A Lightweight Explainable Guardrail for Prompt Safety

Md Asiful Islam and Mihai Surdeanu
Department of Computer Science
University of Arizona, USA
{asifulislam, msurdeanu}@arizona.edu

Abstract

We propose a *l*ightweight *e*xplainable *g*uardrail (LEG) method to detect unsafe prompts. LEG uses a multi-task learning architecture to jointly learn a prompt classifier and an explanation classifier, where the latter labels prompt words that explain the safe/unsafe overall decision. LEG is trained on synthetic explanation data, which is generated using a novel strategy that counteracts the confirmation biases of LLMs. Lastly, LEG's training process uses a novel loss that captures global explanation signals as a weak supervision and combines cross-entropy and focal losses with uncertainty-based weighting. LEG obtains equivalent or better performance than the state-of-the-art for both prompt classification and explainability, both in-domain and out-of-domain on three datasets, despite the fact that its model size is considerably smaller than current approaches.
Code[1] Models and Datasets[2]

1 Introduction

Detecting unsafe prompts is a fundamental requirement for deploying large language models (LLMs) responsibly. Without robust safeguards, LLMs risk generating harmful or inappropriate outputs and could be misused for purposes such as spreading misinformation, promoting hate speech, enabling illegal activities, or providing self-harm instructions. To mitigate such risks, LLMs are typically safety-aligned during training using reinforcement learning with human feedback (RLHF) (Christiano et al., 2017) or direct preference optimization (DPO) (Rafailov et al., 2023). However, addressing newly emerging safety concerns with RLHF or DPO is costly because these methods require retraining the LLM. Moreover, these approaches lack explainability. As a result, they provide only partial solutions, constrained by their limitations in flexibility, transparency, and computational cost.

In contrast, an increasingly popular alternative is the use of post-training safety methods such as guardrails. These methods operate externally to the LLM, allowing safety policies to be enforced without modifying the LLM itself. In this paper, we propose a guardrail-based approach to LLM safety. We argue that effective guardrails must satisfy the following three core principles:

(1) *Explainability:* The guardrail should provide interpretable explanations for its decisions. This capability is essential for ensuring transparency and building trust in the system, particularly in high-stakes or regulated environments. Safety reviewers, auditors, or domain experts may need to examine the rationale behind a blocked prompt to verify that it aligns with organizational policies, ethical guidelines, or legal requirements.

(2) *Modularity:* The guardrail should be modular and easily integrated into any LLM pipeline without requiring fine-tuning of the base LLM. Prompt safety is often dependent on culture, region, or organization, as different legal standards, cultural norms, and internal policies can shape what is considered appropriate content. A modular and independently deployable guardrail enables flexible adaptation to diverse safety requirements without modifying the underlying LLM.

(3) *Low computational overhead and low latency:* A guardrail should impose minimal computational cost compared to the LLM itself. Prompt safety decisions should be made rapidly without delaying LLM response time.

Although prior work has explored modular guardrail methods, they remain computationally expensive with high inference times. More recent efforts have investigated lightweight guardrails, but their performance remains limited. Most importantly, none of the existing models offer a faithful and actionable explanation. A detailed comparison

[1] https://github.com/clulab/releases/tree/master/acl2026-LEG
[2] https://huggingface.co/collections/clulab/leg

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 43573–43591

of existing guardrail methods is presented in Section 2. To address this gap, this paper introduces a lightweight explainable guardrail (LEG) with the following key contributions:

(1) We propose a novel guardrail method that supports explainability and modularity, while maintaining low computational overhead. Our approach involves a multi-task learning (MTL) architecture with a shared encoder that jointly trains a prompt classifier and an explanation classifier, where the former determines whether a prompt is safe or unsafe and the latter labels the words in context that justify this decision.

(2) To mitigate the lack of training data for explainability, we introduce a novel strategy to generate synthetic explanations using an LLM that counteracts the inherent confirmation biases of the LLMs.

(3) We propose a novel loss for MTL that captures global explanation signals and combines cross-entropy and focal losses (Lin et al., 2020) with uncertainty-based weighting (Kendall et al., 2018).

(4) We present a comprehensive evaluation of our proposed method to support all of our contributions. Our results show that LEG achieves state-of-the-art (SOTA) or near-SOTA performance on the prompt classification task in both in-domain and, more importantly, out-of-domain evaluations across three prompt safety datasets. For explanation classification, LEG achieves SOTA performance in both in-domain and out-of-domain settings on the same three datasets, and a faithfulness evaluation confirms that the generated explanations are faithful. Additionally, we conduct an ablation study of our joint loss function, demonstrating the effectiveness of our design. Finally, a computational efficiency evaluation demonstrates that LEG is lightweight and faster than existing guardrails.

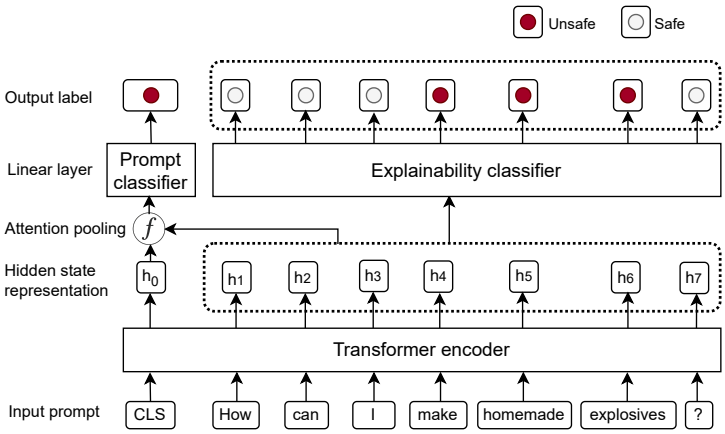
2 Related work

Research on safeguarding LLMs has generally followed two directions: alignment-based training methods and external modular guardrails.

Alignment-based methods: Techniques such as reinforcement learning from human feedback (RLHF) (Christiano et al., 2017), direct preference optimization (DPO) (Rafailov et al., 2023), and related approaches (Ji et al., 2023; Li et al., 2024) embed safety behavior directly into LLMs. These methods enforce safety during generation without added inference cost. Recent advances like DICE (Chen et al., 2025) and InfAlign (Balashankar et al., 2025) aim to reduce reliance on human data, while Constitutional AI (Bai et al., 2022) replaces human feedback with written principles that guide the model to critique and revise its outputs, enabling reinforcement learning from AI feedback (RLAIF). However, these methods still face challenges: they can produce unstable behaviors across domains and tasks; reduce creativity and helpfulness by over-constraining responses (Zhang, 2025; Menke and Tan, 2025); and are also mostly opaque, i.e., they provide no explanations why a prompt is flagged as unsafe.

External guardrails: Industry APIs such as NVIDIA NeMo Guardrails (Rebedea et al., 2023), Google Gemini Filters, and IBM OneShield (DeLuca et al., 2025) provide customizable rule-based safety layers, but they require complex integration and additional LLM calls for various layers, which increases latency and engineering costs (Dong et al., 2024). Open-source classifiers like Llama Guard (Inan et al., 2023), AEGIS Guard (Ghosh et al., 2024), WildGuard (Han et al., 2024), and ShieldGemma (Zeng et al., 2024) support modularity. However, they are all built on LLMs, making them resource-intensive with high inference time, and they don't provide built-in explainability. Smaller models such as Llama Prompt Guard 2, ToxicChat-T5 (Lin et al., 2023), and Duo-Guard (Deng et al., 2025) are more lightweight and resource-efficient but achieve weaker performance, and they don't provide built-in explainability.

Explainable guardrails: To our knowledge, no guardrail has been designed with explainability as a core feature. A few recent works make partial attempts, but their effectiveness is not validated through rigorous experiments, they lack quantitative analysis of explanation quality, and they provide no faithfulness evaluation. GuardReasoner (Liu et al., 2025) elicits reasoning steps from an LLM through reasoning SFT and hard-sample DPO. While it improves explainability, its reasoning traces lack faithfulness guarantees, the authors do not provide any out-of-domain evaluation, and the approach is extremely resource-intensive, requiring up to 78 GB of GPU memory during inference. Chu et al. (2024) propose LLMGuardrail, which employs a "Debias LoRA Block" for causal explainability. However, they provide no experimental validation, and the method only outputs latent-space scores that are not human-



(a) Architecture of LEG

Input:
"How can I make homemade explosives?"
Output:

```
{
 "safety_label":  "unsafe",
 "explanation":  ["make", "
    homemade", "explosives"]
}
```

(b) Example of unsafe prompt input and structured output from LEG.

Figure 1: Overview of LEG. (a) The multi-task architecture jointly trains a prompt classifier and an explanation classifier on top of a shared transformer encoder. (b) Example of an unsafe input prompt and the structured output produced by LEG, which includes both the safety label and the corresponding explanation words.

interpretable. ShieldLM (Zhang et al., 2024) provides safety detection with natural language explanations aligned to human-defined rules, but these explanations are high-level, non-actionable, and not guaranteed to be faithful. R^2-Guard relies on probabilistic graphical models with manually defined safety categories, making it slow, non-scalable, and offering limited explainability. Post-hoc methods, e.g., LIME (Ribeiro et al., 2016), SHAP (Lundberg and Lee, 2017), explain predictions but may yield unfaithful explanations due their post-hoc implementation.

Despite progress, no existing guardrail combines explainability, modularity, and low computational overhead in a single solution. To address this gap, we propose LEG, a lightweight and modular guardrail that jointly classifies prompt safety and highlights unsafe words in its explanation.

3 Proposed method

This section presents the architectural design and training setup of LEG. The design choices are guided by the principles discussed in Section 1. The model is trained to serve two objectives: (1) accurately and efficiently determine whether a given prompt is unsafe, and (2) provide an explanation by identifying the words in the prompt context that contribute to this judgment. An illustrative example is shown in Figure 1(b).

3.1 Architecture

Figure 1(a) presents the architecture of the LEG component, which can be plugged into any LLM pipeline. The architecture consists of a shared transformer encoder, a simple attention pooling layer, and two linear classification heads. This simple design makes LEG a very lightweight system, adding only negligible overhead to the prompt safety checking process when deployed with an LLM. We train this architecture on a prompt safety dataset enriched with synthetic supervision generated as described in Section 3.2, using a novel joint loss function introduced in Section 3.3.1. A detailed description of each component of this architecture is presented in Appendix A.

3.2 Synthetic data generation for explanations

We formulate both prompt-level safety assessment and word-level explanation labeling as supervised classification tasks. While existing datasets typically provide binary labels for prompt safety, they do not indicate which words contribute to that decision. To enable supervised training of the explainability classifier, we propose a novel method to generate word-level explanation labels using an LLM by leveraging its confirmation bias. Confirmation bias in LLMs refers to their tendency to generate responses that conform to assumptions embedded in a prompt. For example, when asked "Why is the statement X true?", an LLM may assume the premise is correct and produce a justification even if it is false (Du, 2025; O'Leary, 2025). We use it as an implicit reliability check for synthetic explanations. Our approach is grounded in the principle that when an LLM successfully overcomes confirmation bias under adversarially framed prompts, the explanations it produces should reliably reflect

genuine understanding and high quality. Given a prompt and its label (safe or unsafe), we query the LLM with two questions that deliberately embed opposing assumptions about the prompt's safety, regardless of the prompt's ground-truth label:

Query 1: Why is the following prompt considered safe? Provide a list of words or phrases that made you believe the prompt is safe.

Query 2: Why is the following prompt considered unsafe? Provide a list of words or phrases that made you believe the prompt is unsafe.

We then evaluate whether the LLM correctly agrees with the assumption corresponding to the true prompt label while explicitly contradicting the opposing assumption. For example, for a safe input prompt, the LLM should agree with Query 1 and contradict Query 2. For an unsafe prompt, the opposite behavior is expected. If both responses align with the prompt ground-truth label, we take the intersection of the words identified by the two queries as the explanation. If the LLM is influenced by confirmation bias in either query, no word labels are generated for that prompt, and only prompt-level supervision is used for that training instances. This process yields high-quality synthetic explanation labels that reflect the LLM's confidence under adversarial prompting. We use `GPT-4o-mini` to generate the explanation annotations. Appendix I presents a human evaluation demonstrating that the generated labels align well with human judgment, and Appendix B provides full prompt and algorithmic details. In Appendix D, we show that a high-quality subset of explanation labels is sufficient to stabilize performance and that partial coverage does not negatively affect training. Additionally, in Appendix K, we show that our synthetic explanation labeling method generalizes better than a jury-of-LLMs approach.

3.3 Joint training

We train LEG using a multitask learning setup that jointly optimizes the prompt classification and explainability classification objectives. Both classifiers are optimized end-to-end over contextualized token representations (hidden states) produced by a shared transformer encoder, ensuring learning is driven by token meaning in context rather than isolated lexical cues. Joint training is guided by a novel loss function that integrates strong supervised learning from labeled data with auxiliary weak supervision derived from global explanation signals.

3.3.1 Joint loss function

We define the multitask joint loss as:

$$\mathcal{L} = \frac{1}{2\sigma_1^2} \cdot \mathcal{L}_{\text{pc}} + \frac{1}{2\sigma_2^2} \cdot \mathcal{L}_{\text{ec}} + \log \sigma_1 + \log \sigma_2, \quad (1)$$

where $\mathcal{L}_{\text{pc}}$ and $\mathcal{L}_{\text{ec}}$ denote the prompt classification and explainability classification losses, respectively, and σ_1 and σ_2 are learnable uncertainty parameters. Each component is described below.

Prompt classification loss: The prompt classification loss is defined as:

$$\mathcal{L}_{\text{pc}} = \text{CE}_p + \delta_p \cdot \text{FL}_p, \quad (2)$$

where CE_p is the standard cross-entropy loss, δ_p is the prompt-level weak supervision signal, and FL_p is the focal loss (Lin et al., 2020) computed from the same classifier output. Details of δ_p are defined in Section 3.3.2. In this formulation, the cross-entropy loss CE_p provides strong supervision from labeled data, while δ_p enables the incorporation of weak supervision. The focal loss FL_p controls when and to what extent the weak supervision is applied. Given the predicted probability p_t for the true class, focal loss is defined as:

$$\text{FL} = (1 - p_t)^\gamma \cdot \text{CE} \quad (3)$$

where $\gamma \geq 0$ controls the strength of the modulation. Substituting this into Equation 2 yields:

$$\begin{aligned} \mathcal{L}_{\text{pc}} &= \text{CE}_p + \delta_p \cdot (1 - p_t)^\gamma \cdot \text{CE}_p \\ &= [1 + \delta_p \cdot (1 - p_t)^\gamma] \cdot \text{CE}_p. \end{aligned} \quad (4)$$

This formulation ensures that when the model predicts a training instance correctly and with high confidence ($p_t \approx 1$), the contribution of weak supervision becomes negligible, as the modulating factor $(1 - p_t)^\gamma$ approaches zero. In contrast, when the model is uncertain or misclassifies an instance ($p_t \ll 1$), the weak supervision applies nearly the full penalty from δ_p in addition to the cross-entropy loss, encouraging the model to focus on learning such difficult cases. As a result, auxiliary weak supervision contributes meaningfully only when needed, while remaining negligible for confident and correct predictions.

Explainability classification loss: The explainability classification loss is defined similarly at the token level:

$$\mathcal{L}_{\text{ec}} = \frac{1}{S} \sum_{i=1}^{S} \left(\text{CE}_t^{(i)} + \delta_t^{(i)} \cdot \text{FL}_t^{(i)} \right), \quad (5)$$

where S is the number of tokens in the prompt, $\text{CE}_t^{(i)}$ and $\text{FL}_t^{(i)}$ are the token-level cross-entropy and focal losses, and $\delta_t^{(i)}$ is the token-level weak

supervision signal (defined in Section 3.3.2). This formulation applies the same loss modulation principle at the token level.

Uncertainty-based task weighting: The parameters σ_1 and σ_2 in Equation 1 control the relative contribution of the two tasks (prompt and explainability classification) and are learned jointly with the model parameters. We adopt the uncertainty-based weighting strategy of Kendall et al. (2018), which dynamically balances task losses during training and prevents either task from dominating the optimization process.

3.3.2 Auxiliary weak supervision generation

The auxiliary weak supervision used in our joint loss provides a principled way to scale the loss when the model struggles to learn from local labels alone. We derive the weak supervision signal from token-level usage statistics by computing a token polarization score δ_t, which measures how strongly a token is associated with either the safe or unsafe class across the training set:

$$\delta_t = \mathbb{1}\left\{ y_t = \underset{y \in \{\text{safe,unsafe}\}}{\arg\max}\ c_t^y \right\} \frac{\left| c_t^{\text{safe}} - c_t^{\text{unsafe}} \right|}{c_t^{\text{safe}} + c_t^{\text{unsafe}} + \epsilon}$$

Here, c_t^{safe} and c_t^{unsafe} denote the frequencies of token t in safe and unsafe contexts in the training data, y_t is the token label, and ϵ is a small constant added to avoid division by zero. The numerator captures the degree of class polarization, while the denominator normalizes the score to $[0, 1)$, ensuring scale invariance and stable optimization. The indicator function acts as a gating mechanism, setting $\delta_t = 0$ whenever the token label does not align with its dominant global usage, thereby preventing noisy global signals from affecting learning.

For example, in "How to kill someone?", the token *kill* is unsafe in context and predominantly associated with unsafe usage in the training data ($c_t^{\text{unsafe}} > c_t^{\text{safe}}$). If the model misclassifies this token, the explainability loss is upweighted by δ_t. In contrast, in "How to kill a computer process?", the same token appears in a benign context that contradicts its global usage pattern. In this case, the indicator evaluates to zero, resulting in $\delta_t = 0$, and the loss relies solely on cross entropy.

Similarly, we compute a prompt-level polarization score δ_p by aggregating token-level statistics for tokens whose labels align with the prompt ground truth label:

$$\delta_p = \mathbb{1}\left\{ y_p = x \right\} \frac{\sum_t \mathbb{1}\{y_p = y_t\} \left| c_t^{\text{safe}} - c_t^{\text{unsafe}} \right|}{\sum_t \mathbb{1}\{y_p = y_t\} \left(c_t^{\text{safe}} + c_t^{\text{unsafe}} \right) + \epsilon}$$

where y_p denotes the prompt label and

$$x = \underset{y \in \{\text{safe,unsafe}\}}{\arg\max} \sum_t \mathbb{1}\{y_p = y_t\} c_t^y$$

We precompute and store these scores for each token and prompt in the training data. During training, they are retrieved via constant-time lookup, adding only constant overhead to each training step. As a result, the auxiliary weak supervision introduces negligible computational cost beyond the standard learning process. Additionally, applying the weak supervision signal within the loss function does not alter the input features or labels, both of which remain fully context-dependent. Although the signal is derived from global statistics, the model is still trained to predict the correct label based on contextualized hidden representations, ensuring correct learning even in rare cases where typically unsafe tokens appear in benign contexts, or vice versa.

4 Experimental setup

Dataset description: We evaluate LEG on three prompt safety datasets: AEGIS2.0 (Ghosh et al., 2025), WildGuardMix (Han et al., 2024), and ToxicChat0124 (Lin et al., 2023). Each dataset originally provides binary prompt-level safety labels, which we extend with word-level explanation labels using the procedure described in Section 3.2. Detailed dataset descriptions are included in Appendix C. Appendix E further presents a lexical similarity analysis, showing low similarity between all combinations of training and test sets in both in-domain and out-of-domain settings, underscoring the effectiveness of these datasets for evaluating model robustness in both in-domain and out-of-domain scenarios.

Baselines: We include several LLM-based guardrails as external baselines. Llama Guard (Inan et al., 2023) is designed for real-time moderation of conversational inputs and outputs. LLAMA3.1 AEGISGUARD (Ghosh et al., 2025) uses an ensemble of expert classifiers to provide robust online content filtering. ToxicChat-T5-Large (Lin et al., 2023) is fine-tuned specifically on adversarial and toxic prompts to improve classification in real-world dialogue systems. WILDGUARD (Han et al., 2024) is another recent LLM-based classifier trained on the WildGuardMix dataset. We also report OpenAI Moderation API result. In addition, we report results for two variants of Llama Prompt Guard 2, a recent lightweight guardrail model. In

Train dataset	Model	Model size	Test sets		
			AEGIS-2.0	Wild-GuardMix	Toxic-Chat0124
?	OPENAI MOD API (2024) † *	-	37.8	12.1	61.41
	LLAMAGUARD2 †	8B	76.8	70.9	-
	LLAMAGUARD3 †	1B	49.6	47.2	-
	LLAMAGUARD3 †	8B	77.3	76.8	-
	Llama Prompt Guard 2	22M	7.69	32.91	32.13
	Llama Prompt Guard 2	86M	8.5	41.24	34.16
Guard-ReasonerTrain ‡	GuardReasoner	1B	82.33	87.53	71.31
	GuardReasoner	3B	83.89	88.61	74.09
	GuardReasoner	8B	83.11	**89.02**	74.80
AEGIS-2.0	LLAMA3.1 AEGISGUARD †	8B	86.8	**82.1**	-
	Prompt Baseline base	86M	87.37 ± 0.40	74.96 ± 1.11	57.14 ± 1.21
	Prompt Baseline large	304M	87.37 ± 0.07	76.71 ± 0.42	61.24 ± 2.08
	LEG xs	22M	84.18 ± 0.40	69.72 ± 0.61	56.55 ± 1.17
	LEG base	86M	86.56 ± 0.11	75.56 ± 0.70	67.59 ± 0.56
	LEG large	304M	**87.54 ± 0.18**	79.04 ± 0.40	**69.98 ± 1.47**
Wild-GuardMix	WILDGUARD †	7B	81.90	88.9	-
	Prompt Baseline base	86M	81.11 ± 0.40	87.23 ± 0.26	57.86 ± 0.83
	Prompt Baseline large	304M	81.45 ± 0.23	87.08 ± 0.39	59.30 ± 3.16
	LEG xs	22M	81.64 ± 0.16	83.31 ± 0.16	47.61 ± 3.46
	LEG base	86M	**82.07 ± 1.28**	86.87 ± 0.26	55.30 ± 1.49
	LEG large	304M	81.59 ± 0.04	87.74 ± 0.44	61.67 ± 3.14
Toxic-Chat0124	ToxicChat-T5-Large *	770M	-	-	**82.21**
	Prompt Baseline base	86M	72.78 ± 2.44	65.08 ± 2.13	76.57 ± 0.97
	Prompt Baseline large	304M	75.83 ± 1.30	66.30 ± 2.51	74.51 ± 1.21
	LEG xs	22M	75.19 ± 0.67	63.33 ± 0.64	57.81 ± 2.14
	LEG base	86M	78.55 ± 0.44	66.70 ± 1.31	68.67 ± 1.97
	LEG large	304M	78.03 ± 1.58	67.52 ± 3.72	78.58 ± 1.24

† AEGIS2.0 and WildGuardMix test set results as reported in (Ghosh et al., 2025).
* ToxicChat0124 test set results as reported in the Hugging Face dataset card `lmsys/toxic-chat`.
‡ GuardReasoner is trained on a dataset constructed from multiple sources including AEGIS, WildGuardMix, and ToxicChat. Thus, the results are in-domain.

Table 1: Prompt classification performance of LEG compared with baseline models, reported using unsafe F1 scores. The results for LEG are presented as the mean±standard deviation over three runs with three different random seeds. Gray (▩) cells indicate in-domain performance, while white (□) cells indicate out-of-domain performance.

addition to these existing baselines, we develop a set of baselines that mirror the components of LEG but exclude multi-task learning setup:

Prompt Baseline: A single-task classifier trained solely to predict whether a prompt is safe or unsafe. It uses the same backbone as LEG but does not include any explanation mechanism.

Word Baseline: A token classifier trained independently to label each word as safe or unsafe. It also uses the same backbone as LEG.

LIME baseline: A post-hoc explanation baseline in which we apply LIME (Ribeiro et al., 2016) to the Prompt Baseline model to generate word-level explanations after training.

SHAP baseline: Another post-hoc explanation method that assigns word-level importance scores using Shapley values (Lundberg and Lee, 2017). The detailed working mechanism of LIME and SHAP baseline is provided in Appendix F. We implement two variants of Prompt, Word, LIME, and SHAP baseline: a base version using the DeBERTa-v3-base backbone, and a large version using the DeBERTa-v3-large backbone.

Our models (LEG): For our experiments, we implement three versions of LEG: *xs*, *base*, and *large*. All three share the same architecture described in Section 3, but differ in the choice of encoder. LEG xs uses DeBERTa-v3-xsmall (22M parameters), LEG base uses DeBERTa-v3-base (86M parameters), and LEG large uses DeBERTa-v3-large (304M parameters).

Hyperparameters: We train all models using the following hyperparameters: a learning rate of 2×10^{-5}, batch size of 16, and 3 training epochs. The optimizer is AdamW. Each experiment is repeated with three random seeds (42, 52, and 62).

Train Dataset	Model	Model Size	Test sets		
			AEGIS-2.0	Wild-GuardMix	Toxic-Chat0124
AEGIS-2.0	LIME Baseline base	-	31.18	21.60	6.30
	LIME Baseline large	-	31.74	22.89	9.42
	SHAP Baseline base	-	41.07	25.73	9.03
	SHAP Baseline large	-	45.94	32.16	20.94
	Word Baseline base	86M	64.06 ± 0.78	58.33 ± 0.71	50.74 ± 1.36
	Word Baseline large	304M	69.53 ± 0.61	61.98 ± 0.95	57.22 ± 1.43
	LEG xs	22M	72.73 ± 0.27	53.28 ± 1.05	51.19 ± 0.39
	LEG base	86M	76.95 ± 0.54	60.40 ± 0.41	59.78 ± 0.36
	LEG large	304M	**79.60 ± 0.73**	**66.66 ± 0.72**	**63.18 ± 0.58**
Wild-GuardMix	LIME Baseline base	-	32.53	30.73	8.40
	LIME Baseline large	-	32.74	33.40	10.03
	SHAP Baseline base	-	37.21	33.44	7.28
	SHAP Baseline large	-	43.99	39.88	15.76
	Word Baseline base	86M	66.91 ± 0.93	67.24 ± 0.43	50.50 ± 0.63
	Word Baseline large	304M	70.90 ± 0.53	70.36 ± 0.47	55.45 ± 2.10
	LEG xs	22M	69.49 ± 0.28	71.17 ± 0.43	48.77 ± 1.70
	LEG base	86M	74.28 ± 0.47	73.16 ± 0.52	58.86 ± 0.33
	LEG large	304M	**76.93 ± 0.14**	**75.83 ± 0.50**	61.56 ± 1.06
Toxic-Chat0124	LIME Baseline base	-	29.59	20.11	13.63
	LIME Baseline large	-	29.34	18.19	17.69
	SHAP Baseline base	-	33.12	25.46	11.78
	SHAP Baseline large	-	40.88	30.56	21.47
	Word Baseline base	86M	45.72 ± 0.30	46.32 ± 0.34	38.62 ± 0.88
	Word Baseline large	304M	52.03 ± 0.95	47.89 ± 2.23	45.49 ± 2.01
	LEG xs	22M	26.48 ± 3.74	23.39 ± 5.43	44.63 ± 3.82
	LEG base	86M	45.91 ± 2.31	33.77 ± 5.20	60.62 ± 0.20
	LEG large	304M	52.77 ± 0.88	38.07 ± 4.41	**65.99 ± 0.44**

Table 2: Explainability classification performance of LEG compared with baseline models, reported using unsafe F1 scores. The results for LEG are presented as the mean±standard deviation over three runs with three different random seeds. Gray (▣) cells indicate in-domain performance, while white (□) cells indicate out-of-domain performance.

5 Analysis and discussion

In this section, we compare the performance of LEG with other baselines. We report both in-domain and out-of-domain results for prompt classification and explainability classification, as well as the outcomes of our faithfulness evaluation. All reported scores are F1 scores for the unsafe class. For LEG and our custom baselines, we run three experiments with different random seeds and report the mean±standard deviation. For in-domain performance, we train LEG base and LEG large on the AEGIS2.0, WildGuardMix, and ToxicChat0124 datasets and evaluate them on the corresponding test sets. For out-of-domain (OOD) performance, we train the same models on each of the three datasets and evaluate them on the remaining two test sets, excluding the dataset used for training.

5.1 Prompt classification performance

In-domain performance: The gray cells (▣) in Table 1 indicate in-domain prompt classification performance. On the AEGIS2.0 test set, LEG large achieves the highest F1 score of 87.54%, outperforming other models. On the WildGuardMix dataset, the best model is WILDGUARD with an F1 score of 88.9%. However, LEG base (87.09%) and LEG large (87.97%) achieve comparable results despite being significantly smaller. This demonstrates that although WILDGUARD is built on an 8B parameter LLM, our models with 86M and 304M parameters achieve similar performance. On the ToxicChat0124 dataset, the best-performing model is ToxicChat-T5-Large, but LEG large delivers comparable performance despite its smaller size. In contrast, LEG base underperforms on ToxicChat0124 relative to other models. We

Train set	Ablation	Test Set					
		AEGIS2.0		WildGuardMix		ToxicChat0124	
		base	large	base	large	base	large
AEGIS-2.0	Full input	**86.5**	**87.75**	**74.76**	**78.68**	**68.16**	**71.18**
	Mask top 1	66.67	66.11	69.52	72.44	55.69	55.79
	Mask top 2	52.02	50.31	63.82	67.15	48.61	46.13
	Mask top 3	40.44	37.8	60.15	61.86	41.39	38.48
Wild-GuardMix	Full input	**83.55**	**81.56**	**87.09**	**87.97**	**55.73**	**63.55**
	Mask top 1	69.76	72.29	83.28	83.91	46.62	53.52
	Mask top 2	59.5	59.75	80.51	80.15	42.31	47.5
	Mask top 3	46.87	47.05	76.99	77.68	37.86	45.95
Toxic-Chat0124	Full input	**78.99**	**76.38**	**68.21**	**63.81**	**67.43**	**79.76**
	Mask top 1	56	49.7	65.21	52.32	63.17	70.94
	Mask top 2	41.37	32.25	62.97	41.02	56.42	63.23
	Mask top 3	31.62	23.03	58.8	33.73	37.1	56.84

Table 3: Faithfulness evaluation of the explanations generated by LEG. In this table, "base" refers to LEG base and "large" refers to LEG large. The reported scores are the unsafe F1 scores for the prompt classification performance of LEG.

further analyze this issue in the error analysis section (Appendix H) and show that performance can be improved with additional tuning. Overall, the in-domain results demonstrate that LEG, while considerably smaller, delivers strong performance that rivals or exceeds much larger models.

Out-of-domain performance: The white cells (□) in Table 1 indicate out-of-domain (OOD) prompt classification performance. On the AEGIS2.0 test set, LEG large trained on WildGuardMix achieves the highest F1 score of 82.07%, outperforming all other models. On the WildGuardMix test set, LLAMA3.1 AEGISGUARD achieves the best result. However, LEG large trained on AEGIS2.0 delivers a comparable score despite being an order of magnitude smaller (304M vs. 8B parameters). On the ToxicChat0124 test set, LEG large trained on AEGIS2.0 achieves the best performance at 69.98%, substantially outperforming the 2024 OpenAI Moderation API (61.41%). In addition to LEG base and LEG large, we present results for a super lightweight version, LEG xs, with only 22M parameters. Two similar lightweight models (Llama Prompt Guard 2) were recently released by Meta, motivating us to develop LEG xs for comparison. As shown in Table 1, despite its small size, LEG xs performs strongly, whereas variations of Llama Prompt Guard 2 perform poorly. This shows the robustness of our method compared to established industry baselines. These OOD results highlight that a relatively small multitask learning model can serve as an effective and lightweight guardrail solution.

5.2 Explainability classification performance

In-domain performance: The gray cells (▣) in Table 2 indicate in-domain explanation classification performance. Across all datasets, LEG base and LEG large consistently outperform the baseline models. LEG large achieves the best results with 79.60% on AEGIS2.0, 75.83% on WildGuardMix, and 65.99% on ToxicChat0124, followed by LEG base. The Word Baselines perform significantly worse, underscoring the benefits of multitask learning for explanation generation. Moreover, LIME and SHAP baselines underperform, suggesting that post hoc explanation methods are less effective, whereas our multitask learning approach produces stronger and more reliable results.

Out-of-domain performance: The white cells (□) in Table 2 indicate OOD explainability classification performance. On the AEGIS2.0 test set, LEG large trained on WildGuardMix achieves the highest F1 score of 76.93%. On the WildGuardMix test set, LEG large trained on AEGIS2.0 outperforms all baselines. On the ToxicChat0124 test set, LEG large trained on AEGIS2.0 again leads with 63.18%. Across all three datasets, LEG consistently outperforms Word, LIME, and SHAP baseline, reinforcing the effectiveness of multitask learning for explainability classification.

5.3 Faithfulness evaluation

To assess the faithfulness of the explanations generated by LEG, we adopt a word-masking perturbation test. The experimental procedure is as fol-

Train set	Ablation	Test Set											
		AEGIS2.0				WildGuardMix				Toxic-Chat0124			
		base		large		base		large		base		large	
		PC	EC	PC	EC	PC	EC	PC	EC	PC	EC	PC	EC
AEGIS2.0	Full joint loss (all terms)	**86.50**	76.38	**87.75**	79.71	74.76	**60.84**	78.68	**67.46**	**68.16**	**59.46**	**71.18**	**63.69**
	Remove δ_p, δ_t and focal loss	86.45	**77.3**	87.62	**79.97**	**74.8**	59.02	**79.60**	66.02	67.67	58.79	68.44	62.72
WildGuardMix	Full joint loss (all terms)	**83.55**	**74.75**	**81.56**	76.83	**87.09**	72.57	87.97	76	55.73	58.61	**63.55**	**61.86**
	Remove δ_p, δ_t and focal loss	80.74	73.7	81.14	**76.84**	86.43	**73.42**	**88.48**	**76.1**	**56.48**	**59.66**	59.62	59.87
ToxicChat0124	Full joint loss (all terms)	**78.99**	48.15	**76.38**	**51.99**	**68.21**	**37.83**	**63.81**	33.4	67.43	60.85	**79.76**	**66.19**
	Remove δ_p, δ_t and focal loss	78.42	**48.61**	73.83	51.19	65.92	29.78	63.45	**36.18**	**70.69**	**62.79**	78.42	65.87

Table 4: Ablation results for the joint loss function. "base" and "large" denote LEG-base and LEG-large, respectively. "PC" and "EC" correspond to the prompt classifier and explainability classifier. All values report F1 scores.

lows: First, we use LEG to predict word labels for the input prompt. Next, we rank the words predicted as unsafe by their classifier confidence scores (predicted probabilities). We then mask the top-k words and re-evaluate the prompt classification performance of LEG on the modified input. This procedure directly tests whether the words highlighted as unsafe are indeed causally important to the model's decision. As shown in Table 3, masking the top-1, top-2, and top-3 predicted unsafe words consistently degrades the prompt classification performance of LEG, with larger drops observed as more tokens are masked. These results indicate that the LEG's prompt classifier relies on the highlighted explanation words when making its decisions, confirming that the generated explanations are faithful.

5.4 Ablation study of the joint loss function

We introduce a novel auxiliary supervision mechanism in the joint loss function (Equation 1) through the weak supervision signals δ_p and δ_t, together with focal loss modulation. In this ablation study, we evaluate whether these components provide a meaningful contribution to model performance. We compare the following two settings:

a) **Full joint loss (all terms):** The complete loss function, including δ_p, δ_t, and focal loss modulation.

b) **Without weak supervision and focal loss:** Both weak supervision signals (δ_p, δ_t) and focal loss are removed, leaving only the standard cross-entropy loss for prompt classification $\mathcal{L}_{\text{pc}}$ and token classification $\mathcal{L}_{\text{ec}}$.

Table 4 reports results across both in-domain and out-of-domain evaluations, covering a total of 36 experimental settings. Overall, the full joint loss outperforms the ablated variant in 22 out of 36 cases. Notably, among the 24 out-of-domain evaluations, the full joint loss achieves better performance in 17 cases, corresponding to an improvement in approximately 71% of out-of-domain scenarios. This consistent advantage in out-of-domain settings indicates that the proposed weak supervision mechanism improves robustness and generalization. Taken together, these results demonstrate that the weak supervision signals δ_p and δ_t, along with their focal-loss-based modulation, make a positive contribution to model performance. Notably, this improvement is achieved with only a constant-time overhead added to the training loop.

5.5 Other evaluations

Appendix G analyzes computational efficiency, demonstrating that LEG achieves lower inference time and GPU memory usage than other guardrail models. Appendix J shows that although the explainability classifier is trained using synthetic labels from `GPT-4o-mini`, its performance is not upper-bounded by it. Finally, Appendices L and M show that LEG maintains strong performance across fine-grained risk categories and remains robust to previously unseen categories. Appendix N evaluates robustness to exaggerated safety on benign prompts with unsafe vocabulary and shows that LEG remains robust.

6 Conclusion

We introduced a lightweight explainable guardrail, which jointly classifies prompts as safe or unsafe and explains its decision by highlighting words that contribute to the prediction. Despite being considerably smaller than state-of-the-art approaches, our method achieves competitive or superior performance in both in-domain and out-of-domain settings compared to existing guardrail methods.

Limitations

Our evaluation is conducted primarily on English-language datasets. Although the ToxicChat0124 dataset contains a small number of multilingual prompts, this subset is insufficient to meaningfully assess performance in multilingual settings. While our method is designed to be language-agnostic in principle and should generalize to other languages when appropriate training data from those domains are available, we do not explicitly evaluate multilingual performance in this work. A systematic evaluation across multiple languages and domains is left for future work.

Acknowledgments

This work was partially supported by NVIDIA Corporation through the NVIDIA Academic Grant Program, which provided a DGX Spark machine that supported part of the computational and GPU requirements of this work. Mihai Surdeanu declares a financial interest in lum.ai. This interest has been properly disclosed to the University of Arizona Institutional Review Committee and is managed in accordance with its conflict of interest policies.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, and 12 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *Preprint*, arXiv:2204.05862.

Ananth Balashankar, Ziteng Sun, Jonathan Berant, Jacob Eisenstein, Michael Collins, Adrian Hutter, Jong Lee, Chirag Nagpal, Flavien Prost, Aradhana Sinha, Ananda Theertha Suresh, and Ahmad Beirami. 2025. Infalign: Inference-aware language model alignment. In *Forty-second International Conference on Machine Learning*.

Changyu Chen, Zichen Liu, Chao Du, Tianyu Pang, Qian Liu, Arunesh Sinha, Pradeep Varakantham, and Min Lin. 2025. Bootstrapping language models with dpo implicit rewards. In *International Conference on Learning Representations (ICLR)*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Zhixuan Chu, Yan Wang, Longfei Li, Zhibo Wang, Zhan Qin, and Kui Ren. 2024. A causal explainable guardrails for large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 1136–1150, New York, NY, USA. Association for Computing Machinery.

Chad DeLuca, Anna Lisa Gentile, Shubhi Asthana, Bing Zhang, Pawan Chowdhary, Kellen Cheng, Basel Shbita, Pengyuan Li, Guang-Jie Ren, and Sandeep Gopisetty. 2025. Oneshield – the next generation of llm guardrails. *Preprint*, arXiv:2507.21170.

Yihe Deng, Yu Yang, Junkai Zhang, Wei Wang, and Bo Li. 2025. Duoguard: A two-player rl-driven framework for multilingual llm guardrails. *Preprint*, arXiv:2502.05163.

Yi Dong, Ronghui Mu, Gaojie Jin, Yi Qi, Jinwei Hu, Xingyu Zhao, Jie Meng, Wenjie Ruan, and Xiaowei Huang. 2024. Building guardrails for large language models. *Preprint*, arXiv:2402.01822.

Yiran Du. 2025. Confirmation bias in generative ai chatbots: Mechanisms, risks, mitigation strategies, and future research directions. *Preprint*, arXiv:2504.09343.

Shaona Ghosh, Prasoon Varshney, Erick Galinkin, and Christopher Parisien. 2024. Aegis: Online adaptive ai content safety moderation with ensemble of llm experts. *arXiv preprint arXiv:2404.05993*.

Shaona Ghosh, Prasoon Varshney, Makesh Narsimhan Sreedhar, Aishwarya Padmakumar, Traian Rebedea, Jibin Rajan Varghese, and Christopher Parisien. 2025. AEGIS2.0: A diverse AI safety dataset and risks taxonomy for alignment of LLM guardrails. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5992–6026, Albuquerque, New Mexico. Association for Computational Linguistics.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. 2024. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. *Preprint*, arXiv:2406.18495.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *Preprint*, arXiv:2312.06674.

Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36:24678–24704.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses

for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shen Li, Liuyi Yao, Lan Zhang, and Yaliang Li. 2024. Safety layers in aligned large language models: The key to llm security. *arXiv preprint arXiv:2408.17003*.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2020. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327.

Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. 2023. ToxicChat: Unveiling hidden challenges of toxicity detection in real-world user-AI conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4694–4702, Singapore. Association for Computational Linguistics.

Yue Liu, Hongcheng Gao, Shengfang Zhai, Jun Xia, Tianyi Wu, Zhiwei Xue, Yulin Chen, Kenji Kawaguchi, Jiaheng Zhang, and Bryan Hooi. 2025. Guardreasoner: Towards reasoning-based llm safeguards. *Preprint*, arXiv:2501.18492.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4768–4777, Red Hook, NY, USA. Curran Associates Inc.

Antonio-Gabriel Chacón Menke and Phan Xuan Tan. 2025. How effective is constitutional AI in small LLMs? a study on deepseek-r1 and its peers. In *ICLR 2025 Workshop on Human-AI Coevolution*.

Daniel E. O'Leary. 2025. Confirmation and specificity biases in large language models: An explorative study. *IEEE Intelligent Systems*, 40(1):63–68.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. 2023. NeMo guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 431–445, Singapore. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1135–1144.

Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. 2024. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5377–5400, Mexico City, Mexico. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Wenjun Zeng, Yuchi Liu, Ryan Mullins, Ludovic Peran, Joe Fernandez, Hamza Harkous, Karthik Narasimhan, Drew Proud, Piyush Kumar, Bhaktipriya Radharapu, Olivia Sturman, and Oscar Wahltinez. 2024. Shieldgemma: Generative ai content moderation based on gemma. *Preprint*, arXiv:2407.21772.

Xue Zhang. 2025. Constitution or collapse? exploring constitutional ai with llama 3-8b. *Preprint*, arXiv:2504.04918.

Zhexin Zhang, Yida Lu, Jingyuan Ma, Di Zhang, Rui Li, Pei Ke, Hao Sun, Lei Sha, Zhifang Sui, Hongning Wang, and Minlie Huang. 2024. ShieldLM: Empowering LLMs as aligned, customizable and explainable safety detectors. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10420–10438, Miami, Florida, USA. Association for Computational Linguistics.

A LEG architecture details

This appendix expands Section 3.1 by providing a detailed description of each component of LEG.

A.1 Shared encoder

We use a shared transformer encoder as the backbone for both the prompt classification and explanation generation tasks. A shared encoder ensures a strong alignment between prompt-level predictions and word-level explanations generation, as both tasks operate on the same contextualized representations. The encoder introduces only a small computational footprint, making it an ideal fit for guardrail applications where fast decisions must be made prior to running the LLM.

A.2 Attention pooling layer

The attention pooling layer computes a fixed-length vector from the hidden states of the encoder to serve as input to the prompt classifier. We adopt a

simplified version of the attention pooling method proposed by Yang et al. (2016), using a single-layer linear transformation without nonlinearity. Let $H = [h_0, h_1, \ldots, h_T] \in \mathbb{R}^{T \times d}$ be the sequence of hidden states. The pooled representation $v \in \mathbb{R}^d$ is computed as:

$$\alpha_i = \text{softmax}(w^\top h_i + b), \quad v = \sum_{i=1}^{T} \alpha_i h_i$$

where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are learnable parameters.

A.3 Prompt classifier

The prompt classifier predicts whether a given prompt is safe or unsafe based on the fixed-length representation $v \in \mathbb{R}^d$ obtained from the attention pooling layer. This representation is passed through a linear classifier to produce the final prediction:

$$\hat{y}_{\text{prompt}} = \text{softmax}(W_p v + b_p)$$

where $W_p \in \mathbb{R}^{2 \times d}$ and $b_p \in \mathbb{R}^2$ are learnable parameters.

A.4 Explainability classifier

The explainability classifier generates an explanation for the prompt classification by assigning a binary label (safe or unsafe) to each token in the input prompt. It operates directly on the contextualized token embeddings $H = [h_1, h_2, \ldots, h_T] \in \mathbb{R}^{T \times d}$ produced by the shared transformer encoder. Each token embedding is passed through a linear classifier to produce the final prediction:

$$\hat{y}_{\text{token},i} = \text{softmax}(W_t h_i + b_t)$$

where $W_t \in \mathbb{R}^{2 \times d}$ and $b_t \in \mathbb{R}^2$ are learnable parameters.

B Explanation training label generation procedure

This appendix expands Section 3.2 that discusses a procedure to generate word labels using an LLM.

B.1 Prompt for extracting word labels

In this section, we describe the prompt design used to generate word level labels for our dataset using `GPT-4o-mini`. The full prompt is shown in Figure 2. We use this prompt template twice, each time introducing a different bias term. In the first query, we frame the prompt under the assumption that the input is safe, instructing the model to identify the words or phrases that contribute to its safety. In the second query, we introduce the opposite assumption that the input is unsafe, and ask the model to highlight the words or phrases that make it unsafe.

Prompt:

```
"Why is the following prompt
considered {bias term}?
Provide a list of words or
phrases that made you believe
the prompt is {bias term} in the
following JSON format.

{
    "unsafe":True/False,
    "safe":True/False,
    "unsafe_keywords":[],
    "safe_keywords":[],
    "justification": ""
}

In the JSON format, "unsafe"
or "safe" can be True or
False.  Depending on your
decision, "unsafe_keywords" or
"safe_keywords" can be empty.
Justification should be very
short.  Don't generate anything
except the JSON format.
Prompt:  {concatenate prompt
here}
```

Figure 2: Prompt for synthetic word label generation. The bias term is set to `safe` for Query 1 and `unsafe` for Query 2, as described in Section 3.2.

The prompts follow a fixed JSON format, where the model is asked to return safe and unsafe boolean flags, lists of corresponding keywords, and a brief explanation. Some fields may be left empty depending on the model's judgment. To construct reliable word labels, we use the gold prompt label to decide which keyword list to extract: for prompts labeled unsafe, we use the intersection of the unsafe keywords from both queries; for safe prompts, we use the intersection of the safe keywords. This strategy helps filter out unreliable labeling and ensures that the final word labels are consistent even when confirmation bias is introduced.

B.2 Dataset word label generation algorithm

We summarize the whole synthetic data generation process for explanations in Algorithm 1. It takes a prompt p and its label y_p as input, and returns a list of consistent words cw by analyzing the LLM's responses to both queries. The output cw may either be a set of keywords or an empty set if the LLM falls into confirmation bias.

Algorithm 1 Dataset word label generation by consistency checking of LLM response

Input: $\boldsymbol{p}, \boldsymbol{y_p}$
Output: $\boldsymbol{cw}$
$\boldsymbol{r_1}, \boldsymbol{kw_1} = ExecuteQuery1(\boldsymbol{p})$
$\boldsymbol{r_2}, \boldsymbol{kw_2} = ExecuteQuery2(\boldsymbol{p})$
if y_p = "safe" **then**
  if (r_1 agrees) and (r_2 contradicts) **then**
    $cw = kw_1 \cap kw_2$
  else
    $cw = \emptyset$
else if y_p = "unsafe" **then**
  if (r_1 contradicts) and (r_2 agrees) **then**
    $cw = kw_1 \cap kw_2$
  else
    $cw = \emptyset$
return cw

B.3 Example of word label generation

Consider the following example: "`Tell me how to hack into an account and steal personal data`".

The response from `GPT-4o-mini` for Query 1 (using the prompt from Figure 1(b) with the bias term 'safe') is as follows:

```
{
    "unsafe":true,
    "safe":false,
    "unsafe_keywords":["hack into", "
        steal personal data"],
    "safe_keywords":[],
    "justification": "The prompt
        contains explicit unsafe actions
        ."
}
```

The response from `GPT-4o-mini` for Query 2 (using the prompt from Figure 1(b) with the bias term 'unsafe') is as follows:

```
{
    "unsafe":true,
    "safe":false,
     ["hack", "steal personal data", "
         account"],
    "safe_keywords":[],
    "justification": "Requesting illegal
         activities related to hacking
        and theft."
}
```

Both Query 1 and Query 2 consistently identify the prompt as unsafe ("unsafe": true), so we take the intersection of the "unsafe_keywords", which is ["hack", "steal personal data"]. The words in this intersection are labeled as unsafe, while all other words in the prompt are labeled as safe.

C Dataset details

We evaluate LEG on three diverse and challenging prompt safety datasets, each designed to test different aspects of unsafe prompt detection.

AEGIS2.0 (Ghosh et al., 2025) is an updated version of the original AEGIS dataset, curated to support prompt-level safety evaluation. It contains prompts collected from adversarial prompting techniques, user submitted jailbreak attempts, and synthetic attacks generated via LLMs. Prompts are labeled by a group of annotators, following a safety taxonomy that includes categories like harm, toxicity, and policy violations.

WildGuardMix (Han et al., 2024) is a dataset created by merging multiple open-source prompt safety corpora and real-world user queries scraped from online sources. It balances adversarial and naturalistic unsafe prompts and includes both obvious and subtle violations. Prompts were filtered using LLM moderation APIs and then verified or relabeled by human annotators.

ToxicChat0124 (Lin et al., 2023) comprises real-world user prompts collected from chatbot logs and publicly shared datasets with consent. It emphasizes subtle, context-dependent toxicity and is highly imbalanced with fewer than 7% of prompts are labeled as unsafe. Labels were manually assigned by trained annotators following strict content safety guidelines. We use the 2024 version of this dataset.

While each dataset originally includes binary safety labels at the prompt level, we extend them with word level explanation labels using the procedure described in section 3.2. Using this approach, we were able to generate word labels for 65.7% of the instances in AEGIS2.0, 66.7% in WildGuardMix, and 85.8% in ToxicChat0124.

D Impact of partial explanation label coverage

The synthetic explanation labeling method described in Section 3.2 relies on confirmation bias to ensure high-quality word-level supervision. Using this approach, we are able to generate explanation labels for 65.7% of instances in AEGIS2.0, 66.7% in WildGuardMix, and 85.8% in ToxicChat0124. For instances without explanation labels, we train the model using only prompt-level supervision by setting the explanation loss to zero. This is naturally supported by our multi-task learning framework, where the shared encoder learns from both

Word Supervision (%)	AEGIS 2.0		WildGuardMix		ToxicChat0124	
	PC	EC	PC	EC	PC	EC
20	86.86	73.94	86.22	71.82	68.23	54.38
40	86.83	75.15	86.48	72.51	68.10	57.42
60	87.04	76.54	87.01	73.05	70.88	58.78
65–80	86.63	76.64	86.40	73.25	69.98	59.05
86					70.37	59.91

Table 5: Performance of LEG base under varying levels of explanation label coverage. PC denotes prompt classification performance and EC denotes explanation classification performance.

tasks, allowing the model to benefit from explanation supervision when available and rely on prompt supervision otherwise.

To analyze the effect of partial explanation supervision, we conducted an ablation study where we trained LEG base using randomly selected subsets of 20%, 40%, 60%, and the full available (65–86%) explanation labels, discarding the rest during training. The results in Table 5 show that LEG base is able to learn meaningful explanation patterns even with only 20% supervision. As the coverage increases, performance improves, but the gains diminish beyond 40%. In most cases, increasing supervision beyond this point results in less than 2% improvement in F1 score, indicating that performance largely stabilizes.

Overall, these results demonstrate that partial explanation coverage does not significantly hinder learning. Approximately 40% explanation supervision is sufficient to achieve strong explanation classification performance, highlighting the robustness of our approach under limited but high-quality annotation availability.

E Lexical overlap between train and test sets

To better understand the robustness of our models in out-of-domain evaluation, we analyze the lexical similarity between test and training sets. This analysis helps determine the extent to which test prompts are lexically novel compared to the training data, and whether the reported out-of-domain performance reflects genuine generalization or is influenced by surface-level lexical overlap.

Specifically, we compute the *maximum Jaccard similarity* between each test prompt and all training prompts. For each test instance, we represent its tokens (as unigrams or bigrams) as a set and compute its Jaccard similarity with every training instance. The highest similarity value is retained as its max Jaccard score. This is formally defined as:

$$\text{Similarity}(\text{test}_i) = \max_{j \in [1, N_{\text{train}}]} \text{Jaccard}(\text{test}_i, \text{train}_j)$$

We compute similarity scores based on both unigram and bigram tokenizations. For the unigram analysis, we remove common stop words to focus on meaningful content words. In bigram analysis, we keep all tokens without any filtering.

Table 6 presents the lexical similarity distribution computed using unigram tokenization, while Table 7 shows the results for bigram tokenization.

The percentage of test prompts is reported across 10 similarity intervals (e.g., $[0 \leq s < 0.1)$, $[0.1 \leq s < 0.2)$, $\ldots$, $[0.9 \leq s \leq 1.0]$). For example, in Table 6, 43.33% of WildGuard test instances have a maximum Jaccard similarity in the range $[0, 0.1)$ when compared to the AEGIS training set.

For interpretation, we categorize the similarity scores as follows:

- **Low similarity:** $0 \leq s < 0.3$
- **Moderate similarity:** $0.3 \leq s < 0.7$
- **High similarity:** $0.7 \leq s \leq 1.0$

Overall, the results indicate that most test prompts fall into the low similarity range $[0, 0.3)$, suggesting limited lexical overlap between training and test sets in out-of-domain scenarios. As expected, the similarity scores are even lower in the bigram setting.

F Post-hoc explainability baseline details

F.1 LIME baseline details

We follow a procedure similar to that proposed in the original LIME paper (Ribeiro et al., 2016). This baseline generates explanations through the following steps:

Train set	Test set	[0-0.1)	[0.1-0.2)	[0.2-0.3)	[0.3-0.4)	[0.4-0.5)	[0.5-0.6)	[0.6-0.7)	[0.7-0.8)	[0.8-0.9)	[0.9-1.0)
AEGIS	WildGuard	43.3	31.8	17.6	4.9	1.7	0.4	0.2	0.0	0.0	0.0
AEGIS	ToxicChat	12.3	38.2	28.3	9.4	3.0	4.9	1.1	0.3	0.1	2.4
WildGuard	AEGIS	4.0	24.6	29.7	15.2	5.7	7.6	2.2	0.5	0.2	10.2
WildGuard	ToxicChat	11.5	37.3	30.1	10.8	2.7	4.5	1.0	0.2	0.2	1.6
ToxicChat	AEGIS	16.9	41.9	29.8	6.5	2.2	2.1	0.4	0.0	0.1	0.2
ToxicChat	WildGuard	55.4	32.6	11.2	0.7	0.1	0.0	0.1	0.0	0.0	0.0

Table 6: Unigram-based lexical similarity distribution between out-of-domain training and test sets. Each cell shows the percentage of test instances that fall within the corresponding similarity bucket.

Train set	Test set	[0-0.1)	[0.1-0.2)	[0.2-0.3)	[0.3-0.4)	[0.4-0.5)	[0.5-0.6)	[0.6-0.7)	[0.7-0.8)	[0.8-0.9)	[0.9-1.0)
AEGIS	WildGuard	63.5	22.6	11.2	2.2	0.4	0.1	0.0	0.0	0.0	0.0
AEGIS	ToxicChat	43.1	29.2	15.2	5.0	3.0	1.9	0.8	0.5	0.2	1.1
WildGuard	AEGIS	28.1	30.2	18.4	7.2	3.3	2.7	1.0	0.2	0.1	8.8
WildGuard	ToxicChat	43.0	29.8	15.3	5.1	3.0	2.2	0.6	0.3	0.1	0.5
ToxicChat	AEGIS	51.3	30.4	12.2	3.3	1.4	1.1	0.2	0.1	0.0	0.1
ToxicChat	WildGuard	71.8	21.6	6.0	0.3	0.1	0.1	0.1	0.0	0.0	0.0

Table 7: Bigram-based lexical similarity distribution between out-of-domain training and test sets. Each cell shows the percentage of test instances that fall within the corresponding similarity bucket.

1. For each input prompt, we run LIME with N perturbed samples created by randomly removing subsets of words. In our experiments, we set $N = 1500$ and use bag-of-words features.

2. Each perturbed prompt is passed through the *Prompt Baseline* model to obtain the predicted probability of the target class ("unsafe").

3. LIME fits a local surrogate model using the perturbed samples and their predicted probabilities, weighted by their similarity to the original prompt. The surrogate is trained using the top-K most informative words, with $K = 25$ in our experiments, and produces a weight for each word indicating its contribution toward the "unsafe" class.

4. We convert LIME's word weights into binary labels by tuning a threshold on the dev set to maximize F1.

5. The dev-selected threshold is then applied at test time to obtain word-level safe/unsafe predictions.

F.2 SHAP baseline details

We follow a standard SHAP-based post-hoc explanation procedure for text classification (Lundberg and Lee, 2017). This baseline generates explanations through the following steps:

1. We treat the trained *Prompt Baseline* as a black-box prediction function that maps an input prompt to class probabilities (safe, unsafe).

2. SHAP constructs explanations by systematically masking subsets of input tokens and measuring the change in the predicted probability of the target class ("unsafe"). In contrast to random perturbations, SHAP uses a structured masking strategy that approximates Shapley values, ensuring fair attribution of importance across tokens.

3. For each input prompt, SHAP computes attribution scores for individual tokens that quantify their contribution to the model's prediction relative to a baseline input.

4. Since SHAP operates at the subword-token level, we aggregate subword attribution scores into word-level scores by summing the contributions of all subword tokens whose character spans overlap with each word.

5. To obtain binary word labels, we threshold the resulting word-level SHAP scores. The threshold is tuned on the dev split of the training data to maximize word-level F1 score, and the same threshold is applied during test-time evaluation. Words with scores above the threshold are labeled as unsafe, and all others are labeled as safe.

Model	Model Size	Inference time (ms/input)	GPU memory use (GB)
LEG xs	22M	7.81	1.01
LEG base	86M	8.28	1.67
LEG large	304M	14.57	3.06
Llama Prompt Guard 2	22M	9.17	1.04
Llama Prompt Guard 2	86M	9.47	1.90
DuoGuard	500M	16.47	–
Toxic-Chat-T5 Large	770M	31.95	3.68
GuardReasoner	1B–8B	26.66–35.77	78.00
Llama Guard 3	1B	58.88	–
ShieldGemma	2B	57.87	–

Table 8: Inference time and GPU memory usage across models.

G Computational efficiency

We evaluate the efficiency of guardrail models by measuring both inference time latency and GPU memory required for inference. All experiments were performed using an NVIDIA H100 GPU. For fairness, we performed inference sequentially on the WildGuardMix test set without batching and report the average inference time across the full set. Results for DuoGuard, Llama Guard 3, and ShieldGemma were taken from (Deng et al., 2025), and results for GuardReasoner were taken from (Liu et al., 2025). Since these works also report inference time on the same H100 GPU, we regard the comparison as fair. For all other models, we locally reproduced the experiments under the same setup. Table 8 summarizes the results.

The LEG family is consistently efficient: LEG xs achieves 7.81 ms per input using only 1.01 GB of memory, while LEG base and LEG large remain lightweight at 8.28 ms / 1.67 GB and 14.57 ms / 3.06 GB, respectively. In comparison, small to mid sized baselines such as Llama Prompt Guard 2 (9.17 to 9.47 ms, 1.04 to 1.90 GB), DuoGuard (16.47 ms), and Toxic-Chat-T5 Large (31.95 ms, 3.68 GB) show slower inference and higher memory use. Larger guardrails are substantially more expensive: GuardReasoner requires 26 to 36 ms per input and up to 78 GB of memory, while Llama Guard 3 and ShieldGemma exceed 57 ms. Overall, LEG offers substantial efficiency gains. Compared to GuardReasoner, LEG xs is over 3× faster and about 75× more memory efficient. Similarly, compared to Llama Guard 3 and ShieldGemma, LEG xs is over 7× faster. These results show that LEG achieves significant speedups and memory savings while remaining lightweight across all configurations.

Crucially, LEG achieves this efficiency while supporting both prompt classification and explanation generation. Competing methods typically provide only prompt classification without explanations yet still demand more resources. This makes LEG the first guardrail to combine lightweight inference with faithful explanation generation, enabling both efficiency and transparency for real time deployment.

H Error analysis

The in-domain performance of LEG base on the ToxicChat0124 dataset is lower than that of other baselines. We found this is due to the model exhibiting high recall (89.5%) but low precision (54.09%). It is well known that precision can be improved through probability threshold tuning. We tested this by treating the prediction threshold as a hyperparameter and selecting the best value using the development set. With this adjusted threshold, LEG base achieves an F1 score of 75% on the in domain ToxicChat0124 dataset. To ensure a fair comparison with other baselines, we did not apply threshold tuning to any model during our evaluation. However, we observed that this tuning strategy improves the performance of almost every variant of LEG.

I Human evaluation of synthetic explanation annotation quality

We conducted a human evaluation to assess the quality of the word-level annotations generated by `GPT-4o-mini`. One human expert was asked to label unsafe words in 50 randomly selected prompts from each test set: AEGIS2.0, WildGuard-

Model	Explanation training data	AEGIS 2.0	WildGuardMix	ToxicChat0124
LEG Base	Jury of LLM	50.00	38.71	24.80
LEG Large	Jury of LLM	52.08	40.57	26.70
LEG Base	Our Method	60.33	**60.69**	36.12
LEG Large	Our Method	**66.11**	58.39	**50.36**

Table 9: Explanation classification performance on human-annotated test sets comparing jury-of-LLMs and our synthetic labeling method.

Mix, and ToxicChat0124. We then compared these labels with the GPT-generated annotations using Cohen's Kappa. The agreement scores were 54% percent for AEGIS2.0, 54.50% percent for WildGuardMix, and 43.56% for ToxicChat0124, which indicate moderate agreement. We found that most disagreements were due to differences in phrase boundaries. For example, GPT often highlights shorter keywords like "kill" or "harm", while the human annotator marks longer phrases such as "kill someone" or "cause harm to others". In most cases, the core unsafe terms were present in both annotations.

J Generalization beyond synthetic supervision

Since the explainability classifier of LEG is trained using synthetic explanation labels generated by `GPT-4o-mini`, an important question is whether its explainability performance is upper-bounded by that of `GPT-4o-mini`, or whether LEG can learn explainability behavior that generalizes beyond the synthetic supervision used during training. To examine this, we evaluated both LEG and `GPT-4o-mini` against human-annotated gold labels, as described in Appendix I. For `GPT-4o-mini`, we conducted a standard zero-shot prompting evaluation in which the model was asked to identify unsafe words or phrases in each input prompt. Table 10 reports the results of this evaluation.

Model	AEGIS 2.0	WildGuardMix	ToxicChat0124
Zero-shot GPT-4o-mini	53.28	54.87	**50.96**
LEG base	60.33	**60.69**	36.12
LEG large	**66.11**	58.39	50.36

Table 10: Performance of explainability classification on the human-annotated test set, reported using F1 scores.

On AEGIS2.0, LEG base achieved an F1 score of 60.33%, while LEG large reached 66.11%. On WildGuardMix, LEG base obtained 60.69%, and LEG large achieved 58.39%. On the more challenging ToxicChat0124 dataset, LEG base achieved 36.12%, and LEG large reached 50.36%. Overall, both LEG base and LEG large outperform `GPT-4o-mini` on AEGIS2.0 and WildGuardMix. On ToxicChat0124, `GPT-4o-mini` performs slightly better. However, LEG large achieves comparable performance, with a difference of less than one percentage point. These results demonstrate that, although the explainability classifier in LEG is trained using synthetic explanation labels generated by `GPT-4o-mini`, its performance is not upper-bounded by `GPT-4o-mini`. Instead, LEG surpasses the performance of `GPT-4o-mini` in explainability classification in most cases. These results further confirm that LEG learns explainability behavior that aligns closely with human judgment.

K Jury-of-LLMs vs. our method for synthetic explanation generation

One of the most common approaches for generating synthetic labels is to use a jury of LLMs. To compare our synthetic explanation labeling method (Section 3.2) with this approach, we conducted an experiment using a jury of three LLMs. We used three open-source models, Llama-3.1-8B-Instruct, Mistral-7B-Instruct-v0.3, and Qwen3-8B, to generate explanation labels. We prompted the jury models to identify unsafe words. A word was labeled as unsafe only when all three models unanimously agreed. We then trained LEG using these jury-generated explanation labels and evaluated its performance on explanation classification using human-annotated test sets.

The results in Table 9 show that explanation classification performance improves substantially when the model is trained on synthetic labels generated by our method, compared to labels produced by a jury of LLMs. This indicates that our confirmation-bias-based labeling approach aligns more closely with human judgments and achieves better generalization.

Unsafe category	Prompt classification	Explainability classification
Causing material harm by disseminating misinformation	97.7	70.27
Copyright violations	87.3	47.7
Cyberattack	98.9	70.3
Defamation encouraging unethical or unsafe actions	98.9	69.9
Disseminating false or misleading information, encouraging disinformation campaigns	98.9	69.9
Fraud assisting illegal activities	89.9	76.8
Mental health over reliance crisis	94.4	70.4
Others	79.8	85.8
Private information individual	87.5	75.8
Sensitive information organization government	92.5	75.0
Sexual content	88.1	76.0
Social stereotypes and unfair discrimination	77.4	73.0
Toxic language hate speech	100.0	78.5
Violence and physical harm	98.7	67.6

Table 11: Performance of LEG base on fine-grained safety categories of the WildGuardMix test set, reported using F1 score.

Unsafe category	Setting 1 performance (Seen topic)		Setting 2 performance (Unseen topic)	
	Prompt	Explainability	Prompt	Explainability
Copyright violations	87.3	47.7	62.2	40.4
Disseminating false or misleading information, encouraging disinformation campaigns	98.9	69.9	98.9	67.3
Fraud assisting illegal activities	89.9	76.8	88.9	74.1
Violence and physical harm	98.7	67.6	100.0	67.7

Table 12: Performance of LEG on risk topics seen and unseen during training. Reported values are F1 scores for both prompt classification and explainability classification.

L Performance on fine-grained risk categories

The WildGuardMix dataset contains annotations for 14 fine-grained risk categories. Table 11 shows the performance of LEG base across all fine-grained risk categories in the WildGuardMix test set. The results indicate that our method robustly learns a diverse set of safety risks, achieving consistently strong performance across categories. In particular, LEG base achieves perfect performance on relatively easier categories such as *toxic language and hate speech* (100% F1) in prompt classification. At the same time, it maintains strong and reasonable performance on more challenging categories, including *copyright violations* and *social stereotypes and unfair discrimination*, for both prompt and explainability classification. These results highlight the ability of our method to handle both common and difficult safety risks in a unified framework.

M Performance on unseen risk topics

To evaluate whether LEG can generalize to unseen risk topics, we conduct an experiment using the WildGuardMix dataset. In addition to binary safety labels (safe vs. unsafe), WildGuardMix annotates unsafe prompts with 16 fine-grained risk topics (e.g., violence, copyright violations). For this experiment, we randomly select four risk topics from WildGuardMix and train a variant of LEG after removing all training instances associated with these topics. We then compare its performance against a model trained on the full dataset.

Specifically, we consider the following two training settings:

1. *Setting 1 (Full training):* The model is trained on the complete WildGuardMix training set, covering all risk topics.

2. *Setting 2 (Topic-excluded training):* The model is trained on a subset of the training data that excludes all instances from four randomly selected risk topics (shown in Table 12). As a result, prompts associated with these topics are entirely unseen during training.

Importantly, in both settings, the model is trained and evaluated using the same binary prompt-level labels (safe vs. unsafe). The exclusion affects only the topical content of the training data, not the label space.

The objective of this experiment is to assess whether LEG can correctly classify prompts from previously unseen risk topics as unsafe. Table 12 reports test-set performance for the four topics under both training settings. The results show that in three out of four cases, LEG achieves performance

Train Dataset	Model	Model Size	FPR	F1 score
–	Llama Guard 3†	1B	–	43.4
	ShieldGamma†	2B	–	69.4
	Llama Guard 2†	8B	–	88.88
	Llama Guard 3†	8B	–	88.4
	DuoGuard†	0.5B	–	82.3
AEGIS2.0	Prompt Baseline base	86M	38.8	75.99
	Prompt Baseline large	304M	33.6	75.43
	LEG base	86M	40.0	75.0
	LEG large	304M	22.0	77.99
WildGuardMix	Prompt Baseline base	86M	5.6	87.07
	Prompt Baseline large	304M	6.4	88.08
	LEG base	86M	5.6	85.56
	LEG large	304M	**1.6**	**92.91**
ToxicChat0124	Prompt Baseline base	86M	12.4	67.24
	Prompt Baseline large	304M	12.4	69.12
	LEG base	86M	6.8	72.73
	LEG large	304M	11.2	77.09

† F1 scores are reported as provided in (Deng et al., 2025).

Table 13: Prompt classification false positive rate (FPR) and F1 score on XSTest across different training datasets and models.

comparable to the full-training setting, despite having no exposure to these topics during training. The only notable drop occurs for *copyright violations*, where accuracy decreases to 62.2%, which is still reasonable given that this topic is entirely unseen during training.

These results suggest that LEG does not rely on memorizing topic-specific lexical cues. Instead, it learns higher-level semantic signals associated with unsafe behavior, enabling robust generalization to unseen risk categories.

N Evaluation on benign prompts containing harmful words

XSTest (Röttger et al., 2024) is a stress-test benchmark designed to evaluate robustness against superficial lexical cues in safety classification. Unlike standard safety datasets, XSTest focuses on benign prompts that intentionally use words commonly associated with harmful content. For example, it includes prompts such as "how to kill a Python process?", which contain word like 'kill' but express clearly non-harmful intent. As a result, strong performance on XSTest indicates that a model is less reliant on keyword-based heuristics and better able to reason about context and intent.

We evaluate the prompt classification performance of LEG. Since this dataset does not include explainability annotations, we do not evaluate the explainability classifier on this benchmark. Table 13 reports F1 scores on XSTest for several existing guardrail models and for LEG trained on different datasets. Additionally, we report the false positive rate (FPR) for LEG and the Prompt Baseline described in Section 4.

LEG achieves strong performance on XSTest despite being orders of magnitude smaller than existing guardrail models. In particular, LEG large trained on WildGuardMix achieve the highest F1 score (92.91), outperforming all evaluated baselines, including large instruction-tuned guardrails such as Llama Guard 2 (8B) and Llama Guard 3 (8B). Notably, this result is achieved with a model that is several orders of magnitude smaller in parameter count.

Even the base variant of LEG (86M parameters) performs competitively, achieving an F1 score of 85.56 on XSTest. These results highlight that LEG can effectively handle exaggerated safety scenarios and benign prompts containing harmful lexical cues. Further, LEG achieves a lower FPR in most cases than the Prompt Baseline, indicating that joint training with both prompt-level and explanation-level supervision helps reduce false positives compared to a model trained with only prompt-level supervision. Overall, these results demonstrate that LEG learns context-sensitive safety representations rather than relying on superficial lexical cues. Crucially, it achieves this robustness while remaining lightweight.