BEAST: BENCHMARKING EVOLVING ABUSE STREAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Online abusive language evolves continually, so classifiers deployed in real-world moderation systems must continually adapt to text that may no longer resemble what they were trained on. Testing whether adaptation methods are effective requires a benchmark that preserves real-world task constraints—naturally occurring content, a fixed prediction task, a continuous stream, and meaningful volume—while methodologically isolating the challenge of continuous linguistic evolution. No existing continual-learning benchmark for abuse detection combines these properties. Instead, they manufacture distribution shift by sequencing tasks, datasets, bias dimensions, or adversarial perturbation types, which tests adaptation to one specified form of change rather than to language that evolves on its own. We introduce BENCHMARKING EVOLVING ABUSE STREAMS (BEAST), a domain-incremental benchmark that holds the task and moderation policy fixed while only the incoming language changes. BEAST draws millions of naturally occurring posts from a single 3.5-year source and orders them so that incoming text grows progressively less familiar relative to the initial training period. Models are scored on an interval of the stream before that interval becomes available for training. We measure and report predictive performance, adaptation cost, and forgetting: across encoder and decoder models, increasing model scale and periodic retraining both improve average performance yet leave substantial end-to-end degradation, while retrieval augmentation performs far worse despite substantially higher compute. BEAST exposes a persistent gap between current adaptation methods and the robustness that evolving abuse detection demands.

1 INTRODUCTION

Abusive language changes as new slang, abuse targets, coded expressions, and cultural references appear and older forms fall out of use. A classifier deployed in a real moderation system must therefore recognize language absent from its training data without losing accuracy on the content it already catches. This setting closely aligns with domain-incremental continual learning, where the prediction task remains fixed while the input distribution changes over time (van de Ven et al., 2022). Prior work has established that abuse classifiers degrade under temporal distribution shift (Florio et al., 2020; Jin et al., 2023; Di Bonaventura et al., 2025), motivating the need to evaluate not only temporal robustness, but whether adaptation can keep pace with evolving language.

Evaluating this capability requires benchmarks that preserve the operational constraints of production abuse classifiers—a fixed moderation task applied to a high-volume stream of naturally occurring language—while providing a controlled setting to measure adaptation as incoming language becomes increasingly unfamiliar. Existing work studies complementary forms of change, including evolving tasks and annotation schemas (Qian et al., 2021; Omrani et al., 2024), adversarially collected or perturbed examples (Vidgen et al., 2021; Kang et al., 2026a;b), and temporal linguistic drift (Florio et al., 2020; Di Bonaventura et al., 2025; Asiri and Saleh, 2026). These settings capture important aspects of the problem, but do not jointly provide a large stream of naturally occurring user content for evaluating continual adaptation under a fixed moderation task.

To address this gap, we introduce BENCHMARKING EVOLVING ABUSE STREAMS (BEAST), a benchmark for continual adaptation over a large stream of naturally occurring human posts. Rather than ordering the evaluation stream by raw chronological time, BEAST uses perplexity to construct a controlled novelty gradient, providing a controlled setting for measuring how adaptation behaves as incoming language becomes progressively less familiar. BEAST contains 6.04M records annotated

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

from a 3.5-year corpus of a single online community (Papasavva et al., 2020); no text is synthetically generated or perturbed to induce distribution shift. Additionally, BEAST decouples semantic annotation from the moderation policy. An LLM first extracts structured semantic attributes; then, deterministic policy functions derive the final benchmark labels. This approach allows researchers to evaluate alternative policies over the same corpus without rerunning the expensive annotation step.

BEAST evaluates each model on an incoming interval before that interval becomes available for adaptation, following the prequential test-then-adapt protocol standard in the concept-drift literature (Gama et al., 2014). This puts static models and continual adaptation strategies on the same footing while tracking predictive performance, computational cost, and forgetting throughout the stream. Cost is an explicit dimension of the benchmark: across millions of posts, even modest per-example or per-update overhead compounds, so an adaptation method is useful only insofar as its gains justify the compute needed to sustain it.

Under this protocol, no tested configuration maintains its initial performance as novelty increases. Greater model scale and periodic retraining raise average performance but leave substantial end-to-end degradation, and continual retrieval-augmented generation performs markedly worse while consuming more compute than any other adaptation method we evaluate.

2 Related Work

Temporal degradation and adaptation in abuse detection. Prior work shows that abusive-language classifiers are sensitive to temporal distribution shift. Florio et al. (2020) study Italian Twitter data using monthly temporal splits and compare strategies for updating classifiers over time, while Di Bonaventura et al. (2025) evaluate models across evolving hate-speech settings. More recently, Hoffmann et al. (2025) study updating hate-speech classifiers under temporal mismatch. Beyond abuse detection, temporal language-modeling work has shown that performance degrades as models encounter increasingly future data and that continual updating can mitigate this degradation (Lazaridou et al., 2021). Together, these works establish temporal evolution as a practical challenge for abuse detection and motivate adaptation to newly observed language. BEAST builds on this line of work by evaluating adaptation over a large, multi-step stream of naturally occurring posts as the encountered language becomes increasingly unfamiliar.

Dynamic and continual abuse detection. Other work studies how abuse-detection systems should adapt as their evaluation environment changes. DynaHate (Vidgen et al., 2021) dynamically collects challenging examples through repeated human-and-model-in-the-loop adversarial rounds. In continual-learning settings, Qian et al. (2021) formulate hate-speech classification as a lifelong learning problem over a sequence of hate-ideology tasks, while Omrani et al. (2024) construct 84 related problematic-content tasks across heterogeneous datasets and annotation schemas. DynDetect and ContiGuard instead study domain-incremental toxicity detection across successive adversarial or evasive perturbation types (Kang et al., 2026a;b). Closest to our setting, Asiri and Saleh (2026) study continual offensive-language detection under temporal linguistic drift in Saudi-dialect Arabic using distinct historical and contemporary datasets. These settings capture complementary forms of change through adversarial data collection, evolving tasks, perturbation domains, or discrete temporal shifts. BEAST instead evaluates repeated adaptation under a fixed moderation task over a single 3.5-year longitudinal corpus of naturally occurring posts, producing a 20-step stream of 6.04M benchmark records.

Table 1: Comparison with closely related dynamic, temporal, and continual abuse-detection benchmarks.

Study	Natural data	Temporal	Fixed task	Continual adaptation	Scale
Vidgen et al. (2021)	×	×	✓	✓	40K
Omrani et al. (2024)	×	×	×	✓	$< 2.49\text{M}^{\dagger}$ (84 tasks)
Di Bonaventura et al. (2025)	✓	✓	✓	×	3K
Kang et al. (2026a)	×	×	✓	✓	38K
BEAST (ours)	✓	✓	✓	✓	**6.04M**

† Scale approximated because the full benchmark is not publicly released.

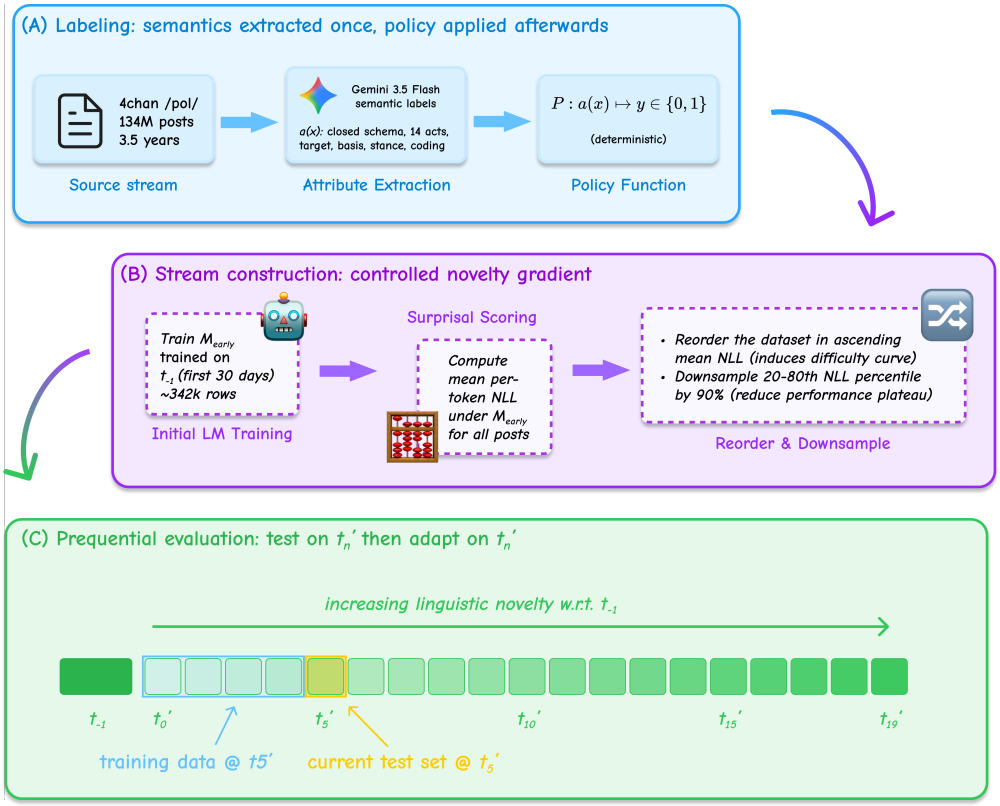


Figure 1: Overview of BEAST. Structured semantic attributes are extracted once and mapped to labels by a deterministic policy, while the source corpus is reordered into a controlled novelty gradient for prequential test-then-adapt evaluation.

3 Benchmark Methodology

BEAST is designed to capture several properties of deployed abuse-detection settings: naturally occurring user-generated content, a fixed moderation task, sequentially arriving data from a single source, and a scale at which the cost of adaptation is operationally meaningful. The benchmark preserves these properties while controlling the degree of linguistic novelty presented over the evaluation stream. Figure 1 summarizes the construction.

3.1 Dataset

BEAST is constructed from an initial subset of 12.73M annotated records uniformly sampled from a 3.5-year longitudinal corpus totaling 134.5M posts (Papasavva et al., 2020). All benchmark text consists of naturally occurring posts from a single community; no examples are synthetically generated or perturbed to induce distribution shift. We apply minimal deterministic preprocessing and otherwise preserve the original linguistic form, including capitalization, punctuation, slang, misspellings, and community-specific markers such as greentext. Full preprocessing details are provided in Appendix B.

The resulting benchmark contains an initial 30-day slice t_{-1} of 342,061 records and 5.70M records in the novelty-ordered evaluation stream. Together, these form 6.04M records used by the final benchmark. Dataset statistics are summarized in Table 2.

Structured Semantic Annotation. Annotating a corpus of this size by hand is infeasible, so we label it with Gemini 3.5 Flash (Google DeepMind, 2026). Rather than eliciting a binary moderation label directly, the model extracts a fine-grained attribute vector $a(x)$ over a closed schema, from which a separate policy function derives the label. Prior work motivates this decomposition: breaking a complex judgment into concrete checklist-style decisions improves the reliability of LLM annotation (Lee et al., 2025). The schema covers attack act (14 values), target type, identity basis, stance, coding, and idiomatic usage.

The annotator sees only the isolated post text, with no timestamps, thread context, or other metadata. This withholds explicit temporal signal from the labels and matches the text-only information available to the classifiers we evaluate. Table 8 gives the full closed-set ontology and Table 7 the inference configuration; the complete prompt appears in the supplementary material. These labels have not been validated against an independent human-annotated subset, so we treat them as benchmark labels rather than as a proxy for human ground truth (Section 5).

Abuse Policy Function. We separate semantic extraction from the final moderation decision. Given the extracted attribute vector $a(x)$, a configurable policy function P deterministically assigns a binary benchmark label $y \in \{0, 1\}$:

$$P : a(x) \mapsto y.$$

Because the underlying semantic attributes are extracted only once, the same annotations can support multiple policy configurations without rerunning the annotation model. For the primary BEAST evaluation, both the annotation schema and policy P remain fixed across all timesteps; only the distribution of incoming language changes. This corresponds to the domain-incremental setting motivated in Section 1 rather than a sequence in which the prediction task itself changes.

The primary policy uses a hierarchical decision tree that captures severe threats while accounting for contextual exemptions such as educational framing and counter-speech. Unlike a scalar severity threshold, this representation can express non-monotonic rules over the extracted attributes. Policy variants can therefore be recomputed inexpensively as long as the distinctions required by the policy are represented in the fixed attribute schema. The complete policy logic is provided in Appendix B.

3.2 Controlled Novelty Gradient

Evaluated in its original chronological order, this corpus does not expose systematic degradation. Static-model performance stays broadly stable across the 3.5 years (Appendix D) even though lexical and semantic diagnostics confirm that the underlying language does change (Appendix E). This stability likely results from data dilution. At any given point, the influx of familiar language outweighs novel terms, obscuring degradation on the evolving subset within aggregate metrics. We therefore construct a controlled novelty gradient by ordering the corpus so that incoming posts become progressively less probable under the language available during the initial training period. BEAST retains the naturally occurring content of the source stream while controlling how much novelty the classifier encounters at each point.

Concretely, we tokenize posts on whitespace and train a 5-gram KenLM language model (Heafield, 2011), denoted M_{early}, on all 342,061 records in the initial slice t_{-1}, rather than on the subset used to fit the classifiers. The vocabulary is fixed from this period, and words outside it are mapped to `<unk>` during scoring. For every remaining record $x = (w_1, \ldots, w_{|x|})$ we compute the mean per-word negative log-likelihood

$$\mathrm{NLL}_{\text{early}}(x) = -\frac{1}{|x|} \sum_{i=1}^{|x|} \log_{10} P_{M_{\text{early}}} \left(w_i \mid w_{i-4}^{i-1} \right),$$

where higher values indicate language less probable under the initial-period distribution. Perplexity is the exponential of this quantity and induces the same ordering, so we work with $\mathrm{NLL}_{\text{early}}$ throughout.

The central mass of this distribution would otherwise dominate the stream, so we keep all records below the 20th and above the 80th percentile of $\mathrm{NLL}_{\text{early}}$ but subsample the middle band to 10% of its original size. Sorting the 5.70M retained records in ascending order and dividing them into

20 intervals yields the evaluation timesteps $t'_0, \ldots, t'_{19}$. Discarding 90% of the middle band is a benchmark-construction choice rather than a principled threshold; results therefore characterize performance under this particular concentration of novelty and not the rate of degradation expected in chronological traffic.

Appendix F validates the resulting construction: Figure 7 reports both the source NLL distribution and its progression across reordered timesteps, while Table 10 provides representative examples from early, middle, and late portions of the stream.

Table 2: Dataset statistics for BEAST.

Source corpus	134.5M posts
Annotated records	12.73M
Final benchmark records	6.04M
Initial training slice t_{-1}	342,061
Novelty-ordered evaluation records	5.70M
Time span	3.5 years
Pseudo-timesteps	20
Positive rate under P	15.73% overall

3.3 Evaluation Setup and Metrics

Prequential Setup. We employ a prequential (test-then-adapt) evaluation protocol (Gama et al., 2014). Candidate models are evaluated sequentially on each pseudo-timestep t'_n before examples from that timestep become available for adaptation. The classification task and moderation policy remain fixed throughout the stream; only the distribution of incoming examples changes. The benchmark consists of the initial training slice t_{-1} followed by 20 novelty-ordered evaluation intervals.

Classification Performance. We report PR-AUC independently at each pseudo-timestep. Because PR-AUC depends on class prevalence, we additionally report the positive-label rate at each timestep and interpret cross-timestep changes in PR-AUC together with changes in prevalence. The positive rate is 15.7% overall and ranges from 10.5% to 19.7% across timesteps; complete class counts are reported in Table 9.

Compute Costs (FLOPs). We track computational cost as an explicit evaluation dimension because real-world moderation systems must process large traffic volumes, causing even modest per-example or per-update overhead to compound over deployment. Therefore, an adaptation method is practically viable only if its predictive gains outweigh its computational overhead. Following the compute-aware perspective of continual adaptation benchmarks such as TiC-LM (Li et al., 2025), we report lifecycle FLOPs alongside predictive performance. Depending on the adaptation regime, this includes initial training, streaming inference, retraining, and retrieval embedding and indexing costs.

Backward Transfer (BWT). Adaptation to newly encountered forms of abuse should not come at the expense of performance on previously observed content that continues to be present in the online traffic. To quantify catastrophic forgetting, adapted models at timestep t'_n are therefore re-evaluated on historical data from previous intervals $(t'_0, \ldots, t'_{n-1})$. We adapt backward transfer (Lopez-Paz and Ranzato, 2017) to the streaming setting. Let $R_{c,t}$ denote PR-AUC on timestep t under adaptation state c, and let $c(t)$ be the first adaptation state whose available data includes timestep t. If c_K is the final adaptation state, we define

$$\mathrm{BWT} = \frac{1}{|\mathcal{H}|} \sum_{t \in \mathcal{H}} \left(R_{c_K,t} - R_{c(t),t} \right),$$

where $\mathcal{H}$ contains timesteps incorporated before the final retraining block. For periodic retraining, an adaptation state corresponds to a model checkpoint; for RAG, it corresponds to the state of the retrieval index. Negative BWT indicates forgetting.

Full adaptation protocols and architecture-specific settings are provided in Appendix A. Full forgetting grids and detailed FLOP breakdown are reported in Appendix G.

4 EXPERIMENTS AND RESULTS

4.1 EXPERIMENTAL SETUP

We evaluate BERT-Large and ModernBERT-Large (Devlin et al., 2019; Warner et al., 2025) alongside Qwen 3.5 causal language models (Qwen Team, 2026). BERT-Large, ModernBERT-Large, and Qwen 3.5 0.8B are evaluated under the adaptation regimes below, while Qwen 3.5 4B and 9B are additionally included as static baselines to study the effect of model scale. All methods begin from a supervised initialization on the initial slice t_{-1}, using at most 100,000 labeled examples split into training, validation, and backward-transfer holdout sets. Encoder models use standard classification heads. For causal decoders, we formulate abuse detection through the language-model output by applying two-class cross-entropy to the `Yes`/`No` logits at the answer position and deriving the binary class probability from the same logits. The complete decoder classification interface and prompt templates are provided in Appendix A.2.

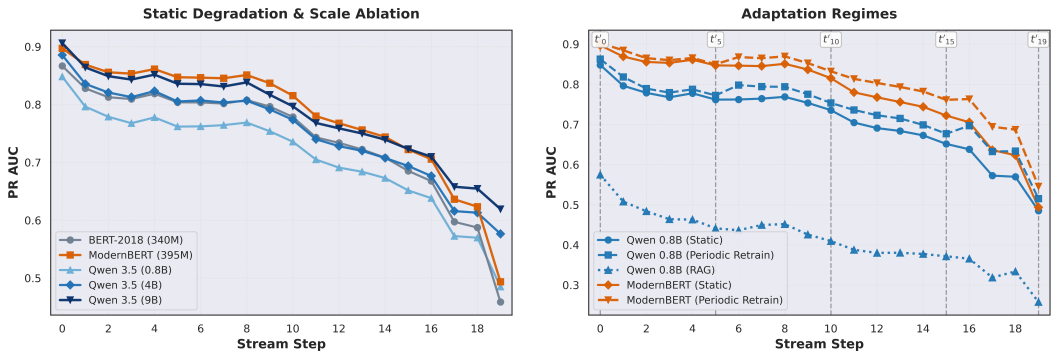
Static. In this setting, models are trained only on t_{-1} and then evaluated across all 20 pseudo-timesteps with frozen parameters.

Periodic retraining. Models are updated every five timesteps ($N = 5$) using up to 100,000 newly observed examples since the previous update, with an additional final update after t'_{19} for backward-transfer evaluation. Each update warm-starts from the preceding checkpoint, and historical holdouts are re-evaluated after adaptation to measure backward transfer.

Continual RAG. For Qwen 3.5 0.8B, model parameters remain fixed while the retrieval corpus grows with the stream. For each query, we retrieve the $k = 5$ most similar previously observed labeled examples using `BAAI/bge-small-en-v1.5` embeddings and a flat FAISS index (Johnson et al., 2021; Xiao et al., 2024), and prepend them to the query. Examples from t'_n enter the index only after evaluation on t'_n, preserving the test-then-adapt protocol. We evaluate RAG only for the causal decoder because retrieved demonstrations are incorporated directly through its prompt.

All models use AdamW with learning rate 3×10^{-5}, weight decay 0.01, and early stopping. Larger Qwen models use LoRA for the static model-size ablation. Architecture-specific batch sizes, sequence lengths, hardware, and optimization settings are reported in Table 5 and Table 6; the full adaptation protocol is summarized in Table 4. We report production-relevant FLOPs from initial training, streaming inference, parameter updates, and retrieval-specific computation, excluding historical re-evaluation performed solely for backward-transfer analysis. All configurations use a single training seed (42).

4.2 RESULTS



(a) Static models **(b)** Adaptation regimes

Figure 2: PR-AUC across the novelty-ordered evaluation stream. *(a)* Static performance across model architectures and sizes. *(b)* Comparison of static, periodic retraining, and continual RAG configurations. Vertical lines indicate periodic retraining events at t'_0, t'_5, t'_{10}, t_15' and t'_{19}.

Table 3: Aggregate performance across the novelty-ordered stream. Mean is averaged across evaluation timesteps, and Δ denotes the change from the beginning to the end of the stream. BWT measures backward transfer after adaptation. FLOPs include production-relevant compute only; historical re-evaluation for BWT is excluded.

		PR-AUC		P@90R		R@90P			
Model	**Regime**	**Mean**	**Δ**	**Mean**	**Δ**	**Mean**	**Δ**	**BWT**	**FLOPs**
BERT-Large	Static	0.7416	−0.4081	0.4114	−0.4314	0.3883	−0.6152	–	1.20×10^{18}
ModernBERT-L	Static	0.7804	−0.4036	0.4754	−0.4684	0.4366	−0.6032	–	1.41×10^{18}
Qwen 3.5 0.8B	Static	0.7093	−0.3632	0.3467	−0.4572	0.2337	−0.6971	–	1.38×10^{18}
Qwen 3.5 4B	Static	0.7519	−0.3091	0.4323	−0.4561	0.3103	−0.7083	–	$7.75\times10^{18\dagger}$
Qwen 3.5 9B	Static	0.7825	−0.2870	0.4720	−0.5181	0.4043	−0.7248	–	$1.65\times10^{19\dagger}$
BERT-Large	Retrain (N=5)	0.7744	−0.3827	0.4467	−0.4087	0.4400	−0.5961	+0.0085	1.73×10^{18}
ModernBERT-L	Retrain (N=5)	**0.8081**	−0.3570	**0.5097**	−0.4582	**0.4955**	−0.6140	+0.0071	2.00×10^{18}
Qwen 3.5 0.8B	Retrain (N=5)	0.7377	−0.3475	0.3667	−0.4078	0.3376	−0.6519	−0.0284	2.35×10^{18}
Qwen 3.5 0.8B	RAG (k=5)	0.4141	−0.3173	0.1834	−0.0602	0.0000	−0.0001	+0.0049	4.99×10^{18}

† Qwen 3.5 4B and 9B FLOPs are estimated by scaling the measured Qwen 3.5 0.8B costs according to the corresponding text-model parameter-count ratios; see Appendix G.3 for the stage-wise breakdown.

Figure 2 and Table 3 summarize performance across BEAST. Additional fixed operating-point results, historical re-evaluations, and stage-wise compute costs are reported in Appendix G.

Robustness remains unsolved. No tested configuration maintains its initial performance as the stream becomes more linguistically unfamiliar. Even the strongest configurations exhibit large end-to-end losses: retrained ModernBERT-Large achieves the highest mean PR-AUC (0.8081) but declines by 0.3570 across the stream, while static Qwen 3.5 9B, which has the smallest PR-AUC decline among the static models, still falls by 0.2870. This pattern extends beyond PR-AUC. P@90R and R@90P also deteriorate substantially for all tested model families (Figures 9 and 8), indicating that none of the evaluated configurations remains robust at either high-recall or high-precision operating points.

Increasing model scale does not prevent degradation. Increasing Qwen model size yields clear gains in aggregate PR-AUC. Scaling from 0.8B to 9B parameters raises mean PR-AUC from 0.7093 to 0.7825 and reduces the PR-AUC decline from -0.3632 to -0.2870. However, the larger models still exhibit severe losses at fixed operating points; Qwen 3.5 9B declines by 0.5181 in P@90R and 0.7248 in R@90P. These gains also come at substantial computational cost. ModernBERT-Large nearly matches Qwen 3.5 9B in mean PR-AUC (0.7804 versus 0.7825) while requiring 1.41×10^{18} FLOPs compared with an estimated 1.65×10^{19} FLOPs for Qwen 3.5 9B. Thus, larger decoder models improve average performance, but scaling alone does not provide robustness commensurate with its additional compute.

Standard adaptation strategies do not prevent degradation. Periodic retraining consistently improves average performance relative to the corresponding static models: mean PR-AUC increases from 0.7416 to 0.7744 for BERT-Large, from 0.7804 to 0.8081 for ModernBERT-Large, and from 0.7093 to 0.7377 for Qwen 3.5 0.8B. However, these gains do not keep pace with the stream. All three retrained models still exhibit substantial end-to-end degradation, including a 0.3570 PR-AUC decline for the best-performing configuration. Retraining also increases lifecycle compute by approximately 44% for BERT-Large, 42% for ModernBERT-Large, and 70% for Qwen 3.5 0.8B. Backward transfer is close to zero for the two encoders ($+0.0085$ and $+0.0071$), while Qwen 3.5 0.8B shows modest negative transfer (-0.0284), suggesting that catastrophic forgetting is not the primary limitation of encoder retraining in this setting.

The tested continual RAG configuration performs substantially worse. Mean PR-AUC falls from 0.7093 for static Qwen 3.5 0.8B to 0.4141 with RAG, while mean R@90P is effectively zero. At the same time, total compute increases to 4.99×10^{18} FLOPs, approximately $3.6\times$ that of the static Qwen 3.5 0.8B configuration. The breakdown in Table 11 shows that this overhead is driven primarily by

retrieval embedding and longer-context inference rather than index construction and lookup. Together, these results show that neither increased model scale nor the tested standard adaptation strategies close the robustness gap exposed by BEAST.

5 Conclusion and Limitations

BEAST is a deployment-motivated, domain-incremental continual-learning benchmark for abuse detection built from millions of naturally occurring human posts. It captures several properties of real-world moderation systems that are largely absent from existing continual abuse benchmarks: a fixed prediction task, a single longitudinal source, sequential adaptation, production-scale traffic, and explicit accounting for adaptation cost. Our experiments expose a substantial gap in current methods: increasing model scale improves average performance but does not provide sustained robustness, while periodic retraining and continual RAG likewise fail to keep pace with increasingly unfamiliar language. Together, these results suggest that compute-efficient adaptation to evolving abusive language remains an open problem.

BEAST has several limitations. The corpus is drawn from a single community and primarily one language, so its findings may not generalize to other platforms, populations, or moderation domains. Benchmark labels are produced by an LLM-based structured annotator and are not independently validated against a human-annotated reference set. BEAST intentionally concentrates linguistic novelty to expose robustness failures that are obscured under the original chronological mixture; consequently, its results should be interpreted as performance under increasing linguistic novelty rather than as an estimate of the rate or magnitude of degradation in chronological deployment traffic. Finally, we evaluate one primary moderation policy, a limited set of model and adaptation strategies, and one training seed per configuration. The structured annotation and policy abstraction are designed to support broader policies, policy changes, and future continual-learning methods.

Ethics and Data Statement

BEAST is constructed from a previously published, publicly available corpus of online posts (Papasavva et al., 2020). Since we operate on the public corpus as released, we do not perform additional removal or anonymization of personally identifiable information beyond the preprocessing described in Appendix B. The data may therefore contain offensive, hateful, sexually explicit, or otherwise harmful language, as well as information that users should treat as potentially sensitive.

BEAST is intended for research on abuse detection, robustness, and continual learning. It should not be used to facilitate harassment, profiling, surveillance, deanonymization, or the generation of abusive content. In particular, the benchmark is not intended as a training corpus for models whose purpose is to generate harmful language. Any use or redistribution of the underlying data should comply with the terms and licensing conditions of the original corpus.

AI Use Disclosure

In this work, we used generative AI tools to generate synthetic annotations for our dataset, implement experimental and visualization code, assist in drafting and editing the paper, and provide feedback on research methodology and experimental design. We did not use generative AI tools to interpret experimental results, develop the conceptual framework or evaluation framework of the work, formulate or prove mathematical claims, or create conceptual diagrams.

We take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

References

A. Papasavva, S. Zannettou, E. De Cristofaro, G. Stringhini, and J. Blackburn. Raiders of the lost Kek: 3.5 years of augmented 4chan posts from the Politically Incorrect board. In *Proceedings of the Fourteenth International AAAI Conference on Web and Social Media (ICWSM)*, pages 885–894, 2020.

K. Florio, V. Basile, M. Polignano, P. Basile, and V. Patti. Time of your hate: The challenge of time in hate speech detection on social media. *Applied Sciences*, 10(12):4180, 2020.

M. Jin, Y. Mu, D. Maynard, and K. Bontcheva. Examining temporal bias in abusive language detection. *arXiv preprint arXiv:2309.14146*, 2023.

C. Di Bonaventura, B. McGillivray, Y. He, and A. Meroño-Peñuela. Hatevolution: What static benchmarks don't tell us. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17695–17707, 2025.

M. Hoffmann, J. Fillies, J. John, A. Maronikolakis, A. Navilarekal, S. Udupa, and N. Hammer. Temporal robustness in hate speech detection: Updating German classifiers with advanced AI infrastructures. In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI 2025)*, pages 1171–1178, 2025.

A. Asiri and M. Saleh. Continual learning for Saudi-dialect offensive-language detection under temporal linguistic drift. *Information*, 17(1):99, 2026.

B. Vidgen, T. Thrush, Z. Waseem, and D. Kiela. Learning from the worst: Dynamically generated datasets to improve online hate detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1667–1682, 2021.

J. Qian, H. Wang, M. ElSherief, and X. Yan. Lifelong learning of hate speech classification on social media. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2304–2314, 2021.

A. Omrani, A. Salkhordeh Ziabari, P. Golazizian, J. Sorensen, and M. Dehghani. Towards a unified framework for adaptable problematic content detection via continual learning. In *Proceedings of the 8th Workshop on Online Abuse and Harms (WOAH 2024)*, pages 68–109, 2024.

H. Kang, J. Chen, Y. Li, X. Miao, M. Xu, S. Zhou, M. Zhong, Y. Zhu, and T. Qian. Developing continuous toxicity detection against increasing types of perturbed toxic text. *Information Processing & Management*, 63(2):104470, 2026.

H. Kang, X. Miao, J. Chen, J. Wen, M. Xu, W. Zhang, W. Lu, and T. Qian. ContiGuard: A framework for continual toxicity detection against evolving evasive perturbations. In *Proceedings of the ACM Web Conference 2026*, pages 4769–4780, 2026.

G. M. van de Ven, T. Tuytelaars, and A. S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4:1185–1197, 2022.

J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4), Article 44, 2014.

D. Lopez-Paz and M. A. Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30, pages 6467–6476, 2017.

J. Li, M. Armandpour, S. I. Mirzadeh, S. Mehta, V. Shankar, R. Vemulapalli, S. Bengio, O. Tuzel, M. Farajtabar, H. Pouransari, and F. Faghri. TiC-LM: A web-scale benchmark for time-continual LLM pretraining. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32231–32273, 2025.

Google DeepMind. Gemini 3.5 Flash model card. May 2026. `https://deepmind.google/models/model-cards/gemini-3-5-flash/`.

Y. Lee, J. Kim, J. Kim, H. Cho, J. Kang, P. Kang, and N. Kim. CheckEval: A reliable LLM-as-a-judge framework for evaluating text generation using checklists. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 15771–15798, 2025.

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

J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021.

S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J.-Y. Nie. C-Pack: Packed resources for general Chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 641–649, 2024.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, 2019.

B. Warner, A. Chaffin, B. Clavié, O. Weller, O. Hallström, S. Taghadouini, A. Gallagher, R. Biswas, F. Ladhak, T. Aarsen, G. T. Adams, J. Howard, and I. Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents. February 2026. `https://qwen.ai/blog?id=qwen3.5`.

A. Lazaridou, A. Kuncoro, E. Gribovskaya, D. Agrawal, A. Liska, T. Terzi, M. Gimenez, C. de Masson d'Autume, T. Kocisky, S. Ruder, D. Yogatama, K. Cao, S. Young, and P. Blunsom. Mind the gap: Assessing temporal generalization in neural language models. In *Advances in Neural Information Processing Systems*, volume 34, pages 29348–29363, 2021.

APPENDIX

A EXPERIMENTAL DETAILS

A.1 TRAINING AND ADAPTATION HYPERPARAMETERS

All methods are initially trained on the same labeled examples from t_{-1}. We sample at most 100,000 labeled examples from the initial slice and partition them into 85% training, 5% validation, and 10% backward-transfer holdout sets. Each model and configuration is trained separately using its corresponding architecture-specific training setup. Table 4 specifies how each method incorporates stream data after this initial fit, and Table 5 reports the architecture-specific optimization and input settings. For the larger Qwen models used in the model-size ablation, we use LoRA in place of full-parameter fine-tuning; the additional ablation-specific settings are given in Table 6.

Table 4: Adaptation protocols used during streaming evaluation. All methods are initially trained on the same labeled examples from t_{-1}.

Setting	**Static**	**Periodic retraining**	**Continual RAG**
Adaptation mechanism	None	Supervised fine-tuning	Retrieval augmentation
Model weights	Frozen	Updated every 5 timesteps	Frozen
Stream data used	None	Examples since previous update	All previously observed examples
Update-set limit	—	100,000 examples	—

Continual RAG configuration. We use `BAAI/bge-small-en-v1.5` with cosine similarity over a flat FAISS index, retrieve $k = 5$ examples, and limit the resulting prompt to 4,096 tokens.

Table 5: Architecture-specific training hyperparameters. Base and RAG denote the standard and retrieval-augmented Qwen configurations, respectively.

	Encoder models		Decoder models	
Parameter	**BERT-Large**	**ModernBERT-Large**	**Base**	**RAG**
Hardware	4× H100	4× H100	8× H100	8× H100
Maximum sequence length	512	512	1,024	4,096
Truncation side	Left	Left	Left	Left
Optimizer	AdamW	AdamW	AdamW	AdamW
Learning rate	3×10^{-5}	3×10^{-5}	3×10^{-5}	3×10^{-5}
Weight decay	0.01	0.01	0.01	0.01
Warmup ratio	0.10	0.10	0.10	0.10
Learning-rate schedule	Linear decay	Linear decay	Linear decay	Linear decay
Maximum epochs	10	10	10	10
Precision	bfloat16	bfloat16	bfloat16	bfloat16
Per-device training batch size	32	32	16	2
Gradient accumulation	—	—	—	8 steps
Per-device evaluation batch size	256	256	32	4
Gradient checkpointing	No	No	Yes	Yes
Early-stopping patience	10 evaluations	10 evaluations	10 evaluations	10 evaluations

All experiments use random seed 42. AdamW uses $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. For static and periodically retrained models, evaluation and checkpointing occur every 200 optimization steps.

Table 6: Additional training settings for the larger Qwen models in the model-size ablation. Settings shared with the base Qwen configuration are reported in Table 5.

Parameter	**Value**
Adaptation method	LoRA
LoRA rank (r)	16
LoRA alpha	32
LoRA dropout	0.05
LoRA target modules	All linear layers
Per-device training batch size	16
Gradient accumulation steps	1
Per-device evaluation batch size	32
Attention implementation	FlashAttention-2
Gradient checkpointing	Yes (`use_reentrant=False`)

A.2 Decoder Classification Interface

Unlike the encoder models, which use a standard classification head, our experiments adapt causal decoders to binary classification through their language-model outputs. For each example, we identify the sequence position at which the model produces its answer and extract the vocabulary logits for the `Yes` and `No` tokens at that position. These two logits are treated as the logits of a binary classification problem, and training applies two-class cross-entropy against the binary label.

At inference time, we renormalize the same two logits to obtain the predicted class probability. Let z_{Yes} and z_{No} denote the corresponding logits. The positive-class probability is

$$p(y = 1 \mid x) = \frac{\exp(z_{\mathrm{Yes}})}{\exp(z_{\mathrm{Yes}}) + \exp(z_{\mathrm{No}})}.$$

The same interface is used for both the standard decoder and continual RAG configurations.

Standard decoder prompt. The static and periodically retrained decoder models receive the target post using the following template:

[System]
You are a harmful content detection assistant for comments. Answer by ONLY outputting Yes or No.

[User]
Is the following comment harmful?
<comment>
  {text}
</comment>
Answer:

Continual RAG prompt. For continual RAG, the top-$k = 5$ previously observed labeled examples are retrieved and prepended to the same target query. No model parameters are updated after the initial training stage.

[System]
You are a harmful content detection assistant for comments. Answer by ONLY outputting Yes or No.

[User]
Past Examples:
<comment>
  {historical_post_1}
</comment>
Answer: {Yes/No}
<comment>
  {historical_post_2}
</comment>
Answer: {Yes/No}
⋮
[repeated for up to $k = 5$ retrieved examples]

Is the following comment harmful?
<comment>
  {query_text}
</comment>
Answer:

B Dataset Construction Details

Preprocessing. We apply minimal deterministic preprocessing to the raw 4chan posts before annotation. We remove HTML tags, decode HTML entities (e.g., > → >), remove numeric reply references (e.g., »123456), collapse consecutive whitespace, and strip leading and trailing whitespace. Posts with missing or empty text, invalid timestamps, or text that becomes empty after cleaning are discarded. We otherwise preserve the original textual content, including capitalization, punctuation, slang, misspellings, and greentext markers. For each retained post, we store its post identifier, timestamp, cleaned text, and thread identifier. Replies use their `resto` value as the thread identifier, while thread-opening posts use their own post identifier.

Structured annotation. Each retained post is annotated independently using only its cleaned text. The annotator receives no thread context, neighboring posts, timestamps, or other metadata. Rather than directly predicting the final abuse label, the model extracts a structured representation $a(x)$ describing the acts, targets, and other semantic properties expressed in the post. A deterministic policy function P subsequently maps this representation to a binary label.

For each potential attack, the annotator attaches an *act*, *target type*, *identity basis*, *coding*, *stance*, and *idiom* indicator. It additionally predicts a post-level class and auxiliary safety flags. Table 8 summarizes the closed-set fields. The annotator also produces free-text unpacking, a decoded plain-text restatement when necessary, short reasoning, target, implied-statement, and evidence fields for interpretability and quality control; these fields are not consumed by P. The complete annotation

Table 7: Bulk annotation inference configuration.

Parameter	Setting
Annotation model	Gemini 3.5 Flash
Temperature	0.0
Random seed	0
Context window	131,072 tokens
Maximum output length	32768
Batch size	1,024

prompt, including field definitions, decoding instructions, output schema, and in-context examples, is included verbatim in the supplementary material.

Table 8: Closed-set fields produced by the bulk annotation model.

Field	Values
Act	slur; dehumanization; stereotype; contempt; exclusion; conspiracy; incitement violence; glorification violence; atrocity denial; threat; doxxing; insult/harassment; sexual aggression; self-harm incitement
Target type	thread individual; public individual; private individual; group; institution; concept; self; other
Identity basis	race/ethnicity; nationality/immigration; religion; gender/sex; sexual orientation; disability; appearance; age; political ideology; occupation; social class; other/none; unclear
Coding	explicit; coded; implicit
Stance	advances; quotes; opposes; unclear
Idiom	Boolean indicator for generic or fossilized board-specific abusive idioms
Post class	no attacks; advanced; referenced only
Safety flags	profanity; sexually explicit content; gore/shock; self-harm expression; minor sexualization

Abuse policy function. The final abuse label is derived deterministically from the structured annotation rather than predicted by the annotator. Let $a(x)$ denote the annotation of post x. The policy considers only attacks whose stance is `advances` and applies different enforcement criteria according to attack severity and target type.

For compactness, define

$$\mathcal{C} = \{\text{race/ethnicity},\ \text{gender/sex},\ \text{sexual orientation},\ \text{disability}\},$$
$$\mathcal{E} = \{\text{threat},\ \text{incitement violence},\ \text{glorification violence},\ \text{doxxing},\ \text{self-harm incitement},\ \text{sexual aggression}\},$$
$$\mathcal{H} = \{\text{dehumanization},\ \text{atrocity denial}\},$$
$$\mathcal{M} = \{\text{insult/harassment},\ \text{contempt},\ \text{stereotype},\ \text{exclusion},\ \text{conspiracy}\}.$$

For an annotated attack z, let $V(z)$ indicate that it triggers the policy. For attacks with `stance=advances`,

$$V(z) = \underbrace{[\,\mathrm{act}(z) \in \mathcal{E}\,] \wedge \neg[\,\mathrm{act}(z) = \text{self-harm incitement} \wedge \mathrm{target}(z) = \text{self}\,] \wedge \Big(\mathrm{target}(z) \notin \mathcal{T}_{\mathrm{other}} \vee \mathrm{act}(z) \in \mathcal{E}_{\mathrm{other}}\Big)}_{\text{egregious safety}}$$
$$\vee\ \underbrace{[\,\mathrm{target}(z) = \text{group}\,] \wedge [\,\mathrm{basis}(z) \in \mathcal{C}\,] \wedge \Big([\,\mathrm{act}(z) \in \mathcal{H}\,] \vee [\,\mathrm{act}(z) = \text{slur} \wedge \neg\mathrm{idiom}(z)\,]\Big)}_{\text{group-targeted abuse}}$$
$$\vee\ \underbrace{[\,\mathrm{target}(z) \in \mathcal{T}_{\mathrm{private}}\,] \wedge \Big([\,\neg\mathrm{idiom}(z) \wedge (\mathrm{act}(z) \in \mathcal{M} \vee \mathrm{act}(z) = \text{slur})\,] \vee [\,\mathrm{basis}(z) \in \mathcal{C} \wedge \mathrm{act}(z) \in \mathcal{H}\,]\Big)}_{\text{private-individual harassment}}$$
$$\vee\ \underbrace{[\,\mathrm{target}(z) = \text{public individual}\,] \wedge [\,\mathrm{basis}(z) \in \mathcal{C}\,] \wedge [\,\mathrm{act}(z) \in \mathcal{H}\,]}_{\text{public-individual abuse}},$$

where

$$\mathcal{T}_{\mathrm{private}} = \{\text{thread individual},\ \text{private individual}\}, \qquad \mathcal{T}_{\mathrm{other}} = \{\text{institution},\ \text{concept},\ \text{other},\ \text{self}\},$$

and

$$\mathcal{E}_{\text{other}} = \{\text{threat},\ \text{incitement violence},\ \text{glorification violence}\}.$$

The post-level policy is then

$$P(a(x)) = \begin{cases} 1, & \text{if } \texttt{minor_sexualization=true}, \\ 0, & \text{if } \texttt{post_class=referenced_only}, \\ 1, & \text{if there exists an attack } z \text{ with } \texttt{stance=advances} \text{ and } V(z), \\ 0, & \text{otherwise.} \end{cases}$$

The policy therefore depends only on the structured decision fields; free-text unpacking, reasoning, decoded text, evidence spans, and the coding field do not affect the final label. The exact implementation is provided in the supplementary material.

C Dataset Composition

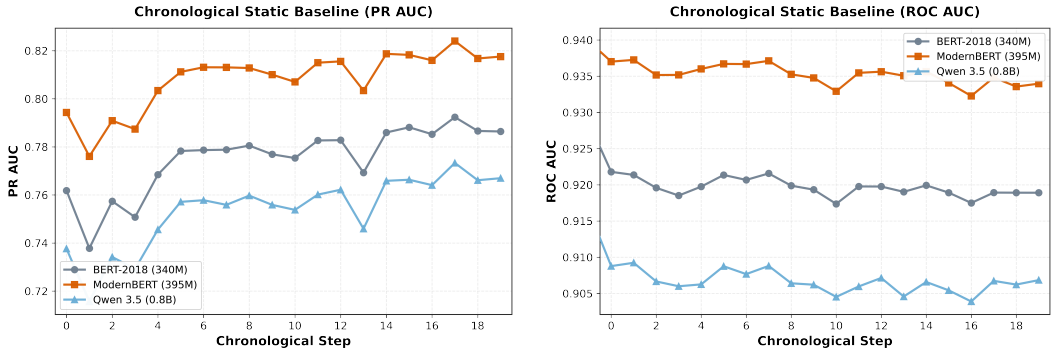
Table 9 reports the class distribution of the initial training slice and each novelty-ordered evaluation timestep. The positive class constitutes 15.73% of the benchmark overall and varies from 10.46% to 19.70% across evaluation timesteps. Because PR-AUC depends on class prevalence, we report these rates to make changes in the underlying class distribution explicit when interpreting performance across the stream.

Table 9: Class balance and record distribution across the initial training slice and the 20 novelty-ordered evaluation timesteps.

Timestep	**Total**	$y=0$	$y=1$	$y=1$ **(%)**
t_{-1} (train)	342,061	289,254	52,807	15.44%
t'_0	284,849	248,767	36,082	12.67%
t'_1	284,848	248,303	36,545	12.83%
t'_2	284,849	243,368	41,481	14.56%
t'_3	284,848	242,002	42,846	15.04%
t'_4	284,849	238,658	46,191	16.22%
t'_5	284,847	239,385	45,462	15.96%
t'_6	284,850	238,237	46,613	16.36%
t'_7	284,847	233,845	51,002	17.91%
t'_8	284,848	232,730	52,118	18.30%
t'_9	284,849	228,794	56,055	19.68%
t'_{10}	284,848	228,722	56,126	19.70%
t'_{11}	284,848	233,000	51,848	18.20%
t'_{12}	284,848	234,597	50,251	17.64%
t'_{13}	284,851	234,819	50,032	17.56%
t'_{14}	284,846	236,194	48,652	17.08%
t'_{15}	284,848	239,946	44,902	15.76%
t'_{16}	284,849	240,123	44,726	15.70%
t'_{17}	351,713	314,251	37,462	10.65%
t'_{18}	217,983	188,909	29,074	13.34%
t'_{19}	284,849	255,043	29,806	10.46%
Overall	**6,039,028**	**5,088,947**	**950,081**	**15.73%**

D Chronological Evaluation

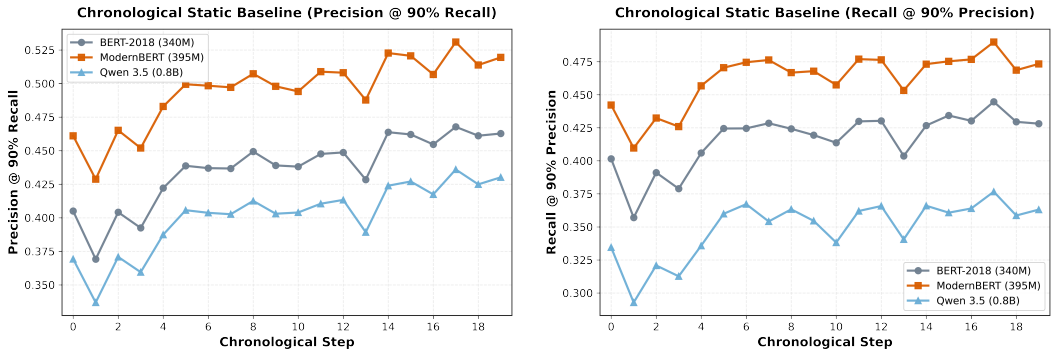
We first evaluate static models under the original chronological ordering of the corpus. Figure 3 shows no systematic degradation over time. PR-AUC generally increases across the stream, while ROC-AUC remains approximately stationary. We hypothesize that this occurs because the mass of linguistically drifting examples is dominated by a larger stationary component of the chronological distribution. As a result, changes within the evolving portion of the stream may be diluted when performance is measured over the full chronological ordering.



(a) PR-AUC **(b)** ROC-AUC

Figure 3: Static-model performance under the original chronological ordering. PR-AUC generally increases across the stream, while ROC-AUC remains approximately stationary. Chronological aggregation therefore does not expose the degradation observed under BEAST's novelty-ordered evaluation.

Figure 4 shows the same pattern at fixed operating points. Precision at 90% recall and recall at 90% precision both generally improve rather than degrade over time. Together with the stable ROC-AUC, these results suggest that the absence of chronological degradation is not an artifact of a single evaluation metric. Instead, they are consistent with the hypothesis that chronological evaluation is dominated by relatively stationary examples, masking degradation on the smaller portion of the distribution undergoing linguistic drift.



(a) Precision at 90% recall **(b)** Recall at 90% precision

Figure 4: Static-model performance at fixed operating points under chronological evaluation. Both precision at 90% recall and recall at 90% precision marginally increase across the stream, consistent with the absence of chronological degradation in Figure 3.

E Chronological Drift Diagnostics

The absence of degradation under chronological evaluation does not imply that the underlying language distribution is stationary. We therefore directly measure distributional change in the original chronological stream using both lexical and semantic diagnostics.

Together, Figures 5 and 6 show that the source corpus does evolve chronologically even though the shift is not sufficiently concentrated to induce systematic degradation under the natural temporal ordering.

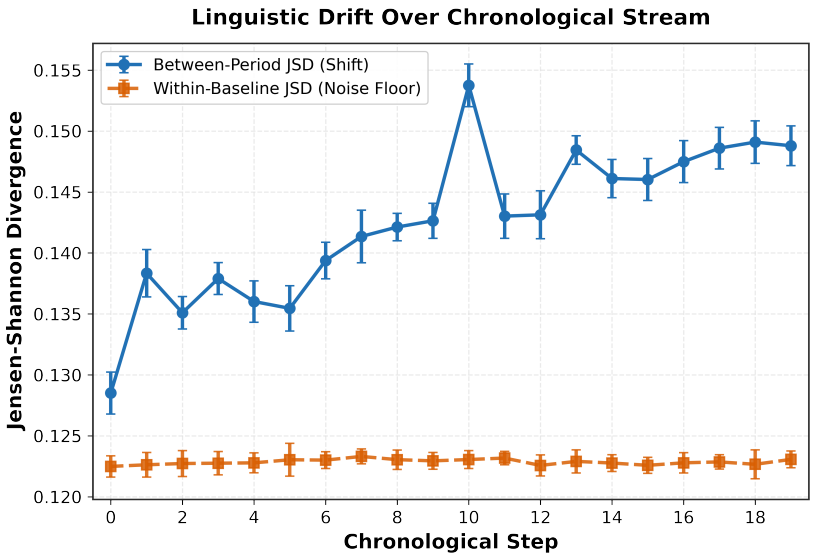


Figure 5: Lexical distribution shift across the original chronological stream. Jensen–Shannon divergence is computed over the global tokenizer vocabulary; the within-baseline noise floor is estimated by bootstrap resampling over 10 random permutations. Between-period divergence remains consistently above this baseline and generally increases over time, indicating measurable chronological change despite stable static-model performance.

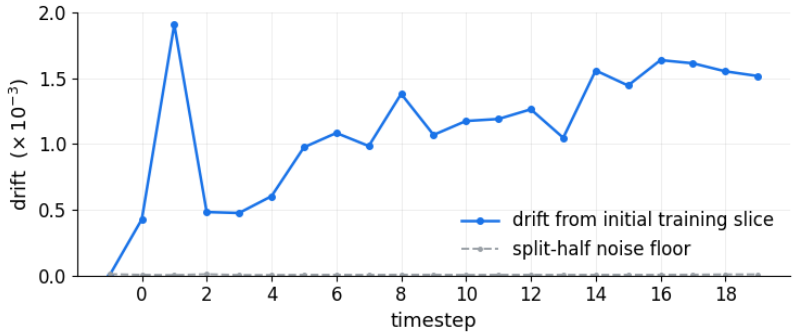
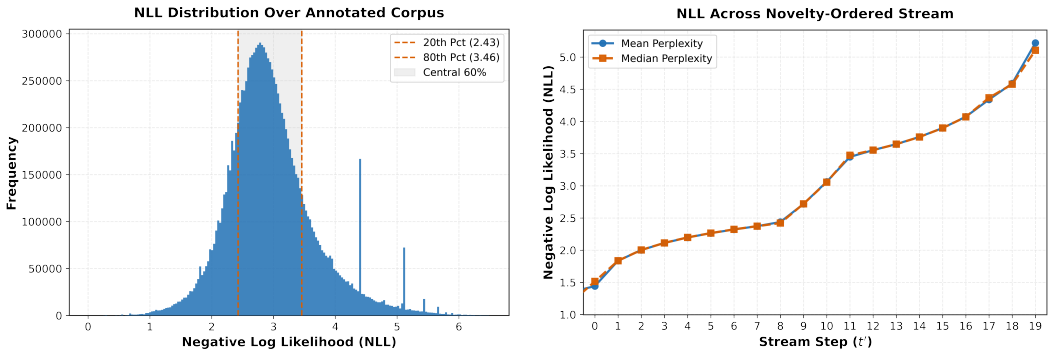


Figure 6: Semantic drift across the original chronological stream. We report distance from the initial training-slice semantic centroid over time, with the split-half noise floor shown for reference. Semantic movement is small in absolute magnitude but non-zero and generally larger in later periods, providing an embedding-based complement to the lexical JSD analysis in Figure 5.

F VALIDATION OF THE NOVELTY-ORDERED STREAM

BEAST converts the diffuse chronological variation above into a controlled novelty gradient using negative log-likelihood under the 5-gram language model trained on the initial slice t_{-1}. We provide two diagnostics of this construction in Figure 7: the distribution of NLL over the annotated corpus and the resulting progression of NLL across pseudo-timesteps.



(a) Corpus-wide NLL distribution

(b) NLL across reordered stream steps

Figure 7: Diagnostics of the novelty-based stream construction. **(a)** Distribution of per-post negative log-likelihood under the 5-gram language model trained on t_{-1}, showing the dominant central mass and higher-surprise tail of the source corpus. **(b)** Mean and median NLL within each reordered pseudo-timestep. Both increase monotonically across the stream, confirming that the reordering produces the intended progression from familiar to increasingly surprising language.

To illustrate the effect of the perplexity-based reordering, Table 10 shows examples from the beginning, middle, and end of the evaluation stream. We use randomly sampled posts from the initial timestep rather than the lowest-scoring examples, which are dominated by degenerate repetitions. The examples are intended to illustrate the linguistic progression induced by the ordering rather than exhaustively characterize each timestep. Higher mean negative log-likelihood (NLL) indicates greater surprise under the 5-gram language model trained on the initial slice t_{-1}.

Table 10: Illustrative examples from early, middle, and late positions in the novelty-ordered stream. Mean NLL is computed under the 5-gram language model trained on t_{-1}; higher values indicate greater linguistic novelty relative to the initial-period distribution.

Stream position	**Post**	**Mean NLL**
Early (t'_0)	Top kek	1.47
	Don't care if you don't believe it, doesn't change the fact that it happened though.	1.61
Middle (t'_{10})	hello. if nazism is so great why its not working until this day ? Also, since technology, are you guys taking DNA tests to trace your origins ?	2.88
	omnifags never broke free from the conditioning and brainwashing never do they question where their food comes from and why they consume it meat eaters are the best examples of npcs	2.88
Late (t'_{19})	"The happenings/ride never end." – Prophet Dr.Ron.Paul	4.81
	'Ate the Sheriff 'Ate the rich 'Ate taxes Luv stealin' Luv shootin' Luv givin' to the poor Luv the Merry Men Luv Sherwood Forest Simple as	4.81

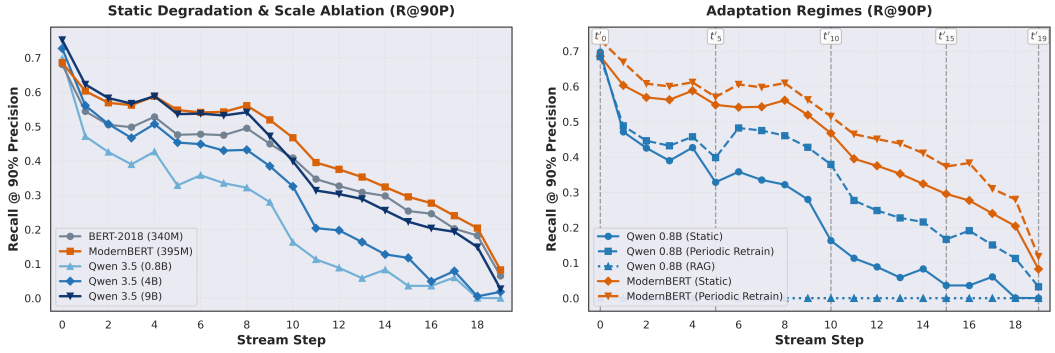
Abuse labels are omitted because they do not enter the stream construction: the ordering depends only on linguistic surprise relative to t_{-1}.

G Additional Evaluation Results

The main evaluation uses PR-AUC as the primary classification metric. Here, we provide complementary results at fixed high-precision and high-recall operating points, together with the full historical re-evaluations underlying the backward-transfer analysis.

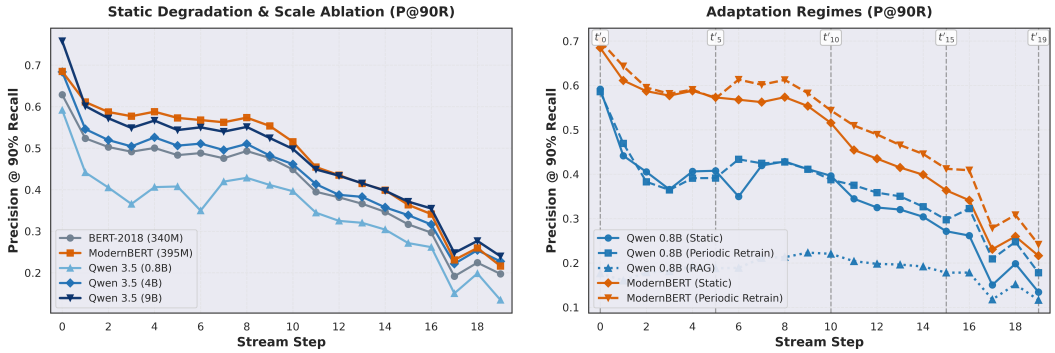
G.1 Fixed Operating-Point Results

We additionally evaluate recall at 90% precision (R@90P) and precision at 90% recall (P@90R). R@90P measures how much abusive content can be recovered while maintaining a high-precision operating point, whereas P@90R measures precision when recall is constrained to remain high. Figures 8 and 9 report these metrics for the same static-model and adaptation comparisons used in the primary PR-AUC evaluation.



(a) Static models **(b)** Adaptation regimes

Figure 8: Recall at 90% precision (R@90P) across the novelty-ordered stream. **(a)** Static-model comparison. **(b)** Comparison of adaptation regimes. R@90P measures recall while constraining predictions to a fixed high-precision operating point.



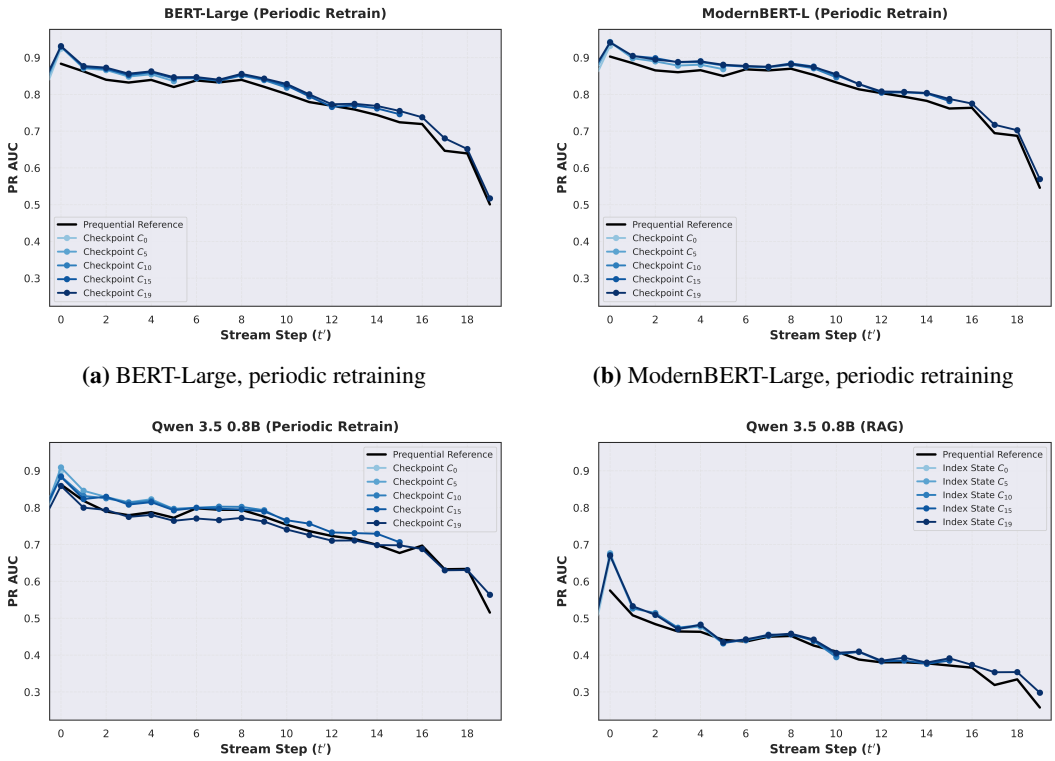
(a) Static models **(b)** Adaptation regimes

Figure 9: Precision at 90% recall (P@90R) across the novelty-ordered stream. **(a)** Static-model comparison. **(b)** Comparison of adaptation regimes. P@90R measures precision while constraining predictions to a fixed high-recall operating point.

G.2 Historical Performance and Backward Transfer

The scalar BWT values reported in the main results summarize a larger set of historical re-evaluations. At each adaptation checkpoint, we re-evaluate the current system on previously observed evaluation intervals. Figure 10 reports the complete PR-AUC matrices for each periodically retrained model and for continual RAG. For retrained models, differences between these historical scores and their original online evaluations are used to compute BWT. The RAG grid provides the analogous view of

historical performance as the retrieval index evolves, while the underlying decoder weights remain fixed.



(a) BERT-Large, periodic retraining

(b) ModernBERT-Large, periodic retraining

(c) Qwen 3.5 0.8B, periodic retraining

(d) Qwen 3.5 0.8B, continual RAG

Figure 10: Historical PR-AUC re-evaluations across the stream. Each matrix reports performance on previously observed evaluation intervals under successive adaptation checkpoints. **(a–c)** Periodically retrained BERT-Large, ModernBERT-Large, and Qwen 3.5 0.8B; these historical re-evaluations underlie the BWT values reported in the main results. **(d)** Continual RAG, showing the corresponding historical performance as the retrieval index accumulates labeled examples. Because the RAG decoder weights remain fixed, this panel reflects changes induced by the evolving retrieval context rather than parameter forgetting.

G.3 Compute Breakdown

Table 11 provides a stage-wise breakdown of the compute required by each model and adaptation regime. We include only production-relevant costs: initial training, streaming inference, parameter updates, and retrieval-specific computation. Historical re-evaluation used to measure backward transfer is excluded because it is required only for benchmark analysis and would not be incurred in deployment.

Table 11: Stage-wise production compute in FLOPs. Dashes indicate stages that are not applicable to a regime. For Qwen 3.5 4B and 9B, direct FLOP measurements were unreliable; values marked † are estimated by scaling the measured Qwen 3.5 0.8B costs according to the corresponding text-model parameter-count ratio. Backward-transfer evaluation costs are excluded.

		Training		Streaming			
Model	**Regime**	**Initial Train**	**Retraining**	**Inference**	**Embedding**	**Index Build & Retrieval**	**Total**
BERT-Large	Static	1.59×10^{17}	–	1.05×10^{18}	–	–	1.20×10^{18}
ModernBERT-L	Static	1.76×10^{17}	–	1.23×10^{18}	–	–	1.41×10^{18}
Qwen 3.5 0.8B	Static	2.36×10^{17}	–	1.15×10^{18}	–	–	1.38×10^{18}
Qwen 3.5 4B	Static	$1.32\times10^{18\dagger}$	–	$6.43\times10^{18\dagger}$	–	–	$7.75\times10^{18\dagger}$
Qwen 3.5 9B	Static	$2.81\times10^{18\dagger}$	–	$1.37\times10^{19\dagger}$	–	–	$1.65\times10^{19\dagger}$
BERT-Large	Retrain (N=5)	1.59×10^{17}	5.22×10^{17}	1.05×10^{18}	–	–	1.73×10^{18}
ModernBERT-L	Retrain (N=5)	1.76×10^{17}	5.96×10^{17}	1.23×10^{18}	–	–	2.00×10^{18}
Qwen 3.5 0.8B	Retrain (N=5)	2.36×10^{17}	9.66×10^{17}	1.15×10^{18}	–	–	2.35×10^{18}
Qwen 3.5 0.8B	RAG (k=5)	9.73×10^{17}	–	2.44×10^{18}	1.57×10^{18}	1.14×10^{16}	4.99×10^{18}