlap with benchmark source documents. The pipeline targets English text. Among the non-prose content types only code has a fully specified repair-and-gate path, and in this study no typed repair was applied: code, mathematics and structured data were protected and restored verbatim (§3.2). The anomaly classes of pre-2013 web text (App. I) are covered by the pipeline's design; where data are that scarce, repairing rather than discarding is the natural choice, and we have not run end-to-end training on such archives.