Lexical Perturbations Disrupt LLM Reasoning: An Empirical Study of Attention Diversion

Jiaqian Zhu[1] **Yang Zhang**[2,*] **Junhua Ding**[2] **Xiaowei Yu**[1,*]
[1]Missouri University of Science and Technology, Rolla, MO, USA
[2]University of North Texas, Denton, TX, USA
{jzbmn, xych8}@mst.edu
{Yang.Zhang, Junhua.Ding}@unt.edu
*Corresponding authors

Abstract

Large Language Models (LLMs) achieve strong reasoning performance, but their robustness to realistic lexical corruption remains poorly understood. We evaluate four open-weight instruction-tuned models and frontier models across four reasoning benchmarks under keyboard noise, character swaps, and filler insertion. Character-level perturbations substantially degrade accuracy, especially on multi-step reasoning tasks, while filler insertion has little effect. We trace this asymmetry to *Attention Diversion*: lexical corruption fragments subword tokenization, and the resulting fragments attract disproportionate attention mass, concentrated in middle and final transformer layers. Length-matched controls confirm that fragmentation, not prompt length, drives the loss. A factorial intervention then shows why the damage is hard to undo: fragmentation corrupts token content and attention allocation together, and the two are coupled. Restoring clean attention while the content remains corrupted is actively harmful, restoring content alone is insufficient, and only restoring both recovers a substantial share of the gap. This coupling explains why inference-time strategies, including chain-of-thought prompting, spell-checking, self-repair, and stronger repair models, fail to consistently recover performance: each addresses one channel at a time. Code and data are available at https://github.com/Jiaqian-Janelle/Attention-Diversion.

1 Introduction

Large Language Models (LLMs) have achieved substantial gains on reasoning benchmarks (Brown et al., 2020; OpenAI, 2023; Wei et al., 2022; Llama Team, AI @ Meta, 2024; Qwen Team, 2026; Jiang et al., 2023), yet remain brittle under small surface-form changes. Minor lexical corruptions that preserve meaning, such as mistyped keys or adjacent-character swaps, can markedly degrade accuracy despite being transparent to human readers. In contrast, filler insertions such as *"um"* or *"you know"* leave performance nearly unchanged, even though they increase prompt length. This asymmetry suggests the failure is not a consequence of longer prompts or semantic distortion, but of how surface-form corruption is represented.

Prior work has studied character-level noise in neural NLP systems (Belinkov and Bisk, 2018; Pruthi et al., 2019; Li et al., 2019; Ebrahimi et al., 2018), but three gaps remain. First, it concentrates on classification and translation, leaving reasoning-intensive tasks underexplored. Second, it alters surface form and prompt length together, leaving the two confounded even though only the former changes subword token identity. Third, it documents accuracy losses without connecting them to changes in internal processing.

We address these gaps through a controlled empirical study of lexical perturbations in LLM reasoning. We evaluate four open-weight instruction-tuned models across BoolQ, PIQA, HellaSwag, and GSM8K, and further test larger-scale, harder, non-English, and frontier-model settings. Across these settings, character-level perturbations consistently degrade performance, with the largest losses on multi-step mathematical reasoning, whereas filler insertion has little effect.

To explain this asymmetry, we identify **Attention Diversion**. Character-level corruption fragments subword tokenization, producing rare or unusual token pieces that attract disproportionate attention mass during inference. As illustrated in Figure 1, corrupted fragments redirect attention away from task-relevant evidence and answer cues. Layer-wise analysis further shows that this diversion concentrates in middle and final transformer layers, where contextual representations are composed and mapped toward output predictions.

Our central claim is that lexical fragility in reasoning is driven primarily by tokenizer fragmentation rather than prompt length alone. Length-

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 16706–16727
October 24-29, 2026 ©2026 Association for Computational Linguistics

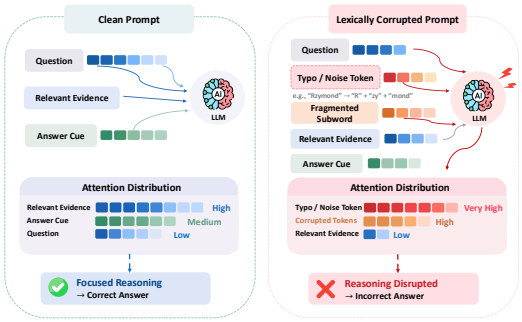

Figure 1: Illustration of Attention Diversion under lexical corruption. In clean prompts, attention concentrates on relevant evidence and answer cues. After perturbation, corrupted token fragments absorb a disproportionate share of attention mass.

matched controls, per-example regression, and representation-level interventions support this account. A factorial intervention isolates what fragmentation damages: it disrupts the identity of the tokens the model reads and the allocation of attention over them at the same time, and these two effects cannot be separated. This interdependence also accounts for our mitigation results: common inference-time strategies, including chain-of-thought prompting, spell-checking, and LLM self-repair, do not consistently recover performance, because each acts on one side of the problem alone.

Our contributions are:

- We systematically evaluate lexical robustness across four reasoning benchmarks, four open-weight model families, frontier models, and realistic perturbation types.
- We identify **Attention Diversion** and, through a factorial intervention, show that the diverted attention is not independently manipulable: it is functionally bound to the corrupted content it addresses, so neither channel can be repaired on its own.
- We trace harm to the irreplaceability of displaced evidence rather than to diversion magnitude, with numeric tokens dominating GSM8K failures. We then bound repair, finding that a tokenizer-level defense detects fragmentation reliably yet cannot invert it.

2 Lexical Perturbations in LLM Reasoning

We use *lexical perturbation* to denote surface-form changes that preserve the intended meaning of a prompt while altering how the input is written. Such perturbations arise naturally from typing errors, speech-to-text transcription, and noisy prompt construction. We focus on three realistic perturbation types, illustrated in Table 1.

Table 1: Illustrative examples of the three perturbation types applied to a single base prompt, with the perturbed span in red.

Perturbation	Prompt
Clean	*Which option best explains why the ice melted faster on the metal surface?*
Keyboard noise	*Which option best expkains why the ice melted faster on the metal surface?*
Typoswap	*Which option best explains why the ice meltde faster on the metal surface?*
Filler	*Um, which option best explains why the ice melted faster on the metal surface?*

Perturbation types. **Keyboard noise** replaces a character with an adjacent QWERTY key (e.g., *reason* → *reaspn*), simulating fast or imprecise typing. **Character swap** (typoswap) transposes adjacent characters within a word (e.g., *problem* → *porblem*), modeling common spelling errors. **Filler insertion** adds conversational disfluencies such as *"um"* or *"you know"* at natural pause points, reflecting spoken or voice-transcribed input.

Perturbation severity. We apply perturbations at four rates $r \in \{0.05, 0.10, 0.20, 0.30\}$, where r denotes the fraction of words perturbed. For keyboard noise and character swap, one randomly chosen character within each selected word is modified. Filler insertion adds disfluency phrases without corrupting existing lexical items. This design lets us compare character-level lexical corruption against prompt length inflation.

Expected representation effect. Character-level perturbations can break learned subword merges, causing a word to be split into rare or unusual token fragments. In contrast, filler insertion mainly adds familiar tokens while preserving the tokenization of the original prompt. This contrast motivates our mechanistic analysis in Section 5, where we test whether tokenizer fragmentation induces Attention Diversion and reasoning failure.

3 Experimental Setup

3.1 Datasets

We evaluate lexical robustness on four reasoning benchmarks: BoolQ (Clark et al., 2019a) for passage-based yes/no question answering, PIQA

(Bisk et al., 2020) for physical commonsense reasoning, HellaSwag (Zellers et al., 2019) for context continuation, and GSM8K (Cobbe et al., 2021) for multi-step mathematical reasoning. Together, they span reading comprehension, commonsense reasoning, contextual inference, and math reasoning.

3.2 Models

Our main experiments cover four open-weight instruction-tuned models: Llama-3.1-8B-Instruct (Llama Team, AI @ Meta, 2024), Mistral-7B-Instruct-v0.3 (Mistral AI, 2024), Qwen3.5-9B (Qwen Team, 2026), and Gemma-2-9B-IT (Gemma Team, 2024). These models have comparable scale (7B–9B) but differ in model family, tokenizer design, and training pipeline, allowing us to test whether lexical fragility generalizes across architectures. We additionally evaluate GPT-4o (OpenAI, 2024) and GPT-5.4 (OpenAI, 2026), the latter under both `none` and `high` `reasoning_effort` settings, for prediction-level robustness. Frontier-model results are reported in Appendix A.

3.3 Evaluation Protocol

For each dataset, we sample 1,000 examples from the evaluation split and repeat experiments with three random seeds (42, 43, 44) to account for subset and perturbation variance. Models are evaluated with greedy decoding, and accuracy is the primary metric. We report mean accuracy across seeds. Standard deviations are small, typically below one percentage point. Main-text results focus on severity $r = 0.1$, with all severity levels reported in Appendix B. For GSM8K, we extract the final numeric answer following the standard #### format. In addition to accuracy, we record tokenization statistics and attention diagnostics to connect prediction-level degradation with internal processing changes.

4 How Robust Are LLMs to Lexical Perturbations?

We first examine whether lexical perturbations produce systematic reasoning failures, and whether these failures are specific to character-level corruption rather than prompt length. Table 2 reports accuracy at severity $r = 0.1$, while Figure 2 summarizes trends across all severity levels. Full results are provided in Appendix B.

Character-level corruption degrades reasoning across models. As shown in Table 2, keyboard noise and typoswap consistently reduce accuracy across models and tasks, while filler insertion has minimal effect. For example, Qwen3.5-9B drops from 0.845 to 0.706 on GSM8K under keyboard noise and to 0.569 under typoswap, whereas filler insertion leaves accuracy nearly unchanged (0.840). This contrast provides the first evidence that the failures are tied to lexical-form corruption rather than prompt length alone.

Degradation scales with perturbation severity. Figure 2 (left) shows that average accuracy decreases monotonically as corruption severity increases from $r = 0.05$ to $r = 0.30$ under both keyboard and typoswap perturbations. At $r = 0.30$, average accuracy drops below 0.40 for both character-level perturbations, compared to 0.74 on clean prompts. Filler insertion remains near the clean baseline across severities.

Multi-step reasoning is most vulnerable. Figure 2 (middle, right) shows substantial task variation. GSM8K exhibits the steepest degradation, falling from 0.619 to 0.191 under keyboard noise and from 0.549 to 0.112 under typoswap as severity increases from $r = 0.05$ to $r = 0.30$. BoolQ is comparatively robust, suggesting that tasks requiring precise numerical interpretation and multi-step reasoning are more sensitive to lexical corruption. This ordering is not explained by the amount of attention diversion: on Llama-3.1-8B, GSM8K sustains a lower diversion ratio than BoolQ yet loses several times as much accuracy (Appendix C). What differs is whether the displaced evidence is recoverable from elsewhere in the prompt, which Section 5.4 quantifies.

The pattern extends beyond the main setting. The same pattern holds beyond 7B–9B English models (Appendix D). At 70B scale, fragility persists but is attenuated: GSM8K drops 13 points under typoswap, while BoolQ is stable. On MATH algebra, typoswap again produces the largest drop. On Chinese CMath (Wei et al., 2023), keyboard perturbation causes a 30-point drop while filler remains harmless. Frontier models show the same pattern, and reasoning does not absorb it (Table 3). GPT-5.4's thinking mode sustains the largest keyboard degradation of any configuration we evaluate (-0.147), $1.6\times$ that of the same model with reasoning disabled, despite near-identical clean accuracy. Per-task results are in Appendix A.

These results establish a robust empirical pat-

Table 2: **Accuracy under lexical perturbations at severity** $r = 0.1$ ($n = 1{,}000$). Character-level perturbations (KB, TS) consistently reduce accuracy, whereas filler insertion (Fil) has minimal effect. Cells are shaded by degradation relative to clean: ≥0.15, 0.05–0.15.

	BoolQ				PIQA				HellaSwag				GSM8K			
Model	Cln	KB	TS	Fil	Cln	KB	TS	Fil	Cln	KB	TS	Fil	Cln	KB	TS	Fil
Gemma-2-9B-IT	.875	.809	.826	.878	.843	.673	.645	.827	.693	.507	.469	.697	.804	.729	.586	.802
Llama-3.1-8B-Instr.	.834	.723	.752	.835	.758	.586	.630	.753	.493	.373	.354	.494	.698	.518	.420	.709
Mistral-7B-Instr.-v0.3	.847	.731	.758	.848	.693	.549	.541	.695	.413	.347	.336	.408	.397	.276	.201	.378
Qwen3.5-9B	.878	.803	.812	.870	.892	.725	.721	.884	.863	.693	.708	.873	.845	.706	.569	.840

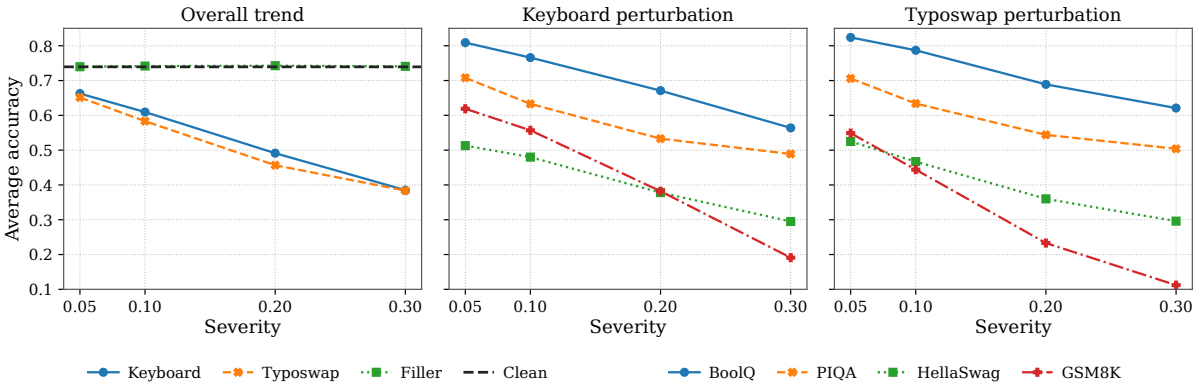

Figure 2: **Impact of lexical perturbations on reasoning accuracy.** Left: average accuracy across severity. Middle and right: task-wise performance under keyboard noise and typoswap. Character-level perturbations degrade monotonically with severity, while filler stays close to clean.

Table 3: **Cross-model comparison at** $r = 0.1$, averaged over the four benchmarks. GPT-5.4 Thinking sustains the largest keyboard degradation of any configuration despite near-identical clean accuracy to Standard. Per-task results are in Appendix A.

Model	**Clean**	**Keyboard**	**Typoswap**
GPT-5.4 (Standard)	.908	.815 (-.093)	.858 (-.050)
GPT-5.4 (Thinking)	.905	.758 (-.147)	.818 (-.087)
GPT-4o	.878	.798 (-.080)	.818 (-.060)
Qwen3.5-9B	.869	.732 (-.137)	.702 (-.167)
Gemma-2-9B	.804	.680 (-.124)	.632 (-.172)
Llama-3.1-8B	.696	.550 (-.146)	.539 (-.157)
Mistral-7B	.588	.476 (-.112)	.459 (-.129)

tern: character-level perturbations degrade reasoning, while filler insertion largely does not. This raises the central mechanistic question: why does corruption of lexical form matter more than added prompt length? We answer this question next by analyzing tokenization and attention.

5 What Causes Lexical Fragility in LLM Reasoning?

Our analysis tests a three-step mechanism. Lexical corruption fragments subword tokenization, changing the token identities the model consumes. The fragments then attract disproportionate attention mass, producing Attention Diversion. Fragmentation in turn drives reasoning failure beyond any effect of prompt length. A factorial intervention (Section 5.3) then shows the second and third steps to be a coupling between attention and token content rather than a one-directional chain. Appendix E illustrates the full pathway.

5.1 From Fragmentation to Attention Diversion

Lexical perturbations change the token sequence presented to the model in two ways. Character-level perturbations substantially increase the number of subword tokens, and this increase grows with corruption severity, while filler insertion remains close to the clean baseline. Token overlap shows the same pattern. Keyboard noise and typoswap sharply reduce overlap with the clean prompt, while filler leaves it largely intact (Appendix F).

This disruption alters attention allocation during inference. We measure the *diversion ratio*, defined as the fraction of attention mass assigned to cor-

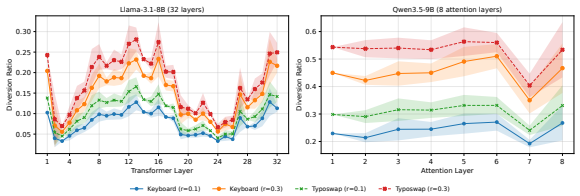

Figure 3: **Layer-wise diversion ratio under lexical corruption.** Llama-3.1-8B (left) peaks in middle and final layers, while Qwen3.5-9B (right) shows stronger late-layer diversion. Averaged across four benchmarks. Shading indicates standard error.

rupted tokens. Keyboard noise and typoswap consistently increase corrupted-token attention across models and severities. Filler insertion does not, staying below 0.025 throughout. The gap widens with severity. At r=0.30, typoswap reaches 0.527 on Qwen3.5-9B, 0.299 on Gemma-2-9B, 0.211 on Mistral-7B and 0.172 on Llama-3.1-8B, against 0.448, 0.256, 0.176 and 0.144 under keyboard noise (Table 14 in Appendix G). The diversion ratio often exceeds the fraction of corrupted tokens in the prompt, indicating that corrupted fragments act as attention attractors.

Diversion is layer-structured rather than uniform. To understand *where* in the transformer stack diversion occurs, we compute the diversion ratio at each layer. Figure 3 shows that Attention Diversion is structured rather than uniform. In Llama-3.1-8B, diversion forms a bimodal pattern, peaking in middle layers and again near the final layers. Qwen3.5-9B shows higher overall diversion, with elevated ratios toward later attention layers. These patterns suggest that lexical corruption affects both contextual composition and prediction-facing representations. Per-task breakdowns in Appendix C show the same qualitative pattern.

5.2 Fragmentation, Not Length, Drives Failure

The preceding analyses establish that character-level perturbations simultaneously increase fragmentation and degrade accuracy, but leave the causal role of fragmentation ambiguous. The loss could stem from fragmentation itself or merely from increased prompt length. We separate the two with a controlled experiment on the same GSM8K samples (Llama-3.1-8B, n=500, r=0.1) comparing **Group A** (filler insertion, which adds tokens but preserves subword structure), **Group B** (keyboard noise and typoswap, which disrupt token identity while preserving semantics), and **Group C** (neutral word padding calibrated to match Group B's token count). Table 4 summarizes the results. Group A adds more tokens than Group B (63 vs. 62) yet maintains near-clean accuracy (0.686 vs. 0.694), so prompt length alone does not impair reasoning. At matched token counts, Group B has 8× the fragmentation of Group C (0.080 vs. 0.010) and lower accuracy (0.642 vs. 0.662), isolating fragmentation beyond length as the driver. The typoswap row does not follow, carrying the same fragmentation as keyboard noise (0.079 vs. 0.080) yet a higher accuracy (0.682 vs. 0.642), so fragmentation magnitude alone does not fix the size of the drop within a single cell. We therefore estimate its effect across perturbation types with the regression on the n=4,160 set (Appendix H) rather than from this comparison.

Across the full set, fragmentation is the strongest univariate predictor of a correct-to-wrong flip ($\beta = +0.66$, OR=1.93, $p<10^{-81}$). Controlling for token-count change strengthens it ($\beta = +1.04$, OR=2.84) while token-count change loses independent positive predictive power (Appendix H). Conditioning on the outcome shows the same signature. Examples that flip to incorrect exhibit higher diversion ratios, lower token overlap, and larger token-count inflation than those that remain correct, especially at low-to-moderate corruption (Appendix F).

Both channels are load-bearing. Two interventions show that neither the corrupted content nor the attention it attracts is inert. Replacing corrupted token embeddings with nearby clean-token embeddings on GSM8K (Llama-3.1-8B, r=0.1 keyboard, n=100) improves accuracy from 0.42 to 0.52 against a clean baseline of 0.80, recovering 26% of the gap with only 2.95 replacements per example (Appendix I). Conversely, suppressing attention to outlier tokens by computing $s_j = \|h_j - \bar{h}\|_2$ and applying $\tilde{A}_{ij} = A_{ij} \cdot \sigma(-\lambda s_j)$ with $\lambda \in \{0.5, 0.9\}$ consistently *worsens* accuracy, with GSM8K dropping up to 0.31 (Appendix J). Corrupted-token attention is therefore not removable noise: the model relies on the fragments to partially reconstruct the intended input. Recovery on one side and harm on the other suggest the channels are not independent.

5.3 Attention and Token Content Are Coupled

Suppression shows that corrupted-token attention cannot be removed without cost, but leaves open

Table 4: **Controlled fragmentation experiment on GSM8K** (Llama-3.1-8B, r=0.1, n=500). At matched token count, keyboard noise fragments more than length-matched padding and scores lower. Typoswap does not follow, and Appendix H estimates the effect across perturbation types.

Group	Tokens	Frag.	Overlap	Acc.
Clean	59	0.000	1.000	0.694
A (Filler)	63	0.010	0.990	0.686
C (Length-matched)	62	0.010	0.990	0.662
B (Keyboard)	62	0.080	0.920	0.642
B (Typoswap)	62	0.079	0.921	0.682

Table 5: **Factorial restoration of attention and token content on GSM8K** (r=0.1, n=100 per cell, paired cluster bootstrap B=2000). Percentages of the clean–corrupted gap recovered, 95% CIs below. Grey marks recovery and purple harm. Unshaded cells have 95% confidence intervals that include zero. Neither channel alone recovers accuracy, and clean attention over corrupted content is harmful, yet restoring both is strongly super-additive.

Restored *gap (pt)*	**Llama** 34	**Mistral** 24	**Gemma** 22	**Qwen*** 14
Attention	−41 [-88, -8]	−25 [-87, +10]	−59 [-164, -3]	−164 [-625, -64]
Content	+56 [+24, +88]	+25 [-8, +56]	−27 [-100, +15]	+21 [-100, +83]
Both	+71 [+41, +103]	+54 [+17, +94]	+45 [+9, +79]	−50 [-300, +18]
Interaction	+56 [+11, +117]	+54 [+5, +131]	+132 [+56, +280]	+93 [+8, +384]

*Recovery's denominator is the clean–corrupted gap, so variance grows as the gap narrows. Qwen's 14-point gap inflates all four intervals, and the same holds for BoolQ, which we do not decompose.

what the model would do if that attention were correct while the content remained corrupted, or the reverse. We therefore cross the source of the attention pattern with the source of the token embeddings. Clean pre-softmax attention logits are injected into the corrupted forward pass through a difflib position map on both axes, with the corrupted causal mask applied afterwards and the softmax recomputed. Embeddings are restored through the same map. We report recovery, $(A_{\text{interv}} - A_{\text{corrupt}})/(A_{\text{clean}} - A_{\text{corrupt}})$, on GSM8K at r=0.1 (n=100 per cell, bootstrap B=2000). Implementation and validation are in Appendix K.

Table 5 shows that neither channel is sufficient on its own. As shown in the attention-only row, clean attention over corrupted content is significantly negative in three of four models: a correctly aimed attention pattern still retrieves the wrong token identities, so directing the model at corrupted positions makes matters worse. Clean embeddings under a corrupted attention pattern reach significance in only one model, which places the 26% recovered by embedding replacement in context as what a content-side intervention achieves while attention remains misallocated. Restoring both together recovers 45–71% of the gap in the three models where the effect is significant, with a significantly positive interaction in every model.

The channels are thus strongly super-additive. Fragmentation does not damage token identity first and divert attention afterwards. It damages both at once, and attention is useful only when the content it addresses is intact. This also explains why suppressing corrupted-token attention degrades accuracy, and why every single-channel mitigation in Section 6 fails.

Table 6: **Predictors of correct-to-wrong flips on GSM8K** (n=237, four models, model fixed effects, standard errors clustered by prompt). Continuous predictors are standardized. OR is reported with its 95% confidence interval.

Predictor	**OR** [95% CI]	p
Numeric-token hit	**12.89** [5.06, 32.85]	8.4×10^{-8}
Corrupted fraction	1.41 [0.58, 3.45]	0.45
Diversion	1.45 [0.29, 7.24]	0.65

5.4 What Determines Task Resilience

Section 4 shows that diversion magnitude does not predict task-level damage. We now test what does. If resilience depends on whether displaced evidence is recoverable, then numeric content should be the vulnerable case in mathematical reasoning. A quantity appears once, determines the answer, and cannot be inferred from surrounding text. A corrupted content word often can.

We fit a joint logistic regression over GSM8K clean-correct records (n=237 observations, four models, typoswap at r=0.1), modelling the correct-to-wrong flip. The numeric-token hit, the corrupted-token fraction, and the diversion ratio enter simultaneously, with model fixed effects and standard errors clustered by prompt (Table 6). Whether corruption lands on a numeral is the only significant predictor. The flip rate is 0.77 when a numeral is corrupted and 0.33 otherwise. Neither the corrupted-token fraction nor the diversion ratio retains independent predictive power once prompt-level dependence is accounted for.

Table 7: **GSM-IC targeted fragmentation** (four models, n=400, typoswap r=0.1, standard errors clustered by prompt). Length and semantics are held fixed, and only the fragmented span differs. Diversion occurs in both conditions and only one is harmful.

Fragmented span	Drop (pt)	95% CI
Reasoning-relevant	21.0	[14.5, 28.0]
Irrelevant (injected)	2.5	[−1.2, 6.2]
Contrast	18.5	[12.2, 25.0]

The regression is observational, in that which words the perturbation lands on is left to chance. To manipulate the target directly we use GSM-IC (Shi et al., 2023), which augments GSM8K problems with irrelevant injected sentences. Within each example we construct variants that differ only in *what* is fragmented, holding length and semantics fixed: either the reasoning-relevant content, or an equal-length irrelevant injected span. Table 7 reports the result. Only the reasoning-relevant condition is harmful, and the contrast between the two is significant ($p < 10^{-4}$), although diversion occurs in both.

Attention Diversion is therefore harmful in proportion to the irreplaceability of the evidence it displaces, which explains why the per-task diversion ordering does not track the accuracy-loss ordering (Appendix C): GSM8K suffers the largest loss despite sustaining less diversion than BoolQ, because its evidence is the least redundant. It also connects to the coupling result of Section 5.3. In GSM8K the evidence is a quantity that appears once, so displacing attention from it and directing attention to a corrupted version of it are equally fatal, which is why restoring clean attention over corrupted content does not help: attention correctly aimed at a corrupted numeral still retrieves the wrong value.

Cross-perturbation controls refine the fragmentation account. The account predicts harm mainly when perturbations create rare or out-of-distribution subword fragments. Table 8 tests this with ASR homophones and OCR character confusions. ASR substitutions produce zero fragmentation and leave accuracy at baseline. OCR produces the *highest* fragmentation of the three perturbations yet still leaves accuracy near baseline, because its fragments follow familiar subword patterns. Only typoswap, whose fragments are out of distribution, substantially reduces GSM8K accuracy. Fragmentation is thus necessary but not sufficient: what matters is whether the resulting pieces are ones the model has seen. Chinese CMath shows the same qualitative pattern (Appendix L).

Table 8: **Cross-perturbation controls** (Llama-3.3-70B, English BoolQ + GSM8K, $n = 100$, separate subset from Appendix D). Rows ordered by increasing fragmentation. Fragmentation is shaded by magnitude (≥0.08 , 0.03–0.08) and accuracy by degradation relative to clean (≥0.15 , 0.05–0.15). OCR fragments most yet leaves accuracy intact. Typoswap alone damages GSM8K.

	Frag.		Acc.	
Perturb.	BQ	GSM	BQ	GSM
Clean	.000	.000	.88	.95
ASR (r=0.1)	.000	.000	.91	.93
ASR (r=0.2)	.000	.000	.89	.94
Keyboard (r=0.1)	.049	.035	.89	.90
Typoswap (r=0.1)	.062	.048	.89	**.79**
OCR (r=0.1)	.086	.061	.89	.91
OCR (r=0.2)	.181	.154	.91	.93

Qualitative case study. A GSM8K example in Appendix M illustrates the full pathway: keyboard perturbation expands the prompt from 48 to 63 tokens, redirects 67% of attention mass to corrupted fragments, and changes the model output from the correct answer 14 to a copied surface number 4.

Together, these analyses support a consistent mechanism. Lexical corruption fragments tokenization, which simultaneously alters the token identities the model reads and redirects attention onto the fragments. The factorial intervention shows the two to be inseparable: correcting either alone leaves accuracy at or below the corrupted baseline, while both together recover most of the loss.

6 Can Inference-Time Strategies Mitigate Lexical Fragility?

If lexical fragility arises from the coupled disruption of Section 5.3, any strategy acting on one channel alone should recover little. We investigate chain-of-thought prompting, prompt-level repair, a 70B repair model, and a tokenizer-level defense. Because these experiments used different subsets and templates, each is reported against its own paired baseline and clean anchor. Recovery is a percentage of the clean–corrupted gap within each comparison.

Prompt-level repair trades one failure for another. Table 9 evaluates every mitigation on a common recovery scale. Spell-checking is the only

Table 9: **Mitigation ledger.** Recovery as a percentage of each arm's own clean–corrupted gap. Every arm is paired against a baseline on its own subset and prompt, so accuracies are not comparable across blocks. Grey marks recovery and purple harm. No strategy is positive in more than two of four conditions.

	BoolQ		GSM8K	
Strategy	KB	TS	KB	TS
Prompt-level repair — Llama-3.1-8B, n=100, r=0.1				
Spell-check	+30	−10	+13	−29
Noise-aware	−150	−80	−25	−50
Self-repair	−600	−620	+50	0
clean .83/.81, corrupted .73/.73/.65/.67				
Stronger rewriter — Llama-3.3-70B-Instruct				
70B repair	+114*	0	0	0
clean .79/.69, corrupted .72/.71/.48/.49				

*Post-repair accuracy (0.80) exceeds the clean anchor (0.79), so recovery exceeds 100%. The excess is within sampling noise.

Tokenizer-level defense (4 models, GSM8K and BoolQ, typoswap r=0.1, n=100/cell). Detection reaches F1 0.77–0.81 without clean text, and fragmentation alone reaches 0.70–0.75. Repair restores the original word 67–70% of the time at a false-repair rate of 22–25%. Recovery is significantly positive in none of the eight cells and significantly negative in one, with no clean-input damage. **Oracle** (clean token IDs restored). Full recovery on all models.

arm positive under keyboard noise on both tasks, recovering 30% of the gap on BoolQ and 13% on GSM8K, yet it turns negative under typoswap on both, where transposed characters yield fluent but incorrect dictionary matches. Noise-aware prompting is negative in all four conditions. Self-repair recovers half the gap on GSM8K under keyboard noise but destroys BoolQ, dropping it from 0.73 to 0.13 by rewriting passage content the question depends on. Substituting Llama-3.3-70B-Instruct as the rewriter recovers the BoolQ keyboard condition fully and leaves the other three unchanged, so the failure is not a matter of rewriter capacity. Correcting the surface form does not restore the token identities the reasoning depends on. Reasoning-time prompting fares no better. CoT lifts clean GSM8K on GPT-4o from 0.42 to 0.90, yet typoswap still pulls it to 0.70 (Appendix A.1), and GPT-5.4's built-in reasoning mode *amplifies* the degradation rather than absorbing it (Appendix A). Attention calibration and prompt restoration, evaluated separately, likewise show no consistent improvement (Appendix N).

6.1 A Tokenizer-Level Defense

The strategies above all operate on the prompt. We now test whether the damage can be addressed where it originates, bounding the problem from both ends: an oracle with access to the clean text, and a deployable defense without it. Headline numbers are given at the foot of Table 9. The protocol is in Appendix O.

An oracle ceiling. Replacing each corrupted word's fragmented tokens with the clean word's exact token IDs restores accuracy to the clean baseline on all four models (GSM8K, typoswap r=0.1, n=100). This is a by-construction ceiling rather than a method, since reconstructing the clean token sequence reconstructs the clean input. Its value is diagnostic: had the perturbation degraded the model's reasoning rather than its input, supplying the correct tokens would not have restored performance. It did, so the reasoning is intact and the damage is confined to the token sequence.

A deployable defense. We then built a defense under the constraint that makes the result interpretable. The detector and the repair observe only the corrupted input, with no access to the clean text or to the perturbation mask, which is used solely for post-hoc scoring. Detection combines rarity, surprisal, and fragmentation. Repair uses vocabulary-constrained edit-distance correction, with a high-precision detector variant and a conservative variant that declines when no confident candidate exists. Thresholds were selected on a held-out split disjoint from the evaluation subset. Character-level fallback and greedy retokenization were implemented but not deployed, because they rewrite every flagged word and so inherit the detector's false-repair rate of 22–25%, the failure mode that drives self-repair from 0.73 to 0.13 on BoolQ.

Detection succeeds while repair fails. Fragmentation alone identifies corrupted words at F1 0.70–0.75 with no access to the clean text, which shows it is a reliable signature of corruption and not only a consequence of it. Repair does not follow: vocabulary-constrained correction recovers the original word only 67–70% of the time, and the substitutions that are confident but incorrect do more harm than the fragments they replace, since a fragment at least signals anomaly whereas a fluent substitution silently alters the semantics. Recovery is significantly positive in none of the eight model–task cells and significantly negative in one. The defense does not damage clean inputs, unlike self-repair, but it does not help either.

The binding constraint is informational. Numeric tokens carry the largest effect on GSM8K failure (Section 5.4), yet a corrupted numeral re-

mains a valid numeral: it is not out of vocabulary, it is too short to fragment, and it admits no dictionary neighbour, so no detector operating on the corrupted input can find it and no corrector can recover the original quantity. Taken with the oracle result, this locates the problem precisely. The damage is localized at the input representation, since reconstructing the clean token sequence restores accuracy in full. But it is irreversible without the clean text, because the corruptions that dominate failure leave no recoverable trace.

Summary. No strategy recovers consistently, matching the coupling of Section 5.3: each acts on one channel while the other remains corrupted, and the corruptions that matter most leave no recoverable trace in the corrupted input.

7 Related Work

Prior work established the phenomenon: corrupted fragments attract disproportionate attention. Our factorial intervention shows that this attention cannot be manipulated on its own, a finding that organizes three lines of research, each locating the remedy elsewhere. Typo-attack work locates it at the input, restoring the surface form before the model reads it (Belinkov and Bisk, 2018; Pruthi et al., 2019; Li et al., 2019; Ebrahimi et al., 2018; Jia and Liang, 2017). Such correction is bounded by what the corrupted input still encodes, and a perturbed numeral stays in vocabulary with no dictionary neighbour (Section 6.1). Attention-sink work locates it in the attention pattern, suppressing positions that absorb attention regardless of content (Xiao et al., 2024). The sinks we observe are created by the perturbation rather than fixed by the model, and suppressing them costs up to 0.31 on GSM8K (Appendix J). Subword regularization locates it in training (Provilkov et al., 2020), the only remaining option once fragmentation has occurred and the information needed to reverse it is gone. Appendix P develops these comparisons.

8 Conclusion

This paper studies how realistic lexical perturbations disrupt LLM reasoning across four benchmarks, four open-weight model families, and frontier systems. Character-level noise fragments subword tokenization, which alters the token identities the model reads and redirects attention onto the fragments at once, while length-matched filler stays benign. Controlled experiments, per-example regression, and representation-level interventions identify fragmentation rather than prompt length as the driver, and a factorial intervention shows the two channels to be coupled rather than sequential. This coupling explains why no inference-time strategy recovers consistently, and why harm tracks what the corruption lands on rather than how much attention it diverts. The damage is localized at the input representation yet irreversible without the clean text, so robustness must be built into the representation before fragmentation occurs rather than repaired afterwards. Character- and byte-aware fallback representations, dynamic retokenization, and training-time alignment all act at that point, and are the directions we see as most promising.

Limitations

We note several boundaries. First, the main analyses focus on English with a QWERTY layout, though we extend the protocol to Chinese (CMath) and OCR/ASR controls. Broader coverage of languages and input methods remains open. Second, the perturbations are synthetic, approximating common noisy-input phenomena rather than sampled from user data. Third, all benchmarks are standard-length single-turn tasks. Long-context and agentic settings introduce position and interaction effects that would need separate control. Fourth, the factorial intervention is run on GSM8K at a single severity, since BoolQ's narrow gap makes the recovery ratio ill-conditioned, so the coupling result rests on one task. Fifth, our interventions are diagnostic and do not identify the internal circuits responsible for failure. Finally, we study open-weight models in the 7B–70B range and frontier configurations, and other recipes or scales may differ.

Ethical Considerations

This work does not involve human subjects, sensitive personal data, or any proprietary datasets. All datasets used are publicly available and commonly used in prior research. We have taken care to ensure that our methods and results do not raise safety, privacy, or fairness concerns.

AI Assistance Disclosure

AI assistance was used only for language polishing and was not used to generate experimental results or analyses. The authors verified all scientific claims, experiments, analyses, and final text.

References

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations (ICLR)*.

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019a. 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)*, pages 2924–2936.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019b. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, and 1 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Gemma Team. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5484–5495.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2021–2031.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. TextBugger: Generating adversarial text against real-world applications. In *Network and Distributed System Security Symposium*.

Llama Team, AI @ Meta. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098.

Mistral AI. 2024. Mistral-7B-Instruct-v0.3. `https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3`. Model card.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

OpenAI. 2024. Hello GPT-4o. `https://openai.com/index/hello-gpt-4o/`.

OpenAI. 2026. Introducing GPT-5.4. `https://openai.com/index/introducing-gpt-5-4/`.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-Dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5582–5591.

Qwen Team. 2026. Qwen3.5-9B. `https://huggingface.co/Qwen/Qwen3.5-9B`. Model card.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting. In *International Conference on Learning Representations (ICLR)*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Tianwen Wei, Jian Luan, Wei Liu, Shuang Dong, and Bin Wang. 2023. CMATH: Can your language model pass Chinese elementary school math test? *arXiv preprint arXiv:2306.16636*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*.

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

Appendix

A Frontier LLM Robustness Evaluation

To examine whether the lexical robustness patterns observed in open-weight models extend to proprietary frontier systems, we evaluate GPT-4o (OpenAI, 2024), GPT-5.4 with `reasoning_effort=none` ("Standard"), and GPT-5.4 with `reasoning_effort=high` ("Thinking"). Without access to internal representations such as tokenization states or attention weights, these models are evaluated only at the prediction level. The protocol follows the perturbation procedure and prompt format of the main experiments on a subset of 100 examples per dataset, with two representative perturbation types (keyboard noise and typoswap) at severity $r = 0.1$.

A.1 Chain-of-Thought Prompting

Table 10 evaluates direct versus CoT prompting on GPT-4o (r=0.1, n=100). CoT lifts clean GSM8K from 0.42 to 0.90, confirming that the model's reasoning capability is intact, yet typoswap still pulls it to 0.70. On the other three benchmarks CoT provides no consistent advantage over direct prompting, indicating that stronger reasoning does not repair corrupted representations. The following sections extend this to built-in reasoning.

Table 10: **Direct vs. CoT prompting** (GPT-4o, r=0.1, n=100). CoT lifts clean GSM8K from 0.42 to 0.90 but leaves the corruption gap intact. Cells are shaded by degradation relative to each mode's own clean accuracy: ≥0.15, 0.05–0.15.

	Clean		Keyboard		Typoswap	
Task	Direct	CoT	Direct	CoT	Direct	CoT
BoolQ	.90	.87	.82	.84	.86	.88
PIQA	.91	.92	.76	.85	.92	.90
HellaSwag	.86	.81	.80	.74	.80	.81
GSM8K	.42	**.90**	.50	.86	.40	**.70**

A.2 GPT-4o Results

Table 11 presents results for GPT-4o. GPT-4o achieves substantially higher clean accuracy than the open-weight models across all four benchmarks, yet still exhibits consistent performance degradation under character-level perturbations. Under keyboard noise, the accuracy drops range from 4 to 14 points depending on the task, while typoswap perturbations produce larger degradation on GSM8K (−18 points) but minimal impact on PIQA.

Table 11: **GPT-4o robustness under lexical perturbations** ($n = 100$, $r = 0.1$). Drop relative to clean in parentheses. Shading: ≥0.15, 0.05–0.15.

Task	**Cln**	**Keyboard**	**Typoswap**
BoolQ	.90	.82 (-.08)	.85 (-.05)
PIQA	.92	.78 (-.14)	.92 (+.00)
HellaSwag	.84	.78 (-.06)	.83 (-.01)
GSM8K	.85	.81 (-.04)	.67 (-.18)
Avg.	*.878*	*.798 (-.080)*	*.818 (-.060)*

A.3 GPT-5.4 Standard and Thinking Results

The two GPT-5.4 configurations use the same underlying model through the OpenAI API and differ only in `reasoning_effort`. Standard mode (`reasoning_effort=none`) disables chain-of-thought reasoning, while thinking mode (`reasoning_effort=high`) enables extended internal reasoning before producing a response, so the comparison directly tests whether built-in reasoning mitigates lexical fragility.

Table 12 presents the results. Standard mode achieves the highest clean accuracy among all frontier models (0.908 average) and degrades by −0.093 under keyboard noise, comparable to GPT-4o (−0.080). Critically, **thinking mode amplifies lexical fragility rather than mitigating it.** Its average keyboard degradation of −0.147 is roughly 1.6× standard mode's −0.093 and 1.8× GPT-4o's −0.080, with the most severe drops on HellaSwag (−0.23 under keyboard noise) and GSM8K (−0.23 under typoswap). This extends the CoT analysis in Appendix A.1. Reasoning-time strategies, whether explicit CoT prompting (Table 10) or built-in model reasoning, cannot compensate for input-level tokenization disruption, and the extended reasoning chain appears to *propagate* early tokenization errors through more computation steps rather than correcting them.

A.4 Cross-Model Comparison

Table 3 in the main text and Figures 4–5 compare all frontier configurations with the open-weight models. Three observations emerge. First, GPT-5.4 Standard achieves the highest clean accuracy

Table 12: **GPT-5.4 robustness under lexical perturbations** ($n = 100$, $r = 0.1$). Drop relative to each mode's own clean accuracy in parentheses. Thinking mode amplifies fragility, with the largest average keyboard drop (-0.147) among all frontier configurations. Shading: ≥0.15, 0.05–0.15.

Task	Cln	Keyboard	Typoswap
GPT-5.4 (Standard)			
BoolQ	.88	.77 (-.11)	.87 (-.01)
PIQA	.95	.79 (-.16)	.88 (-.07)
HellaSwag	.83	.77 (-.06)	.86 (+.03)
GSM8K	.97	.93 (-.04)	.82 (-.15)
Avg.	*.908*	*.815 (-.093)*	*.858 (-.050)*
GPT-5.4 (Thinking)			
BoolQ	.89	.70 (-.19)	.86 (-.03)
PIQA	.91	.83 (-.08)	.90 (-.01)
HellaSwag	.86	.63 (-.23)	.78 (-.08)
GSM8K	.96	.87 (-.09)	.73 (-.23)
Avg.	*.905*	*.758 (-.147)*	*.818 (-.087)*

(0.908) but its corruption robustness does not proportionally improve, suggesting that raw capability gains do not resolve tokenizer-level fragility. Second, GPT-5.4 Thinking, despite near-identical clean accuracy to standard mode, exhibits the *largest* keyboard degradation (-0.147) among all frontier systems, confirming that reasoning amplifies rather than compensates for input-level disruption. Third, the task-sensitivity pattern is preserved under typoswap across all frontier configurations: GSM8K sustains the largest drop in every case (-0.15 to -0.23). Under keyboard noise the ordering differs, with PIQA or HellaSwag taking the largest drop, suggesting that the two perturbation types stress different task properties in frontier models.

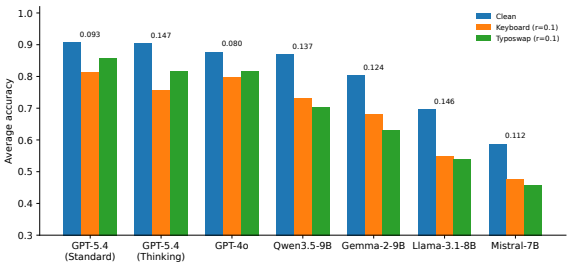

Figure 4: **Cross-model comparison under lexical corruption at $r = 0.1$.** GPT-5.4 Standard achieves the highest clean accuracy. GPT-5.4 Thinking shows the largest degradation among frontier models (-0.147 under keyboard noise), confirming that reasoning amplifies rather than mitigates lexical fragility. Numbers indicate the absolute drop under keyboard noise relative to clean prompts.

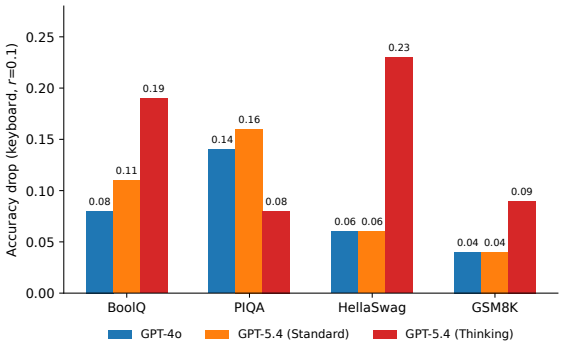

Figure 5: **Task-level accuracy drop under keyboard noise at $r = 0.1$ for frontier models.** GPT-5.4 Thinking exhibits the largest degradation on HellaSwag (0.23) and BoolQ (0.19), while GPT-5.4 Standard is most robust on GSM8K (0.04). Error patterns differ across reasoning modes despite near-identical clean performance.

B Full Robustness Results at All Severity Levels

Table 13 presents detailed accuracy results at severity levels $r \in \{0.05, 0.20, 0.30\}$, complementing the $r = 0.1$ results in Table 2 in the main text.

C Layer-wise diversion ratio by Task

Figure 6 presents the per-task layer-wise diversion ratios for Llama-3.1-8B. The bimodal diversion pattern observed in the cross-task average (Figure 3) is consistently reproduced across all four reasoning benchmarks. The task ordering is model-dependent at the top. HellaSwag and BoolQ show the most pronounced peaks, while GSM8K and PIQA are consistently lower. GSM8K therefore sustains less diversion than BoolQ despite losing far more accuracy, which is the dissociation analysed in Section 5.4. Per-model orderings for the other three models are consistent and omitted for space. These results confirm that the layer-wise distribution of Attention Diversion is a robust architectural property rather than a task-specific artifact.

D Generalization Across Scale, Difficulty, and Language

To verify that the patterns observed on 7–9B open-weight models generalize, we evaluated Llama-3.3-70B-Instruct on three additional settings at $r = 0.1$ ($n = 100$). Section 4 summarizes these results. This appendix provides per-task details and qualitative observations.

70B-scale (English). Llama-3.3-70B was queried via the Together AI API in greedy

Table 13: **Accuracy at severity** $r \in \{0.05, 0.20, 0.30\}$ ($n = 1{,}000$). Shading marks degradation relative to clean: ≥0.30 , 0.15–0.30 , 0.05–0.15 . Three levels are used here because the range is wider than in the main text. No Fil cell reaches the 0.05 threshold at any severity.

		$r = 0.05$				$r = 0.20$				$r = 0.30$			
Model	**Task**	**Cln**	**KB**	**TS**	**Fil**	**Cln**	**KB**	**TS**	**Fil**	**Cln**	**KB**	**TS**	**Fil**
Gemma-2-9B-IT	BoolQ	.875	.830	.857	.875	.875	.737	.787	.878	.875	.631	.686	.874
	PIQA	.843	.736	.725	.833	.843	.584	.566	.827	.843	.504	.495	.828
	HellaSwag	.693	.520	.528	.686	.693	.386	.383	.687	.693	.323	.320	.685
	GSM8K	.804	.775	.692	.807	.804	.575	.349	.797	.804	.351	.179	.809
Llama-3.1-8B	BoolQ	.834	.776	.809	.839	.834	.622	.672	.834	.834	.518	.622	.832
	PIQA	.758	.653	.677	.761	.758	.482	.527	.755	.758	.505	.502	.753
	HellaSwag	.493	.398	.398	.502	.493	.289	.281	.493	.493	.233	.246	.489
	GSM8K	.698	.614	.542	.697	.698	.341	.167	.697	.698	.160	.071	.695
Mistral-7B	BoolQ	.847	.785	.795	.848	.847	.638	.613	.837	.847	.557	.579	.843
	PIQA	.693	.638	.620	.692	.693	.459	.479	.697	.693	.454	.479	.704
	HellaSwag	.413	.363	.367	.413	.413	.297	.267	.414	.413	.263	.250	.413
	GSM8K	.397	.329	.271	.414	.397	.151	.078	.408	.397	.062	.035	.419
Qwen3.5-9B	BoolQ	.878	.845	.837	.877	.878	.688	.685	.875	.878	.549	.598	.873
	PIQA	.892	.805	.801	.893	.892	.608	.604	.881	.892	.494	.541	.889
	HellaSwag	.863	.770	.808	.857	.863	.539	.507	.852	.863	.360	.370	.860
	GSM8K	.845	.760	.692	.836	.845	.461	.338	.842	.845	.191	.163	.840

Figure 6: **Layer-wise diversion ratio by task for Llama-3.1-8B.** The bimodal pattern with peaks at middle layers (11–15) and final layers (30–32) is consistent across all four reasoning benchmarks. BoolQ and HellaSwag show the strongest diversion, while GSM8K exhibits lower absolute values but the same structural pattern.

decoding mode. The fragility pattern is preserved: GSM8K drops 13 points under typoswap, while BoolQ remains stable. Keyboard noise has near-zero effect at this scale (−0.01 on both tasks), while typoswap remains substantial (−0.13 on GSM8K), indicating that scale partially mitigates the milder character-substitution perturbations but does not eliminate the more aggressive transposition pattern.

Harder mathematics (MATH algebra). We evaluated Llama-3.3-70B on the algebra subset of the MATH benchmark (Hendrycks et al., 2021) at $r = 0.1$ ($n = 100$). Clean accuracy is 0.68, substantially below GSM8K (0.92), reflecting the higher difficulty. The same qualitative ordering holds: typoswap produces the largest drop (−0.08, to 0.60), keyboard noise produces a moderate drop (−0.06, to 0.62), filler is nearly harmless (−0.04,

to 0.64). Extremely difficult benchmarks such as AIME or HMMT can confound this signal when clean accuracy approaches the floor. MATH algebra provides a non-saturated harder regime in which the fragmentation effect remains detectable.

Non-Latin script (Chinese, CMath). We adapted the perturbation rules to operate on Chinese characters. Keyboard perturbation replaces a character with a near-Unicode-codepoint neighbor, where Unicode adjacency serves as a structural analog of QWERTY adjacency. Typoswap swaps two adjacent characters. Filler inserts Chinese disfluency phrases (transliterated *"e"* or *"en"*, the Chinese equivalents of *"um"* or *"uh"*) at clause boundaries. On the CMath elementary-school math benchmark, Llama-3.3-70B achieves 0.94 / 0.64 / 0.81 / 0.95 under clean / keyboard / typoswap / filler at $r = 0.1$ ($n = 100$). The keyboard drop of 30 points reproduces the English pattern, filler is harmless, and typoswap is intermediate. The mechanism is not specific to Latin scripts.

E Failure Pathway Overview

Figure 7 illustrates the complete processing pathway under clean and corrupted inputs. The four stages of prompt composition, tokenization, attention allocation, and reasoning are contrasted to show how lexical corruption cascades through the model: corrupted characters trigger tokenizer fragmentation (b), with attention mass reallocating toward noise tokens (c), and reasoning failing in the output stage (d).

F Tokenization Disruption and Error-Conditioned Evidence

Figure 8 reports the two tokenization effects summarized in Section 5: character-level perturbations increase token count with severity while filler remains near the clean baseline, and keyboard noise and typoswap reduce token overlap with the clean prompt.

Figure 9 conditions the same diagnostics on the outcome, comparing examples that remain correct under perturbation (*correct→correct*) against those that flip (*correct→wrong*). Failure cases show higher diversion ratios, lower token overlap, and larger token-count inflation, most clearly at low-to-moderate corruption levels, linking tokenization disruption and Attention Diversion to prediction failures rather than to aggregate statistics alone.

G Cross-Tokenizer Diversion Analysis

To examine whether tokenizer design affects diversion magnitude, we report per-model diversion ratios at three severities for keyboard noise and typoswap on Llama-3.1-8B, Mistral-7B-v0.3, Gemma-2-9B-IT, and Qwen3.5-9B (Figure 10 and Table 14).

Three observations emerge. First, **vocabulary size is not the determinant**. Gemma's 256K vocabulary is roughly 8× Mistral's 33K, yet Gemma's diversion is consistently higher. Second, **the ranking Qwen > Gemma > Mistral > Llama holds across all eight reported conditions** (4 severities × 2 perturbation types) under the all-layers diversion metric used throughout the paper, supporting an architecture or tokenizer-level effect rather than a sampling artifact. Third, we hypothesize that tokenizer training coverage and merge-rule differences (multilingual vs. English-dominant corpora) may affect how aggressively English text is split when characters are perturbed. Qwen's high diversion ratio is also consistent with its large GSM8K typoswap drop in the main results (Table 2).

H Per-Example Regression Analysis

Section 5 reports a logistic regression quantifying how each input-level signal predicts the *correct-to-wrong* flip. This appendix provides the full regression results, dataset construction, and effect sizes.

Data construction. We aggregated per-example records from our four open-weight models across all four benchmarks, four character-level perturbation rates ($r \in \{0.05, 0.10, 0.20, 0.30\}$), and two perturbation types (keyboard, typoswap). We restricted the analysis to examples that the model solves correctly under clean input ($n = 4{,}160$ total), since these are the cases where the perturbation alone determines the flip. The binary outcome is $\mathsf{failure} = 1$ if the clean-correct example becomes wrong under perturbation.

Predictors. Each example carries four standardized predictors: fragmentation ratio frag, attention diversion ratio diversion (averaged over all attention layers, matching the metric used in the main text), absolute token-count change n_delta, and token overlap overlap.

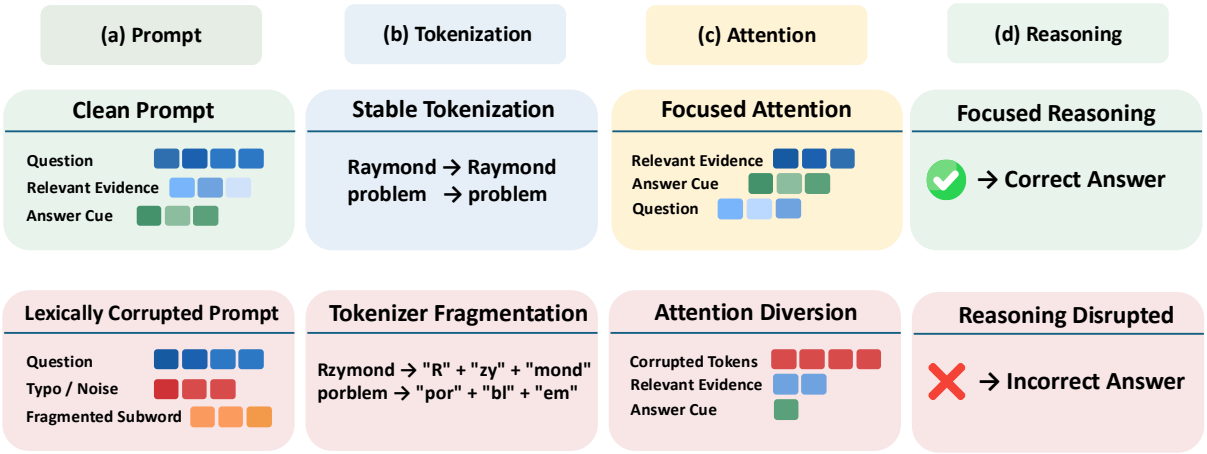

Figure 7: **Complete failure pathway under lexical corruption.** Top: clean input proceeds through stable tokenization, focused attention, and correct reasoning. Bottom: lexical corruption fragments tokens into novel subword pieces, the fragments absorb a disproportionate share of attention, and reasoning fails.

Table 14: **Cross-tokenizer diversion ratio at four severities.** The ranking Qwen > Gemma > Mistral > Llama is consistent across both perturbation types and all severities, supporting an architecture/tokenizer-level effect rather than a sampling artifact. Shading marks magnitude only (≥0.20 , 0.10–0.20) and is deliberately distinct from the degradation shading used elsewhere: Section 5.4 shows that diversion magnitude does not track accuracy loss.

		Keyboard (r=)				Typoswap (r=)			
Model	**Tokenizer (Vocab)**	0.05	0.10	0.20	0.30	0.05	0.10	0.20	0.30
Llama-3.1-8B	tiktoken BPE (128K)	0.046	0.077	0.118	0.144	0.061	0.101	0.148	0.172
Mistral-7B-v0.3	SentencePiece (33K)	0.058	0.096	0.146	0.176	0.075	0.122	0.181	0.211
Gemma-2-9B-IT	SentencePiece (256K)	0.090	0.146	0.215	0.256	0.118	0.186	0.263	0.299
Qwen3.5-9B†	tiktoken BPE (152K)	0.146	0.241	0.367	0.448	0.190	0.307	0.451	0.527

†Qwen3.5-9B uses a hybrid Gated-DeltaNet/Attention architecture; the comparison conflates tokenizer with architecture, though the same ordering holds among the three pure transformers.

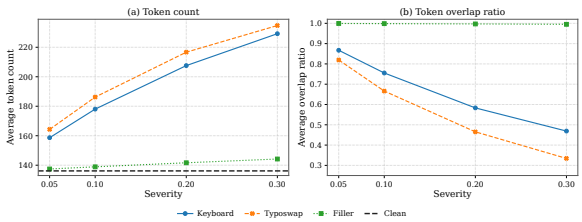

Figure 8: **Tokenization disruption under lexical perturbations.** (a) Character-level perturbations increase token count with severity, while filler remains near the clean baseline. (b) Keyboard noise and typoswap reduce token overlap with clean prompts, indicating token identity disruption.

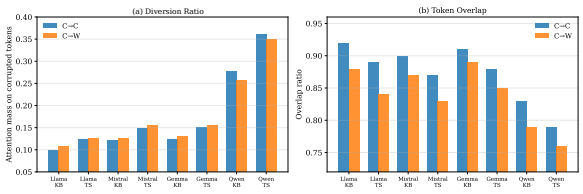

Figure 9: **Error-conditioned evidence for lexical fragility.** Failed cases (C→W) exhibit higher diversion ratios and lower token overlap than cases that remain correct (C→C).

Univariate effects. Table 15 reports the univariate logistic regression coefficients with 95% confidence intervals. Fragmentation is the strongest predictor by an order of magnitude.

Joint model controlling for length. To test whether fragmentation's effect is a proxy for prompt-length inflation, we regress on fragmentation and token-count change jointly (Table 16). Fragmentation's coefficient *strengthens* from β=0.66 to β=1.04 after controlling for length (OR 1.93 to 2.84), while token-count change loses its independent predictive effect.

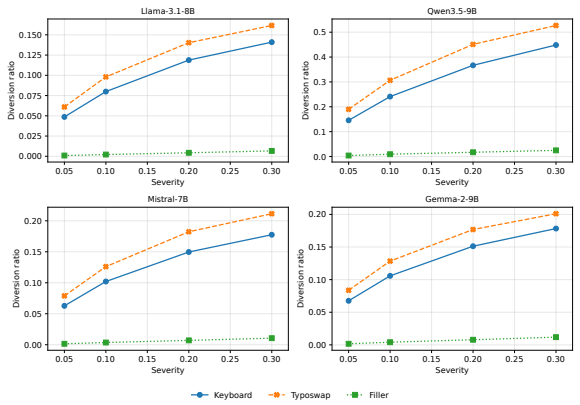

Figure 10: **Attention redistribution toward corrupted tokens.** Diversion ratio under increasing corruption severity across four models, computed over all attention layers. Character-level perturbations consistently increase corrupted-token attention, with typoswap producing stronger diversion than keyboard noise. Filler insertion stays below 0.025 throughout. Table 14 gives the same quantities numerically.

Table 15: **Univariate logistic regression.** Each predictor's standalone effect on $P(\text{failure})$. β is the log-odds coefficient per 1 SD of the predictor, and OR is the corresponding odds ratio. All effects are highly significant.

Predictor	β	**OR**	p
fragmentation	$+0.66$	1.93	$< 10^{-81}$
overlap	-0.54	0.58	$< 10^{-58}$
n_delta	$+0.25$	1.29	$< 10^{-14}$
diversion	$+0.23$	1.26	$< 10^{-12}$

Summary. These results provide statistical confirmation of the controlled fragmentation experiment in Section 5: among input-level signals, fragmentation is the strongest and most robust predictor of reasoning failure.

I Embedding-Replacement Intervention Protocol

Section 5 reports an embedding-replacement intervention that directly tests the causal role of fragmented-token embeddings. This appendix gives the full protocol and per-example diagnostics.

Setup. We use Llama-3.1-8B-Instruct loaded in bfloat16 on a single GPU. We evaluate on 100 GSM8K examples randomly sampled from the test split with seed 42, under keyboard perturbation at $r = 0.1$.

Procedure. For each example we run three forward passes. The **clean baseline** encodes the clean prompt, decodes greedily, and records accuracy.

Table 16: **Joint logistic regression**: fragmentation and token-count change as simultaneous predictors of failure. Fragmentation strengthens when length is controlled, demonstrating that the two signals are not interchangeable.

Predictor	β	**OR**	p
fragmentation	$+1.04$	2.84	$< 10^{-88}$
n_delta	-0.50	0.61	$< 10^{-23}$

Table 17: **Embedding-replacement intervention on GSM8K** (Llama-3.1-8B, keyboard $r = 0.1$, $n = 100$).

Condition	**Accuracy**
Clean baseline	0.80
Corrupted baseline	0.42
Embedding replacement	**0.52**
Absolute recovery	$+0.10$
Fraction of gap recovered	26%
Mean embeddings replaced	2.95

The **corrupted baseline** does the same on the corrupted prompt. The **embedding-replaced intervention** encodes the corrupted prompt to obtain its token-embedding sequence $\mathbf{e}^{\text{corr}}_{1:L_c}$ and the clean prompt to obtain $\mathbf{e}^{\text{clean}}_{1:L_d}$. For each corrupted token position i whose token ID does *not* appear in the clean sequence within a ± 5-position window, we replace its embedding with the most-similar clean-position embedding (cosine similarity > 0.3 within the window). The resulting hybrid sequence is passed via `inputs_embeds` to greedy decoding.

Results. Table 17 reports accuracy and the average number of token embeddings replaced per example. On average 2.95 embeddings are replaced out of a corrupted-prompt length of ≈ 75, and accuracy recovers from 0.42 to 0.52, restoring 26% of the 0.38 gap to clean accuracy (0.80).

Interpretation. The intervention is conservative: it modifies only embeddings whose token IDs differ from the clean sequence within a small local window, and matches each replacement by embedding cosine similarity. Recovering 26% of the accuracy gap with $\sim$3 embedding edits per example provides direct causal evidence that fragmented-token embeddings carry the load-bearing representational signal that disrupts reasoning, independent of any downstream attention dynamics. This local cosine-matching procedure is the one used for the round-1 intervention reported in Section 5. The factorial experiment in Section 5.3 uses the difflib alignment described in Appendix K for both of its arms.

J Attention Suppression Intervention

Section 5 reports that suppressing attention to outlier tokens worsens rather than improves accuracy. For each token j we compute an outlier score $s_j = \|h_j - \bar{h}\|_2$ from the hidden state and rescale attention as $\tilde{A}_{ij} = A_{ij} \cdot \sigma(-\lambda s_j)$, sweeping $\lambda \in \{0.5, 0.9\}$ on Llama-3.1-8B (n=200). Figure 11 shows that stronger suppression consistently degrades accuracy across tasks, with GSM8K dropping up to 0.31. Corrupted-token attention is therefore load-bearing rather than passive noise. The factorial intervention of Section 5.3 explains why: removing attention from corrupted positions leaves the model with neither the corrupted evidence nor any substitute for it.

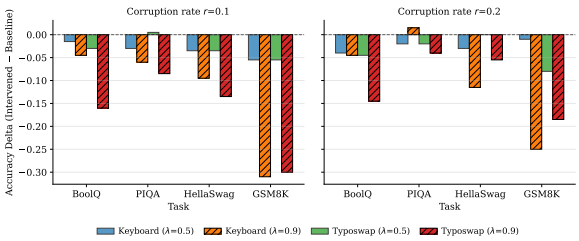

Figure 11: **Intervention via attention suppression.** Suppressing corrupted-token attention degrades accuracy, indicating that it is load-bearing rather than passive noise (Llama-3.1-8B, n=200).

K Factorial Intervention Protocol

Section 5.3 reports a factorial intervention that crosses the source of the attention pattern with the source of the token embeddings. This appendix gives the implementation and the validation checks.

Setup. Four open-weight models are evaluated on GSM8K and BoolQ under typoswap at r=0.1, with n=100 examples per model–task cell drawn from a fixed seed-42 subset, bfloat16 weights, eager attention, and greedy decoding. Confidence intervals are estimated using a paired cluster bootstrap over example IDs (B=2000, percentile method), with the interaction term computed on the recovery scale within each replicate. Main-text results report GSM8K. BoolQ's clean–corrupted gap of 4–10 points makes the recovery ratio ill-conditioned, so we do not decompose it.

Position alignment. Clean and corrupted token-ID sequences are aligned with `difflib.SequenceMatcher`, which yields a map from each corrupted position to a clean counterpart. Within a replaced block, a clean token aligned to several corrupted fragments maps to each of them. Where a corrupted fragment has no clean counterpart, the map falls back to the adjacent clean position. The same map is used for both arms, so the attention and embedding interventions operate on a common correspondence. Across the evaluation set, 99.3% of positions receive a genuine clean counterpart and 0.7% are clamped.

Attention restoration. For the clean-attention arm we capture the clean run's pre-softmax $QK^\top/\sqrt{d}$ logits at every attention-bearing layer and head, and write $S^{\text{restored}}[p, q] = S^{\text{clean}}[\pi(p), \pi(q)] - \log K_{\pi(q)}$, where π is the position map and $K_{\pi(q)}$ is the number of corrupted fragments aligned to clean position $\pi(q)$. The $-\log K$ term is required for mass preservation. Without it, a clean token aligned to K fragments has its logit duplicated across K columns and its post-softmax mass inflated K-fold. With the correction, attention mass on corrupted fragments falls from 0.121 to 0.089 against a clean-run level of 0.079, and without it the same quantity rises to 0.145. The corrupted run's causal mask and positional encodings are retained, and the softmax is recomputed after injection.

Embedding restoration. For the clean-content arm, the embedding at each corrupted position is replaced by the embedding of its aligned clean token, using the same map. Because the hybrid sequence retains the corrupted sequence length, the restoration is approximate. A clean token aligned to several fragments is written into each of them, and clean tokens with no corrupted counterpart are dropped. The restoration is therefore conservative, and if anything understates the content channel.

Validation. Five checks were run before the main grid. *Identity*: with an empty corruption set the patched forward pass is bit-identical to the unpatched model (maximum logit difference 0.0). *Normalization*: post-softmax attention rows sum to one, matched against the unpatched baseline rather than an absolute tolerance, since bfloat16 imposes a floor of roughly 3×10^{-3} on row-sum error. *Causality*: the upper triangle of the attention matrix is exactly zero after softmax, confirming that the causal mask is applied after injection rather than before. This matters because the position map can send a later corrupted position to an earlier clean one. *Coverage*: the fraction of positions with a

Table 18: **Mean fragmentation across four open-weight tokenizers**, averaged across BoolQ and GSM8K ($n = 100$ per task). Keyboard fragmentation exceeds filler by 2–3× at matched word-level perturbation rates.

r	Filler	Keyboard
0.05	0.006	0.013
0.10	0.015	0.039
0.20	0.029	0.097
0.30	0.048	0.147

genuine clean counterpart is reported above. *Mass*: attention mass on corrupted fragments moves toward the clean-run level, as reported under attention restoration.

L Fragmentation Rates Across Perturbations and Languages

This appendix provides the per-perturbation fragmentation rates that support the cross-perturbation analysis in Section 5. We report mean fragmentation (the ratio of new tokens introduced by perturbation to clean-prompt token count) and token overlap (the fraction of clean-prompt token types preserved). Because the cross-perturbation controls require matched ASR/OCR variants, the English diagnostic subset used for Table 8 is separate from the 70B generalization subset reported in Appendix D. Small differences in clean accuracy therefore reflect sampling variation across $n = 100$ subsets.

English (BoolQ + GSM8K, Llama-3.3-70B tokenizer). Table 8 in the main text reports fragmentation under six perturbation–severity conditions spanning four perturbation types at $r = 0.1$–0.2, alongside the corresponding accuracy from Section 5.

Filler vs. Keyboard across four English tokenizers. To verify that filler's negligible impact on accuracy is grounded in low fragmentation, we measured fragmentation rates across four tokenizers (Llama-3.1, Mistral-7B-v0.3, Gemma-2-9B, Qwen3.5-9B) on BoolQ+GSM8K averaged. Table 18 reports mean fragmentation at four severities. Keyboard perturbation produces 2–3× higher fragmentation than filler at matched r, quantitatively explaining the asymmetric accuracy patterns in Section 4.

Chinese (CMath, Llama-3.1 tokenizer). Table 19 reports fragmentation under Chinese keyboard, typoswap, and filler perturbations at three

Table 19: **Chinese CMath fragmentation rates** (Llama-3.1 tokenizer, $n = 200$).

Perturbation	r=0.05	r=0.10	r=0.20
keyboard	0.049	0.100	0.198
typoswap	0.024	0.047	0.080
filler	0.010	0.026	0.050

severities. The ordering matches English: keyboard > typoswap > filler, with filler producing fragmentation an order of magnitude below keyboard at all severities.

M Qualitative Case Study

Figure 12 presents a per-token attention comparison for a GSM8K example that Llama-3.1-8B answers correctly under clean input but incorrectly after keyboard perturbation at r=0.1. After perturbation, tokenization fragmentation increases the token count from 48 to 63, and corrupted fragments absorb 67% of attention mass, leaving only 33% for task-relevant tokens.

N Attention Calibration and Prompt Restoration

Section 6 notes that attention calibration and prompt restoration show no consistent improvement. This appendix reports those probes.

Setup. Both probes were run on Llama-3.1-8B at r=0.1 with n=100, on a subset drawn by `shuffle(seed=42)` and using an independently implemented corruption function. The subset and prompt template therefore differ from the repair experiments in Section 6, so the baselines are not comparable across the two. Each is internally paired. Table 20 reports both probes.

Calibration. Attention calibration rescales attention mass away from tokens flagged as outliers by representation norm, leaving the input unchanged. It yields negligible or negative changes in all three conditions tested.

Restoration. Prompt restoration rewrites flagged tokens toward their nearest in-vocabulary form before inference. It degrades accuracy in all three conditions, most sharply on BoolQ, where it miscorrects valid passage tokens.

Interpretation. Neither probe recovers accuracy, and both are consistent with the coupling result of Section 5.3. Calibration adjusts attention while

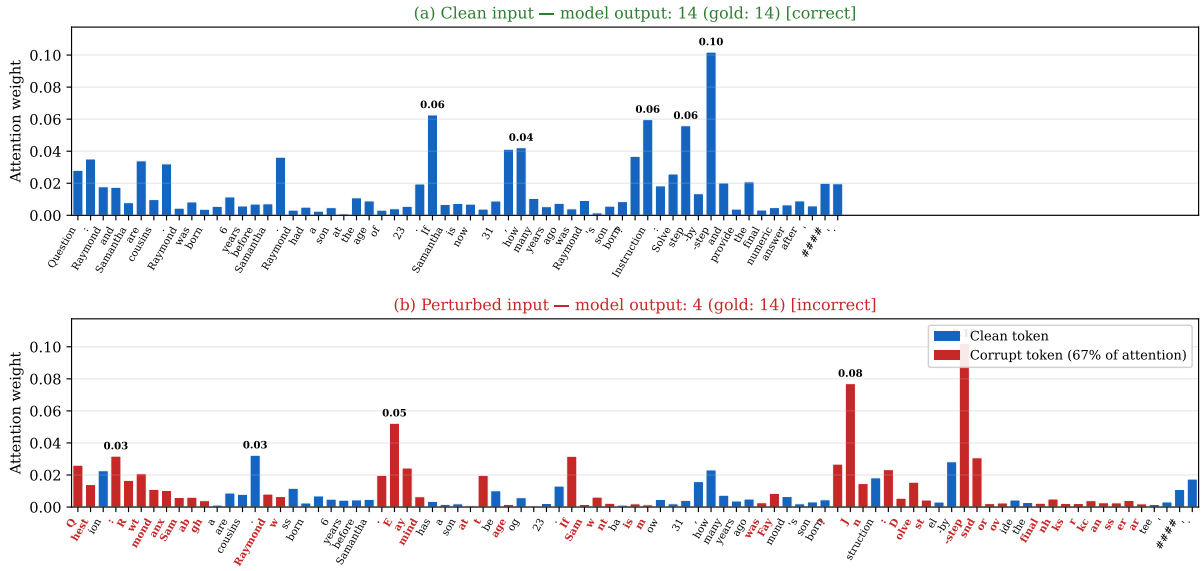

Figure 12: **Qualitative attention analysis on a GSM8K failure case** (Llama-3.1-8B, keyboard perturbation, r=0.1, layers 12–15). **(a)** Under clean input, attention concentrates on semantically relevant tokens such as names, ages, and instruction keywords, yielding the correct answer 14. **(b)** After perturbation, 67% of the attention mass shifts to corrupted subword fragments (red bars), while task-critical tokens are starved of processing bandwidth. The model outputs 4 instead of 14. Token count increases from 48 to 63 due to tokenization fragmentation.

Table 20: **Diagnostic probes on Llama-3.1-8B** (n=100, r=0.1; shuffle-42 subset with its own corruption implementation, so baselines differ from Section 6). Δ is the change under restoration relative to baseline, shaded by magnitude ($\geq$0.15 , 0.05–0.15).

Condition	Base.	Calib.	Rest.	Δ
BoolQ + KB	0.64	0.61	0.53	−0.11
BoolQ + TS	0.83	0.82	0.65	−0.18
HellaSwag + KB	0.22	0.18	0.19	−0.03

leaving the corrupted content in place, which the factorial intervention shows to be ineffective at best. Restoration modifies content without knowing which tokens were corrupted, and its errors propagate in the same way as the confident-but-wrong substitutions analysed in Section 6.1.

O Tokenizer-Level Defense Protocol

Section 6.1 reports a detection-and-repair defense operating without access to the clean text. This appendix gives the protocol.

Constraint. The detector and the repair observe only the corrupted input. The perturbation mask is used solely to score detector precision and recall after the fact, in a code path never called during detection or repair. All thresholds and the deployed detector were selected on a held-out split disjoint from the evaluation subset.

Detection. Three signals are combined. Rarity flags words that are out of vocabulary or fall below a unigram-frequency threshold. Surprisal flags tokens whose local negative log-likelihood under a small reference model exceeds a threshold. Fragmentation flags words whose subword-pieces-per-character ratio is unusually high under the target tokenizer. We evaluate each signal alone, their union, and a majority vote, and additionally a high-precision variant requiring both the majority vote and out-of-vocabulary status.

Repair. Vocabulary-constrained correction replaces each flagged word with the highest-frequency in-vocabulary word within edit distance two. A conservative variant restricts this to edit distance one and declines to act when no candidate with non-zero frequency exists. Character-level fallback, which re-encodes flagged spans as per-character tokens, and greedy longest-match re-tokenization were implemented and dry-run but not deployed: both rewrite every flagged word and therefore inherit the detector's false-repair rate of 22–25%.

Evaluation. Four open-weight models on GSM8K and BoolQ, typoswap at r=0.1, n=100 per model–task cell. Recovery is reported as a percentage of the clean–corrupted gap with paired bootstrap confidence intervals. Every variant is

additionally run on clean inputs, so that a defense which improves corrupted accuracy at the cost of clean accuracy can be identified. No variant shows significant clean-input degradation.

Results. Detection reaches F1 0.77–0.81 with the majority-vote detector, and fragmentation alone reaches 0.70–0.75 without any clean text. Vocabulary-constrained correction restores the original word in 67.0% of truly corrupted words at edit distance two and 69.9% under the conservative variant. Recovery of the accuracy gap is significantly positive in none of the eight model–task cells and significantly negative in one.

P Extended Related Work

Section 7 organizes prior work by where each line locates the remedy for lexical corruption: at the input, in the attention pattern, or during training. This appendix gives the full citations and expands each comparison with the evidence behind it.

Robustness to textual perturbations. Character-level noise and spelling errors have long been known to degrade neural NLP systems, particularly in machine translation and sequence labeling (Belinkov and Bisk, 2018; Pruthi et al., 2019). Adversarial attack methods such as TextBugger (Li et al., 2019) and HotFlip (Ebrahimi et al., 2018) demonstrate that small surface-form modifications can substantially alter predictions even when semantic meaning is preserved. Related work on reading comprehension shows that neural models often rely on brittle lexical cues (Jia and Liang, 2017). These studies primarily target earlier architectures (LSTMs, BERT-scale transformers) and search for the edit that maximally changes an output, whereas our perturbations are undirected and sampled uniformly over words, so the degradation we report is the expected cost of ordinary typing and transcription noise rather than a worst case. They also locate the remedy at the input, correcting the surface form before the model reads it. Our cross-perturbation controls delimit when that succeeds. ASR homophones produce no fragmentation and leave accuracy at baseline, and OCR confusions fragment more heavily than any other perturbation yet still leave accuracy near baseline, because their pieces follow familiar subword patterns. Only typoswap, whose fragments are out of distribution, damages GSM8K (Table 8). Correction therefore works exactly where the original form is still recoverable from vocabulary and context, and fails on the corruptions that dominate mathematical reasoning, where a perturbed numeral remains a valid, in-vocabulary numeral admitting no dictionary neighbour. The tokenizer-level defense makes the bound concrete: detection reaches F1 0.77–0.81 without access to clean text, yet vocabulary-constrained repair restores the original word only 67–70% of the time and recovers no accuracy in any of eight model–task cells (Section 6.1). A separate line intervenes during training instead. Subword regularization, in particular BPE-Dropout (Provilkov et al., 2020), randomizes segmentation so that a model never comes to depend on a single canonical tokenization. Our mechanism supplies the reason this is the effective point of intervention: once fragmentation has occurred, the information needed to undo it is no longer present in the input, so any later stage is working from a representation that has already lost what it would need.

LLM reasoning and evaluation. Reasoning benchmarks such as BoolQ (Clark et al., 2019a), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), and GSM8K (Cobbe et al., 2021) are widely used to evaluate LLM capabilities. Chain-of-thought prompting (Wei et al., 2022) has been shown to improve multi-step reasoning, and subsequent work explores self-consistency (Wang et al., 2023) and other inference-time strategies. Recent studies also examine LLM sensitivity to prompt formatting (Sclar et al., 2024; Lu et al., 2022), showing that minor template variations can affect performance. In all of these settings the prompt varies while the tokenization of its content stays intact, so the failure is one of framing and better prompting can address it. Lexical corruption changes the token identities themselves, which is why stronger reasoning does not help: chain-of-thought lifts clean GSM8K on GPT-4o from 0.42 to 0.90, confirming the reasoning capability is intact, yet typoswap still pulls it to 0.70, and GPT-5.4's built-in reasoning mode shows the largest keyboard degradation of any frontier configuration we evaluate, amplifying the damage rather than absorbing it (Appendix A).

Attention analysis and mechanistic interpretability. Attention patterns are widely used to analyze transformer behavior (Vaswani et al., 2017; Clark et al., 2019b). Recent mechanistic interpretability work examines how specific attention heads contribute to reasoning (Geva et al.,

2021; Elhage et al., 2021). Studies on attention sinks (Xiao et al., 2024) show certain tokens attract disproportionate attention regardless of relevance. Attention Diversion describes a related but distinct phenomenon: the absorbing positions are not fixed properties of the model but are created by the input itself, as corrupted fragments become transient sinks that draw attention mass away from task-relevant tokens. The difference matters for what can be done about it. Static sinks can be suppressed or offset, whereas input-induced sinks cannot be ablated without cost, since the fragments that absorb attention are also what the model uses to reconstruct the intended word. Two results establish this. Rescaling attention away from outlier tokens degrades accuracy in every condition tested, by up to 0.31 on GSM8K (Appendix J), so the diverted attention is load-bearing rather than passive noise. The factorial intervention then shows it is not independently manipulable either: supplying a clean attention pattern over corrupted content is significantly harmful in three of four models, restoring content alone reaches significance in only one, and only restoring both together recovers 45–71% of the gap, with a significantly positive interaction in every model (Section 5.3). Diversion is therefore neither a passive symptom to be corrected nor a free-standing cause to be suppressed, but is functionally bound to the corrupted content it addresses.