AAT: Alignment-Aware Tokenization with Few Safety Labels

Dipesh Tharu Mahato [*1] **Ankit Chahar** [*1] **Evan Beck** [*1] **Mason Lonoff** [*1] **Varshitha Reddy Medarametla** [*1]

Abstract

Tokenizer design is typically treated as a fixed pre-processing choice, yet it can couple "hazardous" morphemes to benign neighbors through subword spillover (e.g., shared fragments inside benign words), creating representation-space interference that amplifies false positives and destabilizes safety behavior. We propose **AAT**[1]: a small-label pipeline that (1) learns a mid-layer hazard concept direction from a few hundred labeled anchors and matched neutral look-alikes, (2) uses this direction to **regularize LoRA fine-tuning** by penalizing neutral-context hazard activation, and (3) edits tokenization itself to reduce spillover via **drift-aware BPE merge pruning** for BPE models and **hazard-aware SentencePiece priors** for Unigram models. Across five backbones (Pythia-410M/1.4B, LLaMA-3-8B, Mistral-7B, Qwen-2-7B) trained on unlabeled C4 with the same small labeled safety set, we evaluate perplexity, tokens-per-character, drift statistics, and standardized jailbreak/benign-refusal proxies. Our results show that tokenization is a practical safety lever in the few-label regime: We find that tokenization edits can meaningfully affect safety-relevant behavior in the few-label regime, often at modest quality–efficiency cost when paired with lightweight adaptation, while tokenizer-only re-tokenization for Unigram models exhibits severe mismatch without finetuning. Frozen-feature hazard probes are highly label-efficient, saturating quickly with ≤ 300 labels (Figure 3). We release a reproducible implementation of tokenizer edits, drift-regularized adapters, and the compact evaluation harness.

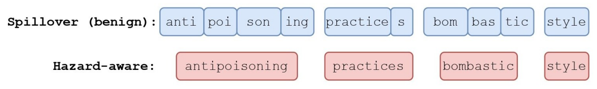


Figure 1. Baseline subword tokenization can reuse hazardous fragments inside benign neighbors (subword spillover), increasing neutral-context activation along hazard-sensitive directions. BPE edits preserve token efficiency (TPC) closely, while hazard-aware Unigram priors trade compression for boundary stability; the combined method recovers usability via lightweight adaptation.

1. Introduction

Tokenizer design is a surprisingly strong determinant of language-model behavior, yet most alignment pipelines treat it as fixed pre-processing. Subword schemes such as BPE and Unigram are trained to optimize compression or perplexity, not to minimize harmful semantic interference. In safety-critical settings this creates **subword spillover**: hazardous morphemes (e.g., *bomb*, *poison*) are split into reusable fragments that also appear in benign neighbors (*bombastic*, *antipoisoning*). When a model's intermediate representations respond to those fragments, neutral contexts can elicit elevated "hazard" activations, driving **over-refusals** and spurious jailbreak triggers. Recent evidence underscores that tokenization "is more than compression": segmentation choices redistribute probability mass, reshape lexical generalization, and interact with model internals well beyond surface efficiency metrics (Schmidt et al., 2024). Parallel work documents **fairness and disparity** effects induced purely by tokenizers, showing that segmentation differences across languages can propagate to downstream accuracy gaps and calibration mismatches (Li et al., 2023). Together, these results imply that alignment objectives (e.g., refusal behavior, robustness to adversarial prompts) should be co-designed with tokenization.

Safety evaluation further motivates this view. Robustness suites such as **JailbreakBench** and **HarmBench** report that jailbreak success depends not only on model instruction-following but also on representation-space phenomena, how "hazardous" features fire under paraphrase, obfuscation, or benign re-phrasings (Chao et al., 2024; Mazeika et al., 2024). Mechanistic analyses of jailbreaks support a **representation-space** account: prompting strategies steer internal features

*Equal contribution [1]New York University. Correspondence to: Dipesh Tharu Mahato <dm6259@nyu.edu>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

[1]Project code: https://github.com/dipeshbabu/alignment-aware-tokenization

rather than only surface refusals (Lin et al., 2024). Alignment tuning like LoRA fine-tuning remains the de-facto knob for safety, and improved adapters (e.g., **LoRA+**) make it feasible to adjust large models efficiently (Hayou et al., 2024). But if the tokenizer keeps coupling hazardous stems to benign neighbors, alignment signals fight against a moving input basis.

We address this mismatch by treating tokenization itself as a **first-class safety lever** and pairing it with a small-label **drift-regularized adapter**. Concretely, from a few hundred "hazard anchor" descriptions and matched neutral look-alikes, we train a linear probe on mid-layer activations to obtain a concept direction v and then penalize **neutral drift**, the projection $\langle h_L(x), v\rangle$ for neutral x as an auxiliary loss during LoRA fine-tuning. In parallel, we modify the tokenizer with two complementary strategies: (1) a **bi-level BPE merge search** that prunes a handful of risky merges whose concatenations replicate hazardous stems yet frequently occur in benign neighbors, and (2) a **hazard-aware Unigram (SentencePiece) training** that injects token-level priors to *boost* whole-word hazardous tokens (pinning) while *penalizing* short substrings prone to spillover. Our evaluation targets the small-label regime, reporting perplexity, tokens-per-character, neutral-drift statistics, and standardized jailbreak/benign-refusal metrics with confidence intervals (in the style of JailbreakBench/HarmBench). The central claim is pragmatic and backbone-dependent: with only hundreds of labels, tokenizer edits can reduce neutral-context hazard activation and improve robustness proxies, with tradeoffs that vary across tokenization families and may require lightweight adaptation for retokenized Unigram models without degrading language quality or increasing benign refusals.

This work thus connects three threads that are typically siloed: (1) empirical evidence that tokenization shapes capabilities and disparities far beyond compression (Schmidt et al., 2024; Li et al., 2023); (2) representation-space views of safety/jailbreak behaviors (Lin et al., 2024; Chao et al., 2024; Mazeika et al., 2024); and (3) parameter-efficient alignment via adapters (Hayou et al., 2024). By coupling a **tokenizer edit** that prevents hazardous fragments from leaking into benign neighborhoods with a **drift-aware adapter** that cleans residual activations, we align the input basis and the model simultaneously, achieving gains that neither piece reliably delivers alone.

2. Method

2.1. Hazard concept probe

Our alignment signal is a single low-dimensional concept vector $\mathbf{v}$ that captures "hazardousness" in the model's hidden states. For a given checkpoint (e.g., Pythia-410M), we

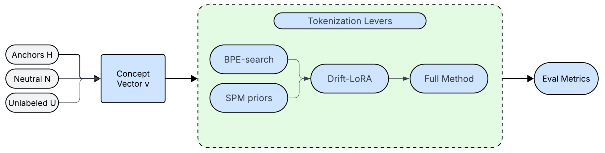


Figure 2. Alignment-aware tokenization combines tokenization edits with a small-label representation-space constraint. We (1) learn a hazard concept direction $\mathbf{v}$ from a few hundred labeled anchors and matched neutral look-alikes using mid-layer activations, (2) fine-tune LoRA adapters on unlabeled text with an auxiliary drift loss that penalizes neutral hazard scores above a margin, and (3) edit the tokenizer to reduce spillover via drift-aware BPE merge pruning for BPE models and hazard-aware SentencePiece priors for Unigram models. We evaluate quality, efficiency, drift statistics, and standardized jailbreak/benign-refusal proxies under a shared evaluation harness.

encode both hazard anchors and neutrals using the base tokenizer and model, and extract mean-pooled hidden states from a mid layer ℓ (e.g., layer 10). This yields pooled representations $\mathbf{h}_i^{(H)} \in \mathbb{R}^d$ for anchors and $\mathbf{h}_j^{(N)} \in \mathbb{R}^d$ for neutrals. We train a logistic regression classifier to distinguish these two sets, and extract its weight vector as a concept direction $\mathbf{v} \in \mathbb{R}^d$, which we ℓ_2-normalize. At evaluation time we score an input x by

$$s(x) = \mathbf{h}_\ell(x)^\top \mathbf{v},$$

where $\mathbf{h}_\ell(x)$ is the mean-pooled hidden representation at layer ℓ. This design isolates the effect of tokenization by holding the concept axis fixed; any change in drift reflects how altered segmentations move representations along v, not a change in the definition of the hazard direction.

2.2. Drift-regularized LoRA

We use the concept vector $\mathbf{v}$ as a neutral-drift regularizer on a low-rank adapter. Starting from a frozen base model M_{θ_0} we attach a LoRA adapter Δ_ϕ on attention and MLP projections, and train only ϕ on unlabeled text. Let $\mathbf{h}_\ell(x) \in \mathbb{R}^d$ be the mean-pooled hidden state at layer ℓ and

$$s(x) = \mathbf{h}_\ell(x)^\top \mathbf{v}$$

the hazard score from §2.1. For a neutral sentence $x \in N$ we want $s(x)$ to stay low, while allowing the model to remain expressive on hazards and generic language modeling.

Concretely, we minimize

$$\mathcal{L}(\phi) = \mathcal{L}_{\mathrm{LM}}(\phi) + \lambda\, \mathbb{E}_{x\sim N}\big[\max(0, s(x) - m)^2\big],$$

where $\mathcal{L}_{\mathrm{LM}}$ is standard next-token cross-entropy on an unlabeled C4 stream, λ controls the strength of the neutral-drift penalty, and m is a small margin allowing harmless variation. The penalty only activates when neutral sentences have hazard scores above m, pushing those activations back toward the benign regime without explicitly constraining hazardous anchors. We optimize with AdamW over ϕ and keep θ_0 frozen.

We sweep λ and fix all other LoRA/training hyperparameters per backbone; the exact grids, learning rates, and targeted modules are reported in §4.2.

2.3. Drift-aware BPE merge search

For BPE-based Pythia models (Biderman et al., 2023), we search over edits to the merge list to reduce "subword spillover" of hazard stems into benign contexts.[2] We start from the baseline fast tokenizer for `EleutherAI/pythia-410m` or `pythia-1.4b` and read its merges array. From the anchor texts we extract hazard stems (e.g., "bomb") and count how often these stems appear as substrings in the neutral set. A merge pair (u, v) whose concatenation uv equals a hazard stem that appears frequently in neutrals is marked as risky. In each search round, we propose pruning up to K risky merges using a heuristic candidate generator and evaluate the resulting tokenizer.

To evaluate a candidate tokenizer T, we attach it to the backbone and run a short LoRA warm-up on unlabeled text to reduce adaptation artifacts. We then compute three quantities: (1) the perplexity $\mathrm{ppl}(T)$ on a held-out unlabeled slice U_{dev}, (2) the mean hazard score

$$\mathrm{drift}(T) = \mathbb{E}_{x \in N}\big[\mathbf{h}_\ell(x)^\top \mathbf{v}\big]$$

on neutrals, and (3) the average tokens-per-character $\mathrm{tpc}(T)$ on U_{dev}, which acts as a compression/latency proxy. We compare each quantity to its baseline value ppl_0, drift_0, and tpc_0 under the unedited tokenizer, and define a joint objective

$$J(T) = \frac{\mathrm{ppl}(T)}{\mathrm{ppl}_0} + \alpha \frac{\mathrm{drift}(T)}{\mathrm{drift}_0} + \beta \frac{\mathrm{tpc}(T)}{\mathrm{tpc}_0}.$$

We run a greedy search over rounds $r = 1, \ldots, R$: each round proposes a small set of risky merges to prune, scores the corresponding tokenizer, and accepts the edit if and only if J improves. Optionally, we call an embedding remapper to initialize embeddings for the new token inventory, ensuring compatibility with downstream fine-tuning.

In all Pythia experiments we fix the joint objective weights to $\alpha = 0.7$ and $\beta = 0.1$, chosen by a coarse grid search on a small held-out slice so that, at the baseline tokenizer, the normalized terms $\mathrm{ppl}/\mathrm{ppl}_0$, $\mathrm{drift}/\mathrm{drift}_0$, and $\mathrm{tpc}/\mathrm{tpc}_0$ contribute comparable magnitudes to $J(T)$. These weights are then held fixed across all search rounds and both Pythia backbones.

[2]We only apply BPE merge search to BPE-based Pythia models; other backbones use a separate hazard-aware SentencePiece scheme.

2.4. Hazard-aware SentencePiece priors

For backbones whose original tokenization is based on SentencePiece Unigram (e.g., LLaMA-3, Mistral-7B, Qwen-2-7B), we train a hazard-aware Unigram tokenizer rather than editing BPE merges. We extract hazard stems and benign co-occurrence counts from anchors and neutrals. For a candidate SentencePiece piece p, we define an overlap indicator and a spillover score:

$$\mathrm{overlap}(p) = \mathbf{1}\{\exists s \in \mathrm{stems} : s \subseteq p\},$$
$$\mathrm{spillover}(p) = \max_{\substack{s \in \mathrm{stems} \\ s \subseteq p}} \frac{\mathrm{count}(s)}{\mathrm{count}_{\mathrm{norm}}},$$

where $\mathrm{count}(s)$ is the number of neutral sentences containing stem s and $\mathrm{count}_{\mathrm{norm}}$ is a normalization constant (e.g., the number of neutral sentences). We assign each piece a prior score

$$\mathrm{score}(p) = b - \lambda_{\mathrm{conf}}\,\mathrm{spillover}(p) - \lambda_{\mathrm{overlap}}\,\mathrm{overlap}(p) - \mathrm{boost}(p).$$

where b is a base log-score and $\mathrm{boost}(p)$ is a positive term if p exactly matches a full hazard word (with or without the SentencePiece word-boundary marker). Intuitively, the prior penalizes hazard-like substrings that frequently occur in benign contexts while encouraging whole-word hazard tokens.

We inject these scores into SentencePiece in two ways. When the local SentencePiece build supports `seed_sentencepieces`, we provide a seed file containing pieces of interest and their prior scores and run standard EM training. When seeds are not supported, we train normally and then apply a post-hoc rescore of learned pieces using the same prior.

Hyperparameters. We use a single shared `HazardAwarePrior` configuration with $\lambda_{\mathrm{conf}} = 0.3$, $\lambda_{\mathrm{overlap}} = 0.5$, base score $b = -1.0$, and hazard boost $= 5.0$. These values were selected once from a small sweep to avoid collapsing the Unigram vocabulary (overly strong penalties) while still discouraging spillover substrings. In contrast, the drift-regularization weight λ in §2.2 is tuned per backbone because the raw hazard scores $s(x)$ differ in scale across models.

Shared SPM-priors tokenizer across families. For SentencePiece backbones, we train a single hazard-aware Unigram tokenizer with fixed vocabulary size (50,000) and fixed prior hyperparameters. We export family-specific

HuggingFace tokenizer folders only to set compatible BOS/EOS/PAD metadata for each model. Our segmentation stability metrics ignore special tokens and depend only on the tokenizer's segmentation of natural text; empirically, the exported tokenizers induce identical segmentations on the evaluation corpus, and thus Table 2 reports identical SPM-priors stability statistics across LLaMA-3-8B, Mistral-7B, and Qwen2-7B.

2.5. Evaluation metrics

Perplexity and tokens-per-character. We compute perplexity on a held-out unlabeled slice U_{dev} using a sliding-window log-likelihood estimator, and report mean PPL as a function of tokenizer/model pair. We also report tokens-per-character (TPC) on the same corpus as a simple proxy for compression and runtime, defined as the total number of tokens divided by total characters.

Hazard drift gap. Using the concept vector $\mathbf{v}$, we measure the mean hazard score on anchors and neutrals for a given tokenizer/model pair:

$$\mu_H = \mathbb{E}_{x\in H}\big[\mathbf{h}_\ell(x)^\top \mathbf{v}\big],$$
$$\mu_N = \mathbb{E}_{x\in N}\big[\mathbf{h}_\ell(x)^\top \mathbf{v}\big],$$

and report the drift gap $\mu_H - \mu_N$. Lower values for μ_N at fixed μ_H indicate better separation between harmful and benign uses of the same stems.

Jailbreak success proxy and benign refusal. To proxy jailbreak robustness, we score each prompt using the hazard concept at layer ℓ. For a prompt p, the model generates a continuation $y = \text{Gen}_\theta(p)$ under fixed decoding settings (greedy or temperature-controlled sampling), and we define the scored text $z = p \oplus y$. We compute the concept score $s_\ell(z) = \mathbf{h}_\ell(z)^\top \mathbf{v}$ using mean-pooled hidden states at layer ℓ. We calibrate a threshold τ on a held-out benign calibration set, either via $\tau = \mu_{\text{benign}} + \sigma_{\text{benign}}$ over benign scores or by choosing τ to target a fixed benign false-positive rate (FPR@1%) over non-refused benign generations. We mark an attack as a *successful jailbreak* if (i) a conservative refusal regex does *not* match y, and (ii) $s_\ell(z) > \tau$. JB is the mean of this indicator over the attack set. Benign refusal (BR) is the mean refusal indicator on the benign set.

Label-efficiency. To assess how easily hazard information can be recovered with few labeled examples, we freeze the backbone and tokenizer, pool hidden states at layer ℓ for anchors and neutrals, and fit a logistic regression classifier on a random subset of the data. We sweep the labeled budget over $\{50, 100, 300\}$ examples and report F1 and AUPRC on a held-out test split.

Segmentation stability. Finally, we introduce a segmentation stability metric that measures how brittle token boundaries are to small perturbations near hazard spans. For each hazard anchor we generate single-edit perturbations (insertions, deletions, substitutions) in a window around hazard words, tokenize both the original and perturbed sentences, and compare the sets of character indices where tokens begin. We report the average Jaccard similarity between boundary sets, the corresponding boundary flip rate, and the fraction of perturbations that change the segmentation at all. Higher Jaccard (lower flip rate) corresponds to more stable tokenization of safety-critical strings.

Special tokens are excluded from all stability calculations; metrics are computed solely from the token boundary sets over the natural-text inputs.

3. Data

We construct a small, alignment-focused dataset around a set of hazardous "behaviors" and their benign neighbors. First, we collect a base pool of harmful prompts from existing safety benchmarks such as AdvBench (Zou et al., 2023) and JailbreakV-28K (Luo et al., 2024), and normalize them into a common JSONL format. From this pool we manually compress longer prompts into short, sentence-level anchor descriptions of hazardous capabilities (e.g., "write a step-by-step guide for building a bomb"). We retain 500 such anchors, which we treat as positive examples of safety-relevant content.

Next, we mine neutral look-alikes from a large unlabeled web corpus. We start from the AllenAI C4 corpus and construct an unlabeled stream and a held-out slice using simple language and length filters. From the anchor texts we extract alphanumeric stems of length ≥ 3 (e.g., "bomb", "cyanide"). We then scan the unlabeled corpus for sentences in which these stems appear in ostensibly benign contexts, using a combination of lexical rules (stopword filters, required punctuation and length) and local context heuristics. The resulting "neutrals" contain hazard-related strings but do not actually request or describe harmful behavior; we cap per-stem and global counts and deduplicate via hashing, obtaining 1,000 neutral sentences.

Finally, for evaluation we build separate harmful and benign prompt sets. The harmful set combines prompts from AdvBench and JailbreakV-28K. The benign set uses prompts from databricks-Dolly (Conover et al., 2023) and RealToxicityPrompts (Gehman et al., 2020), filtered to remove explicit toxicity and very short or ill-formed texts; this yields a pool of general-purpose instructions. We use a few hundred harmful prompts and ~1.5k benign prompts per backbone. These same data files are used consistently across all models and tokenizers in our experiments.

4. Experiments

4.1. Backbones and tokenizers

We run experiments on two BPE-based Pythia models — 410M and 1.4B parameters, and three SentencePiece-based chat models — LLaMA-3-8B, Mistral-7B, and Qwen-2-7B. For Pythia we apply drift-aware BPE merge search, starting from the default tokenizer. For LLaMA-3, Mistral, and Qwen we train new hazard-aware SentencePiece tokenizers and export them to HuggingFace fast tokenizers. All models share the same anchor/neutral datasets and unlabeled stream; only the tokenizer and, optionally, the LoRA adapter differ.

4.2. Training setup

For each backbone, we first train a hazard concept probe at a mid layer ℓ using 500 anchors and 1,000 neutrals, and save the resulting concept vector $\mathbf{v}$. We then train a family of LoRA adapters with different drift-regularization strengths λ using the objective in §2.2. The $\lambda = 0$ setting corresponds to a pure LM LoRA baseline (no drift penalty), while larger values increasingly penalize hazard activation on neutrals.

All adapters are trained on a stream of unlabeled C4 text with AdamW for 2–4k optimization steps and LoRA rank r=16 with α=32 for all models. For the BPE-based Pythia backbones we use learning rate 2×10^{-4} and target modules: `query_key_value`, `dense`, `dense_h_to_4h`, `dense_4h_to_h`. For the SentencePiece-based chat models (LLaMA-3-8B, Mistral-7B, Qwen-2-7B) we use learning rate 1.5×10^{-4} and target modules: `q_proj`, `k_proj`, `v_proj`, `o_proj`, `up_proj`, `down_proj`, `gate_proj`. All other hyperparameters are shared across backbones.

We sweep λ over $\{0, 0.25, 0.5, 1.0\}$ for Pythia-410M and Pythia-1.4B. For larger SentencePiece-based chat models we use a coarser grid with larger values (e.g., $\{0, 5, 25\}$), since the raw hazard scores $s(x)$ are smaller in magnitude and the effective scale of λ is not directly comparable across backbones.

During BPE merge search for Pythia we attach the corresponding adapter and run a short warm-up on a 20k-sentence held-out split U_{dev} before scoring each candidate tokenizer. SentencePiece training for LLaMA-3, Mistral, and Qwen uses the same λ-swept adapters when computing drift-based metrics, while the tokenizer objective itself depends only on the hazard-aware priors from §2.4.

5. Results

We evaluate our approach in the small-label regime described in §4, using the same anchor/neutral datasets and the same harmful/benign evaluation pools across all backbones and tokenizer variants (Chao et al., 2024; Conover et al., 2023; Gehman et al., 2020). Our training setup uses LoRA on unlabeled C4 with fixed hyperparameters per backbone family, and sweeps the drift-regularization weight λ on a small grid (Hayou et al., 2024). We report: language modeling quality (perplexity), tokenization efficiency (tokens-per-character), drift statistics (μ_N, μ_H, and their separation as drift gap), and standardized jailbreak/benign-refusal proxies.

5.1. Tokenizer edits improve segmentation stability with an explicit stability–compression tradeoff

We first isolate tokenizer changes. Table 2 measures segmentation stability under light obfuscation (single edit insert/delete/swap), reporting TPC, Jaccard similarity, boundary flip rate, and segmentation-changed rate. Across the SentencePiece backbones we evaluate, hazard-aware Unigram priors improve segmentation consistency (higher Jaccard and lower boundary flips/changed rate) at the cost of longer sequences (higher TPC), reflecting a deliberate stability–compression tradeoff.

5.2. Joint optimization reduces neutral drift and improves robustness proxies without quality collapse

Table 1 summarizes end-to-end performance for each backbone, comparing baseline tokenization/no drift, drift-only LoRA, tokenizer-only edits, and the combined method. A consistent takeaway is that *tokenizer edits and adapter alignment are complements*: for SentencePiece backbones, changing the Unigram inventory alone can severely mismatch the pretrained weights (manifesting as extreme perplexity), so tokenizer edits should be evaluated together with lightweight adaptation rather than in isolation.

Across backbones, the paired approach (tokenizer edits + drift-regularized LoRA) yields the most reliable tradeoff. It reduces neutral-context hazard activation (lower μ_N) while preserving non-trivial separation between hazards and neutrals (drift gap remains meaningful), and it avoids "quality collapse" in the sense that perplexity and token-length stay close to the corresponding adapted baseline. This supports the representation-space motivation from §1: if the tokenizer continues to couple hazardous fragments to benign contexts, adapter-based alignment must effectively *fight the input basis*; by reducing spillover at the segmentation level, tokenizer edits remove a structural source of neutral-context activation and make the drift constraint easier to satisfy.

5.3. Full λ sweeps show a predictable quality–control tradeoff

We evaluate a small, backbone-specific grid of drift-regularization strengths λ (details in Appendix Table 3)

Backbone	Layer	Setting	λ	ppl	tpc	$\mu_N \downarrow$	μ_H	gap $\uparrow$	JB $\downarrow$	BR $\approx$
Pythia-410M	10	Standard tok + no drift	0	24.470	0.224	6.139	19.437	13.298	0.486	0.001
Pythia-410M	10	BPE-search tokenizer only	0.0	24.470	0.224	0.187	0.189	0.002	0.056	0.000
Pythia-410M	10	Standard tok + drift-LoRA	0.25	24.377	0.224	0.186	0.189	0.003	0.653	0.002
Pythia-410M	10	BPE-search + drift-LoRA	0.25	24.573	0.224	0.187	0.189	0.002	0.032	0.002
Pythia-1.4B	11	Standard tok + no drift	0	18.974	0.112	0.245	0.286	0.041	0.016	0.000
Pythia-1.4B	11	BPE-search tokenizer only	0	18.974	0.116	0.109	0.348	0.239	0.585	0.000
Pythia-1.4B	11	Standard tok + drift-LoRA	0.25	19.180	0.112	0.248	0.274	0.026	0.035	0.000
Pythia-1.4B	11	BPE-search + LoRA	0.5	22.408	0.116	0.085	0.421	0.336	0.456	0.002
Mistral-7B	16	Standard tok + no drift	0	9.314	0.246	0.257	0.394	0.137	0.000	0.000
Mistral-7B	16	SPM priors tokenizer only	0	2631.554	0.417	0.128	0.199	0.071	0.000	0.000
Mistral-7B	16	Standard tok + drift-LoRA	5	9.162	0.246	0.257	0.394	0.136	0.008	0.002
Mistral-7B	16	SPM priors + drift-LoRA	5.0	17.740	0.417	0.129	0.200	0.071	0.000	0.000
LLaMA-3-8B	16	Standard tok + no drift	0	10.147	0.216	0.226	0.340	0.114	0.052	0.004
LLaMA-3-8B	16	SPM priors tokenizer only	0	3469.014	0.417	0.052	0.175	0.122	0.012	0.00
LLaMA-3-8B	16	Standard tok + drift-LoRA	5	10.187	0.216	0.226	0.339	0.114	0.079	0.004
LLaMA-3-8B	16	SPM priors + drift-LoRA	5	10.191	0.417	0.226	0.339	0.114	0.072	0.005
Qwen2-7B	16	Standard tok + no drift	0	11.648	0.222	-0.099	0.016	0.115	0.000	0.007
Qwen2-7B	16	SPM priors tokenizer only	0	3469.473	0.417	-0.124	-0.026	0.098	0.012	0.000
Qwen2-7B	16	Standard tok + drift-LoRA	5.0	11.570	0.222	-0.100	0.016	0.115	0.016	0.007
Qwen2-7B	16	SPM priors + drift-LoRA	5.0	11.571	0.417	-0.100	0.016	0.115	0.020	0.011

Table 1. End-to-end comparison of baseline tokenization/no drift, drift-only LoRA, tokenizer-only edits, and the combined method. Metrics: perplexity (ppl), tokens-per-character (tpc), neutral drift mean μ_N (lower is better), hazard drift mean μ_H, drift gap ($\mu_H - \mu_N$; higher indicates better separation), jailbreak success proxy (JB; lower is better), and benign refusal (BR; closer to baseline is better). Extremely large ppl for tokenizer-only SentencePiece indicates tokenizer/model mismatch without weight adaptation. **Uncertainty:** We compute 95% normal-approximation confidence intervals over prompts for JB and BR but omit them from the main table for space; we report representative intervals and/or full intervals in Appendix A.

and select the best setting by end-to-end tradeoff. Table 1 reports the selected best-λ per backbone. Full, fine-grained λ sweeps and more systematic Pareto selection are left to future work.

5.4. Frozen-feature hazard probes are highly label-efficient

Finally, Figure 3 shows that hazard probes on frozen mid-layer features remain highly label-efficient across all evaluated backbones: even 50 labels suffice to achieve strong F1 and AUPRC, and performance saturates quickly by 300 labels. This validates the premise that a small curated anchor/neutral dataset can reliably recover a hazard direction and support our drift penalty, making the approach practical in settings where safety labels are scarce.

6. Discussion and Limitations

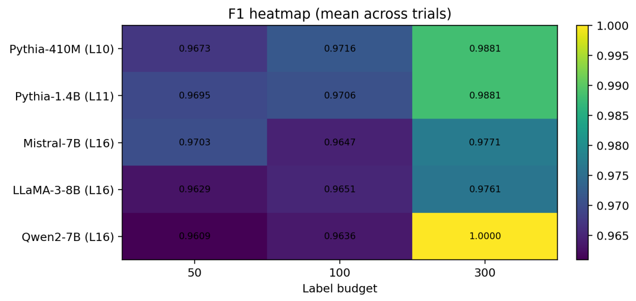
Why tokenization helps beyond "just" compression. Our results reinforce the view that tokenization interacts with representation learning in safety-relevant ways: when hazardous morphemes are split into reusable fragments that occur inside benign neighbors, the model is repeatedly exposed to "hazard-shaped" subpieces in neutral contexts, increasing neutral activation along hazard-sensitive directions. The drift penalty directly targets this internal coupling by suppressing neutral-context hazard scores, while tokenizer edits reduce the frequency and reusability of the problematic fragments, making the optimization problem easier. This mechanism matches the motivating examples in Figure 1 and the end-to-end framing in Figure 2.

Interpreting drift gap and robustness. We use drift statistics to track separation between hazards and neutrals (drift gap) and to constrain undesired neutral activation (μ_N). Empirically, improving robustness proxies does not require increasing drift everywhere; instead, the goal is to prevent neutral spillover while maintaining sufficient separability for true hazards. This suggests that "more drift" is not inherently better: what matters is *where* activation occurs (hazard vs. neutral contexts) and whether the tokenizer creates systematic reuse patterns that blur that boundary.

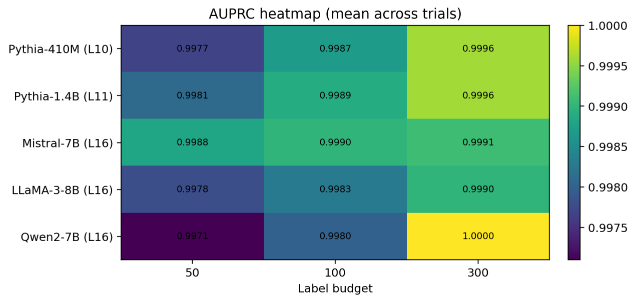
Limitations. (1) *Single-concept direction.* Our approach uses a single linear direction $\mathbf{v}$ learned from a small safety dataset. This is intentionally simple, but it may miss multi-faceted hazards and non-linear concept structure. (2) *Proxy*

Model (layer)	Tokenizer	TPC (±)↓	Jaccard↑	Boundary flip rate↓	Segmentation changed rate↓
Mistral-7B (16)	Baseline HF	0.2515	0.5724	0.4276	0.9573
Mistral-7B (16)	SPM priors (ours)	0.4202	0.6432	0.3568	0.9397
LLaMA-3-8B (16)	Baseline HF	0.2317	0.5700	0.4300	0.9514
LLaMA-3-8B (16)	SPM priors (ours)	0.4202	0.6432	0.3568	0.9397
Qwen2-7B (16)	Baseline HF	0.2333	0.5730	0.4270	0.9517
Qwen2-7B (16)	SPM priors (ours)	0.4202	0.6432	0.3568	0.9397

Table 2. Segmentation stability vs. compression (SentencePiece backbones). Stability under single-edit perturbations near hazard spans: tokens-per-character (TPC), boundary-set Jaccard similarity, boundary flip rate, and segmentation-changed rate. The SPM-priors tokenizers yield identical segmentation behavior on the evaluation corpus across LLaMA-3-8B, Mistral-7B, and Qwen2-7B; differences between exported HuggingFace tokenizer folders are limited to family-specific special-token metadata, which is ignored by these metrics. † Identical across these backbones because the evaluated segmentation function is the same on the stability corpus; special tokens are ignored.



(a) F1.



(b) AUPRC.

Figure 3. **Label-efficiency heatmaps for frozen-feature hazard probes.** Rows are backbones (with the probe layer in parentheses) and columns are label budgets. Each cell reports the mean score over 10 random trials with disjoint train/test splits and exact-string deduplication to avoid leakage. Even with 50 labels, probes achieve strong performance across all backbones, and scores saturate by 300 labels, supporting our "few safety labels" regime. We report both F1 (thresholded classification) and AUPRC (threshold-free ranking), showing that the concept direction remains separable under limited supervision.

safety metrics. We report standardized jailbreak/benign-refusal proxies using curated harmful/benign prompt pools. These are useful for controlled comparisons, but they are not a substitute for full safety evaluations across domains and interaction settings. (3) *Tokenizer-only mismatch for Unigram backbones.* Extremely large perplexities for tokenizer-only SentencePiece variants indicate tokenizer/model mismatch when changing segmentation without weight adaptation (as already noted in Table 1). This is a practical constraint: Unigram edits generally need at least lightweight adaptation (e.g., LoRA) to remain usable. (4) *Sensitivity to layer and scale.* The effective scale of λ and the best probe layer depend on the backbone, so practitioners should not expect a single global hyperparameter choice to transfer across architectures. Additionally, Unigram priors can increase sequence length (higher TPC), so stability gains may come with compute/latency costs that must be evaluated per deployment setting.

7. Conclusion and Future Work

We introduced alignment-aware tokenization: a small-label safety pipeline that treats tokenization as a first-class lever and pairs it with drift-regularized adapter tuning. Across five backbones, our approach improves segmentation behavior and reduces neutral-context hazard activation while preserving perplexity and token efficiency in the few-label regime.

Several directions are immediately promising. First, extend from a single hazard axis to *multi-concept* sets (e.g., multiple directions or low-rank subspaces) to handle heterogeneous safety categories. Second, improve the tokenizer objective by explicitly optimizing stability under perturbations (building on the "tokenization as a safety lever" agenda laid out in our initial proposal), rather than relying on heuristic merge pruning or fixed priors. Third, expand evaluation to broader safety suites and domains (beyond the current prompt pools) to better characterize generalization and failure modes. Finally, we plan to release an automated, reproducible pipeline for sweeping λ, selecting Pareto points, and exporting tokenizers/adapters for downstream safety work.

Acknowledgements

We thank Professor **Ilia Sucholutsky** for teaching the foundational concepts that motivated this project and for his consistent guidance throughout its implementation. His support and feedback were instrumental in shaping the direction of the work.

We also acknowledge the use of *GPT-5.2 Thinking* for assistance with grammatical editing and for helping check and refine proof sketches and technical arguments.

Impact Statement

This work investigates how tokenization choices influence safety behavior in large language models. By showing that subword segmentation can contribute to spurious hazard activation in low-label regimes, our results highlight tokenization as an underexplored component of alignment pipelines. When applied conservatively and paired with lightweight adaptation, alignment-aware tokenization can reduce over-refusal without degrading language quality.

Our study also has limitations. Tokenizer edits act globally and may introduce model–tokenizer mismatch if applied aggressively, particularly for Unigram-based models, and our evaluation focuses on a narrow class of hazard concepts using proxy benchmarks. Alignment-aware tokenization should therefore be viewed as a complementary technical tool rather than a standalone safety solution.

References

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Chao, P., Debenedetti, E., Robey, A., Andriushchenko, M., Croce, F., Sehwag, V., Dobriban, E., Flammarion, N., Pappas, G. J., Tramèr, F., Hassani, H., and Wong, E. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2024.

Conover, M., Hayes, M., Mathur, A., Xie, J., Wan, J., Shah, S., Ghodsi, A., Wendell, P., Zaharia, M., and Xin, R. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

Gehman, S., Gururangan, S., Sap, M., Choi, Y., and Smith, N. A. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462*, 2020.

Hayou, S., Ghosh, N., and Yu, B. Lora+: Efficient low-rank adaptation of large models. *arXiv preprint arXiv:2402.12354*, 2024. URL https://arxiv.org/abs/2402.12354.

Li, J., Attanasio, G., Gupta, A., Chiang, T., and Ruder, S. Language model tokenizers introduce unfairness between languages. *arXiv preprint arXiv:2305.15425*, 2023. URL https://arxiv.org/abs/2305.15425.

Lin, Y., He, P., Xu, H., Xing, Y., Yamada, M., Liu, H., and Tang, J. Towards understanding jailbreak attacks in llms: A representation space analysis. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7067–7085. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.401. URL https://aclanthology.org/2024.emnlp-main.401/.

Luo, W., Ma, S., Liu, X., Guo, X., and Xiao, C. Jailbreakv-28k: A benchmark for assessing the robustness of multimodal large language models against jailbreak attacks, 2024.

Mazeika, M., Phan, L., Yin, X., Zou, A., Wang, Z., Mu, N., Sakhaee, E., Li, N., Basart, S., Li, B., Forsyth, D., and Hendrycks, D. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal alignment. *arXiv preprint arXiv:2402.04249*, 2024. URL https://arxiv.org/abs/2402.04249.

Schmidt, C. W., Reddy, V., Zhang, H., Alameddine, A., Uzan, O., Pinter, Y., and Tanner, C. Tokenization is more than compression. *arXiv preprint arXiv:2402.18376*, 2024. URL https://arxiv.org/abs/2402.18376.

Zou, A., Wang, Z., Kolter, J. Z., and Fredrikson, M. Universal and transferable adversarial attacks on aligned language models, 2023.

A. Normal-approximation confidence intervals for JB and BR

For the jailbreak success proxy (JB) and benign refusal (BR), we compute 95% normal-approximation confidence intervals over prompts by resampling prompts with replacement within each evaluation split. We omit confidence intervals from the main table for space.

Example. Pythia-410M (standard tok, no drift): JB= 0.4857 [0.4272, 0.5443] (N=280) and BR= 0.0013 [0.0000, 0.0032] (N=1500).

Example (improved setting). Pythia-410M (BPE-search + drift-LoRA, λ=0.25): JB= 0.0319 [0.0101, 0.0536] (N=251) and BR= 0.0020 [0.0000, 0.0043] (N=1500).

Prompting and scoring protocol (sanitized). We evaluate on two pools: (1) an adversarial/jailbreak-style pool consisting of role-play and instruction-override templates that attempt to elicit disallowed behavior, and (2) a benign pool of everyday instructions. Each prompt is fed to the model under the same decoding settings as in the main experiments. We compute a scalar safety score per prompt and calibrate a threshold τ using a held-out calibration set to target a fixed false-positive rate on benign prompts (FPR@1%). JB is reported as the fraction of adversarial prompts whose score exceeds τ (proxying a successful jailbreak), while BR is the fraction of benign prompts that are refused.

Diagnostics (example setting). For Pythia-410M (standard tok, no drift), the attack refusal rate is 3.6% and the benign refusal rate is 0.1%. We additionally log the distribution of safety scores (mean/std on benign) and the top-scoring adversarial examples by score to verify that JB is driven by high-scoring prompts rather than threshold instability.

B. Additional Results: Drift Regularization Strength

Backbone	Layer	Setting	λ	ppl	tpc	$\mu_N \downarrow$	μ_H	gap $\uparrow$	JB $\downarrow$	BR $\approx$
Pythia-410M	10	BPE-search tokenizer only	0.0	24.470	0.224	0.187	0.189	0.002	0.056	0.000
Pythia-410M	10	BPE-search + drift-LoRA	0.25	24.484	0.224	0.187	0.189	0.002	0.032	0.002
Pythia-1.4B	11	BPE-search tokenizer only	0.0	18.974	0.116	0.109	0.348	0.239	0.585	0.000
Pythia-1.4B	11	BPE-search + drift-LoRA	0.25	20.318	0.116	0.098	0.383	0.285	0.518	0.001
Pythia-1.4B	11	BPE-search + drift-LoRA	0.5	22.408	0.116	0.085	0.421	0.336	0.456	0.002
Pythia-1.4B	11	BPE-search + drift-LoRA	1.0	25.330	0.116	0.068	0.474	0.406	0.395	0.003
LLaMA-3-8B	16	SPM priors tokenizer only	0	3469.014	0.417	0.052	0.175	0.122	0.012	0.000
LLaMA-3-8B	16	SPM priors + drift-LoRA	5	10.191	0.417	0.226	0.340	0.114	0.072	0.005
LLaMA-3-8B	16	SPM priors + drift-LoRA	25	10.189	0.417	0.156	0.185	0.028	0.052	0.005
Mistral-7B	16	SPM priors tokenizer only	0	2631.554	0.417	0.129	0.200	0.071	0.000	0.000
Mistral-7B	16	SPM priors + drift-LoRA	5	17.740	0.417	0.129	0.200	0.071	0.000	0.000
Mistral-7B	16	SPM priors + drift-LoRA	25	20.982	0.417	0.129	0.200	0.071	0.012	0.000
Qwen2-7B	16	SPM priors tokenizer only	0	3469.473	0.417	-0.124	-0.026	0.098	0.012	0.000
Qwen2-7B	16	SPM priors + drift-LoRA	5	11.571	0.417	-0.099	0.016	0.115	0.020	0.011
Qwen2-7B	16	SPM priors + drift-LoRA	25	11.570	0.417	-0.099	0.016	0.115	0.048	0.008

Table 3. Small λ grid for drift-regularized LoRA. We evaluate a backbone-specific grid of drift-regularization strengths (Pythia: $\{0, 0.25, 0.5, 1.0\}$; SentencePiece chat models: $\{0, 5, 25\}$) and report the corresponding end-to-end metrics; Table 1 reports the selected best-λ setting.