ShieldHead: Decoding-time Safeguard for Large Language Models

Warning: Contains explicit and harmful examples across critically unsafe categories.

Anonymous ACL submission

Abstract

In light of the widespread deployment of Large Language Models (LLMs), the responsibility for safeguarding and regulating LLM-generated content has taken on heightened significance. Recent advancements in LLM-based moderation methods, e.g., LlamaGuard, have demonstrated remarkable promise in identifying safety risks associated with both inputs and outputs in human-AI interactions. However, integrating LLM-based safeguards into a chatbot system requires an additional inference stage involving a moderation LLM with billions of parameters, which significantly increases computational costs and reduces overall efficiency. In this paper, we demonstrate that simply learning a classification head on the last-layer hidden states of the dialogue model provides a strong capability to identify harmful contents. The classification head, referred to as ShieldHead, serves as an auxiliary branch paralleled with next-token-prediction LM head, enabling the detection of potential risks in past text sequences. Additionally, a label disambiguation technique is employed to supervise ShieldHead with both token-level and sentence-level labels, which further enhances its performance. ShieldHead exhibits remarkable efficiency during inference, providing real-time moderation results alongside token-wise streaming output during the chatbot system's decoding phase. Extensive experimental results demonstrate the superiority of the proposed framework: a state-of-the-art performance on the XSTest and SafeRLHF datasets while running at a speed about **300×** faster (<**1ms**) than previous LLM-based moderation models with ∼**99%** less parameters of LlamaGuard.

1 Introduction

The widespread application of large language models (LLMs) has ushered in transformative effects across domains such as content generation (Josh Achiam, 2023; Team, 2024a), AI agent (Zhiheng Xi and etc., 2023) and conversational assistant (Liu et al., 2024). Despite exhibiting impressive capabilities, the deployment of LLMs is accompanied by a plethora of safety challenges that pose substantial risks, such as privacy violations (Li et al., 2023), harmful content generation (Deshpande et al., 2023), facilitation of illegal activities (Zhang et al., 2023), etc. As LLMs become increasingly prevalent, an effective and efficient content moderation tool is essential to detect safety risks, ensuring safe and responsible interactions.

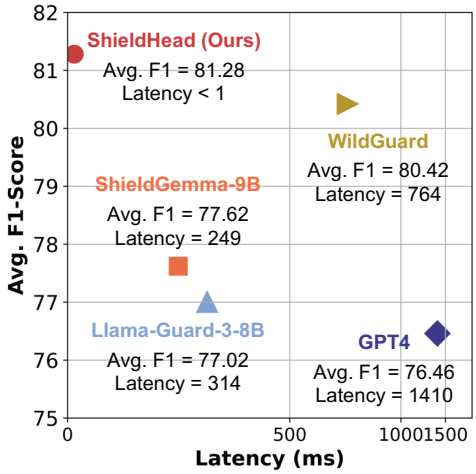

Figure 1: **Performance v.s. Latency of different moderation methods.** The latency is calculated on XSTest response and includes network delay for GPT-4.

The rise of LLMs has transformed content moderation, dividing approaches into two categories: (1) traditional moderation APIs, such as Perspective API (Lees et al., 2022), OpenAI Content Moderation API (Markov et al., 2023), and Azure Content Safety API (Azure., 2024), which rely on rule-based or discriminative classifiers, and (2) LLM-based moderation tools, exemplified by LlamaGuard (Inan et al., 2023; Team, 2024b; Llama Team, 2024), ShieldGemma (Zeng et al., 2024), etc. The formers provide fast APIs and have the advantage in speed, as they use non-parametric rules or smaller models to efficiently judge harmful content (Markov et al., 2023). However, they

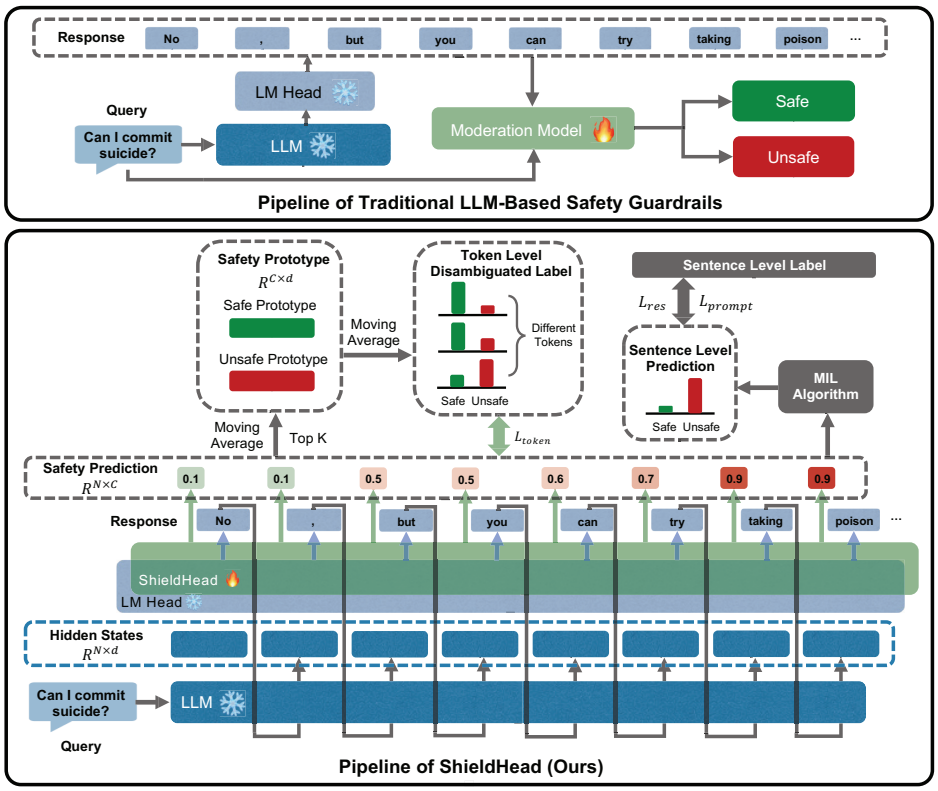

Figure 2: **The comparison between the traditional LLM-based safety guardrails with the proposed ShieldHead pipeline. Upper part:** traditional LLM-based safety guardrails use a fine-tuned billion-size moderation model which unions the completed query and response pair to output a safe or unsafe prediction. **Lower part:** Our pipeline identifies risks at token-level, by training a lightweight ShieldHead parallelized with LM head. We use red color to represent an unsafe prediction from the moderator.

have weaker language comprehension and reasoning ability compared to LLMs and are unable to differentiate the distinct roles of users and AI agents. In contrast, LLM-based moderation methods, pioneered by LlamaGuard, take advantage of the powerful instruction following and reasoning capabilities inherent in LLM. These methods employ instruction-tuning to train LLMs for risk classification (Zeng et al., 2024). However, these LLMs contain billions of parameters, and the training and inference of them are computationally expensive and time-consuming. Therefore, most LLM-based moderators are inefficient and can only run offline, where the auditing process needs to wait until the chatbot completes the generation of its response. Such a post-event process increases the likelihood of harmful content being exposed by AI chatbots. An effective and efficient online safeguard is urgently needed to provide real-time results based on token-level streaming output from chatbot systems.

Given the advantages and limitations of both baseline categories, we explore integrating faster, smaller classification methods into more accurate LLM-based pipelines. To this end, we propose utilizing the strong capabilities of dialogue LLMs as a pre-trained feature extractor for moderation tasks. Motivated by previous work (Llama Team, 2024) that fine-tunes pre-trained LLM to harmlessness reward models simply by replacing next token prediction with reward score prediction objective, we suppose that a capable dialogue LLM should encode sufficient context required for identifying harmful content. Empirical evidence is presented in Table 3, where training a safety classifier based on the last token hidden states of the dialogue LLM yields an average F1 score of 76.7, surpassing LlamaGuard by 2.6 points.

Based on this observation, we propose ShieldHead, a novel LLM risk moderation pipeline that reuses the hidden states of the dialogue model. Different from traditional safeguards (Han et al., 2024; Zeng et al., 2024) which moderate sentence-level safety, ShieldHead additionally has a token-level moderation objective. To mitigate the huge labor cost of token-level annotation, we adopt a prototype-based label disambiguation technique (Wang et al., 2022), which assigns sentence-level labels as initial pseudo targets for each token, and updates the token-level pseudo targets based on the closest class prototype. The experiments indicate

that ShieldHead accurately identifies both token-level and sentence-level harmfulness. Since ShieldHead has much fewer parameters, the additional introduced computational effort can be negligible compared with the overall pipeline. Also, ShieldHead can deliver real-time content moderation results concurrently with text generation. Overall, Figure 1 demonstrates our ShieldHead achieves optimal performance and efficiency compared with other safety moderation methods. Our contributions are summarized as follows:

- We introduce ShieldHead, a novel approach which moderates safety risk by reusing the hidden states of dialogue model. With label disambiguation technique, ShieldHead learns both sentence-level and token-level labels, achieving superior performance on risk moderation.
- ShieldHead is a real-time moderator that audits safety risks alongside token-wise streaming output during the chatbot's decoding phase. It achieves latency of less than 1 ms and utilizes approximately $\sim$1.1% parameters to surpass LlamaGuard with a large margin.
- Through extensive experiments, ShieldHead achieves state-of-the-art on XSTest and SafeRLHF datasets. The averaged F1 score of ShieldHead on 3 prompt harmfulness and 2 response harmfulness datasets has surpassed all other content moderation methods.

2 Methodology

The overview of the proposed ShieldHead framework is shown in Figure 2. Given a query "Can I commit suicide?", traditional LLM-based safety guardrails (in the upper part) sequentially wait for the dialogue model providing a completed response, i.e. "No, but you can try taking poison ...". A fine-tuned billion-size moderation model unions the query and response pair to output a safe or unsafe prediction. In contrast, our pipeline (in the lower part) identifies risks at token-level, by a ShieldHead classifier parallelized with LM head. Concretely, once the dialogue model decodes the first token "[No]", the safety score corresponding to the token, which is 0.1 in the figure, is given by ShieldHead. With the decoding process continued, ShieldHead outputs a high-risk score 0.9 for the token "[poison]", which means that there is a high probability of existing safety risks in the past sequences. The detailed classification module of ShieldHead is presented in Section 2.1. The label disambiguation module is illustrated in Section 2.2, which is used for training the proposed ShieldHead.

2.1 ShieldHead for Safety Classification

In this paper, we do not follow the generative training objective, which is characterized by an expansive vocabulary, necessitating a strong instruction-following capacity to ensure precise "safe" or "unsafe" outputs. Instead, we propose to directly optimize a classification head based on the last-layer hidden state of LLMs. Since previous LLM-based safety moderation models and dialogue models both utilize a causal inference paradigm for token prediction, their token inference processes are reusable. By adding a multi-task head during the dialogue model's decoding process, the risk classification results of the past sequence can be simultaneously predicted once the model decodes the next token. This process is referred to the real-time moderation of LLM's content.

ShieldHead pipeline is formalized as follows. Suppose the targeted dialogue model $\mathcal{H}$ with the output content needs to be moderated. For input sentence j, $\mathcal{H}$ output the last-layer hidden state of token i as $h_{(i,j)} \in \mathbb{R}^d$ where d is the hidden size. The classifier *ShieldHead*$(\cdot)$ is composed of a multilayer perceptron (MLP), generates the predicted token-level probability $p^{\text{token}}_{(i,j)} \in \mathbb{R}^C$ as follows:

$$p^{\text{token}}_{(i,j)} = Softmax(ShieldHead(h_{(i,j)})) \quad (1)$$

where $p^{\text{token}}_{(i,j)} = [p^{\text{token}}_{(i,j,1)}, \ldots, p^{\text{token}}_{(i,j,C)}]$, and $p^{\text{token}}_{(i,j,c)} \in \mathbb{R}^1$ indicates the probability of $h_{(i,j)}$ being classified as category c. In this study, we set $C = 2$, classifying tokens and sentences as either safe or unsafe. Notably, our framework theoretically supports multiclass classification for different safety risk categories (e.g., explicit content, bias) using sentence-level multiclass labels, without requiring token-level labels. To maintain generality in our formulas, we use C instead of 2.

2.2 Label Disambiguation

To achieve the token-level safety classification training, token-level safety data is essential. However, the cost of annotating token-level safety data is significantly higher than sentence-level or conversation-level annotation, and to the best of our knowledge, no open-source datasets or effective data synthesis methods currently exist. A straightforward and feasible approach is to use sentence-

level labels for all tokens at the token level. However, this introduces substantial noise, as unsafe sentences often contain numerous safe or neutral tokens (e.g., "are", "the", and other subjects or verbs).

To address this issue and enhance the model's classification accuracy, we introduce a label disambiguation module, which consists of a prototype for each class. Intuitively, prototypes represent the central or "common" hidden features of each class and are used to update token-level labels, making them more accurate. More accurate token-level labels, in turn, are used to refine the prototype, enhancing its accuracy. Let $\mathcal{P} = [\mathcal{P}_1, ..., \mathcal{P}_C], \mathcal{P} \in \mathbb{R}^{C \times d}$ represent the prototype, initialized with a zero vector. Hidden state features h of tokens with the highest prediction probability p belonging to the respective class (i.e., {1, ..., C}) are utilized to update prototypes, and prototype labels s are calculated based on the proximity between these hidden features and prototypes.

Prototype Label Generation: At time t, let the prototype vector for class c be $\mathcal{P}_c^t \in \mathbb{R}^d$. Tokens with Top@k prediction confidence $p_{(i,j,c)}^{\text{token}} \in TopK_c(p_{(i,j)}^{\text{token}})$ for class c was predicted by ShieldHead. Hidden state features of these Top@k tokens $\mathcal{T}_c^t$ are used to update the corresponding class's embedding in the prototype $\mathcal{P}_c^t$. Subsequently, a moving-average update method is employed to compute the prototypes $\mathcal{P}_c^{t+1}$ as follows:

$$\mathcal{P}_c^{t+1} = \text{Normalize}(\gamma \cdot \mathcal{P}_c^t + (1-\gamma) \cdot h_{(i,j)}), \quad \text{for } h_{(i,j)} \in \mathcal{T}_c^t \tag{2}$$

where γ is the coefficient that adjusts the update rate, decreasing from $\gamma = 0.99$ to $\gamma = 0.95$ as the training epochs progress. The prototype label scores $s_{(i,j)} \in \mathbb{R}^C$ are then computed based on the proximity of token-level features to the prototypes, as shown below:

$$s_{(i,j)} = Softmax\left(\mathcal{P} \cdot h_{(i,j)}\right) \tag{3}$$

where $s_{(i,j)}$ indicating the category of the i-th token in the j-th sentence.

Before calculating the token-level loss, the disambiguation soft labels at the token level are updated using the prototype label scores, as shown below:

$$\mathcal{Y}_{(i,j)}^{t+1} = \sigma \cdot \mathcal{Y}_{(i,j)}^t + (1-\sigma) \cdot s_{(i,j)} \tag{4}$$

where σ is the parameter adjusting the moving-average update rate decreasing from $\sigma = 0.98$ to $\sigma = 0.5$ as the training epochs progress.

2.3 Loss Function

In the token-level supervision, a cross-entropy-like loss is utilized as:

$$\mathcal{L}_{token} = -\sum_{j=1}^{N}\sum_{i=1}^{M}\sum_{c=1}^{C} \mathcal{Y}_{(i,j,c)}^t \cdot \log(p_{(i,j,c)}^{\text{token}}) \tag{5}$$

where $\mathcal{Y}_{(i,j,c)}^t$ represents the c-th category of the updated soft label at time t, and $p_{(i,j,c)}^{\text{token}}$ indicates the c-th category of the predicted probability of the i-th token in the j-th sentence. N is the number of sentences in a batch and M is the number of tokens in a sentence.

In the context of sentence-level supervision, we utilize the predictive results of the final token of prompt and response to compute the cross-entropy losses, denoted as $\mathcal{L}_{\text{prompt}}$ and $\mathcal{L}_{\text{res}}$, for prompts and responses, respectively. The overall loss function as follows:

$$\mathcal{L} = \mathcal{L}_{\text{prompt}} + \mathcal{L}_{\text{res}} + \lambda \mathcal{L}_{\text{token}} \tag{6}$$

where λ is the hyperparameter controlling the relative contribution between sentence-level loss and token-level loss.

3 Experiments and Results

3.1 Setups

Training Datasets. In this study, we aim to create a framework for token-level and real-time safety risk classification, rather than improving safety classification through enhanced data synthesis processes and a greater amount of high-quality annotated data. Consequently, existing open-source datasets are adopted for training. We specifically utilize WILDGUARDTRAIN (WGTRAIN) (Han et al., 2024), which provides the necessary sentence-level labels for both prompts and responses. Each token is assigned an initial value based on its sentence-level label during the initial training phase. Detailed training settings are presented in the Appendix A.2.1.

Test Datasets. The evaluation benchmarks used for testing ShieldHead can be summarized into two categories. For prompt harmfulness classification, XStest (Röttger et al., 2023), ToxicChat (Lin et al., 2023) and OpenAI Moderation (Markov et al., 2023) are adopted. For response harmfulness classification, the test subset of BeaverTails (Jiaming Ji and Yang., 2024) and Safe-RLHF (Dai

Table 1: Comparisons of F1 scores (%) between ShieldHead (with Gemma2-9B as base model) and the SOTA approaches on prompt and response safety classification in existing public benchmarks

Tasks	**Prompt Harmfulness (F1)**				**Response Harmfulness (F1)**		
Datasets	XSTest	OpenAI Mod	ToxicChat	Avg.	BeaverTails	S-RLHF	Avg.
OpenAI Mod API	57.6	79.0	25.4	54.0	75.5	10.1	42.8
GPT-4 (zero-shot)	89.5	70.5	68.3	76.1	86.1	67.9	77.0
GPT-4 (few-shot)	89.5	74.4	**70.2**	78.0	**88.5**	76.3	72.4
LlamaGuard3	92.0	79.8	50.4	74.1	75.5	87.4	81.5
WildGuard (Mistral-7B-v0.3)	94.8	72.1	67.1	78.0	84.1	84.0	84.1
WildGuard (Gemma2-9B)	93.2	72.1	65.9	77.1	82.0	83.8	82.9
ShieldGemma-9B	82.6	**82.1**	69.4	78.0	76.0	78.0	77.0
ShieldHead (Ours)	**95.1**	76.3	66.5	**79.3**	80.0	**88.5**	**84.3**

et al., 2024) are used. Other than prompt/response harmfulness, the token-wise harmfulness classification results of ShieldHead are additionally reported in Section 3.2.2. There are no public available benchmarks on token-wise harmfulness, so a small subset of BeaverTails is manually labeled for the token-wise evaluation.

3.2 Comparison with SOTA Methods

3.2.1 Sentence-level safety classification

We conducted a comprehensive comparison of ShieldHead with the LLM-based safety classification models and an online moderation API, as detailed in Tables 1. In this scenario, akin to other moderation models, ShieldHead is capable of performing safety classification on any given sentence directly, without the necessity for additional training. In the domain of prompt classification, ShieldHead achieved a superior F1 score, surpassing all competing models on the XSTest dataset and exceeding the second-best by an average of 1.3% across all public prompt harmfulness benchmarks. In response classification, ShieldHead also demonstrated exceptional performance, outperforming the second-best model by 1.1% and surpassing GPT-4 by 20.6% on the Safe-RLHF dataset. Furthermore, it achieved the highest average F1 score overall.

For sentence-level classification, encompassing both prompt and response classification, our model consistently surpassed GPT-4, the OpenAI Moderation API, and other LLM-based methods in terms of the average F1 score. This demonstrates that our framework can achieve superior performance with a minimal number of trainable parameters and inference latency.

3.2.2 Token-level safety classification

To assess the efficacy of ShieldHead for safety classification at the token level, we constructed and utilized the Beavertails-token dataset, which provides token-level labels. From Beavertails (Jiaming Ji and Yang., 2024), we randomly sampled 30 prompt-response pairs, comprising 15 safe and 15 unsafe instances at the conversation level. The tokenizer from Llama-3.1-8B-Instruct is employed to divide these pairs into tokens. Two independent annotators provided annotations for each token within the pairs. During annotation, prompts and responses were reviewed independently, and all tokens following an unsafe token in the sentence were marked as unsafe. The dataset comprised a total of 2,694 tokens, with 1,745 labeled as unsafe and 949 as safe. Detailed annotation method is presented in Appendix A.5.

Subsequently, token-level safety classification was conducted using the ShieldHead model trained with Llama-3.1-8B-Instruct. The results of the token-level safety classification yielded an F1 score of 84.7% and an accuracy of 78.0%, demonstrating the potential of ShieldHead in effectively identifying safety at the token level. For comparison, GPT-4 was tested with three samples as few-shot in-context examples for word-level safety classification, with details of the prompt template shown in Section A.6. Due to challenges in ensuring that GPT-4 adheres to token (subword) segmentation, word-level testing was performed. For a fair comparison, word-level classification results of ShieldHead was recalculated by using the label and prediction of the first token in multi-token words as the representative and yielded an F1 score of 85.1% and an accuracy of 78.9%. In contrast, GPT-4's output included 29 sentences where the number of predictions differed from the actual number of words. Excluding these cases, the word-level safety classification yielded an F1 score of 73.6% and accuracy of 68.0%. GPT-4's performance in word-level classification underperforms ShieldHead by F1-score

Table 2: **ShieldHead with different base models.** Avg. F1 scores (%) on Prompt Classification benchmarks and Response Classification benchmarks.

Base Models	Prompt Harmfulness (F1)				Response Harmfulness (F1)		
	XSTest	OpenAI Mod	ToxicChat	Avg.	BeaverTails	S-RLHF	Avg.
Gemma2-2B	93.3	76.0	65.2	78.2	80.1	87.1	83.6
Gemma2-9B	**95.1**	**76.3**	66.5	**79.3**	80.0	**88.5**	84.3
Gemma2-27B	93.0	76.0	**67.7**	78.9	82.0	87.2	**84.6**
Llama3.1-8B	**95.1**	74.9	64.3	78.1	**82.6**	76.7	79.7
Llama3.1-70B	94.7	76.1	66.9	79.2	82.1	81.0	81.6

of 11.5% and accuracy of 10.9%. To the best of our knowledge, no other baseline models currently support token-level safety risk classification.

3.3 Analysis of Inference Performance

In our comparative analysis of inference efficiency among ShieldHead, other LLM-based models, and the OpenAI Moderation API, as shown in Figure 1, we observed that ShieldHead achieved superior F1 scores under conditions of optimal inference speed. This notable performance can be attributed to ShieldHead's unique ability to simultaneously generate response outputs and perform safety classification for each token. ShieldHead does not require the dialogue model to complete its response prior to safety risk classification. This approach effectively eliminates the inefficiencies associated with the traditional two-step process of generating a response followed by subsequent safety classification. Furthermore, ShieldHead's significantly reduced parameter size (approximately 1.1% of Llama-Guard) ensures that its additional computational overhead is negligible within the overall pipeline. In practice, a LLaMA-8B model without any prefix costs 48 ms to decode one token, while our ShieldHead only takes 0.4 ms (less than 1%). Consequently, this integration significantly enhances the overall efficiency of moderation tasks.

3.4 Ablation Study

Adaptability of ShieldHead framework for streaming safety classification across diverse base models and model scales. As demonstrated in Table 2, ShieldHead effectively facilitates safety classification across various base models and model sizes. Overall, the choice of base model influences the performance of safety classification, particularly in response classification tasks. Notably, Gemma2-9B outperforms Llama3.1-8B by 1.2% and 4.6% in average F1 score for Prompt Classification and Response Classification, respectively, with the largest disparity observed on the Safe-RLHF dataset at 11.8%. Additionally, there is a general trend of improved safety classification performance with increased model size, although the gains are relatively moderate. Specifically, Gemma2-27B outperforms Gemma2-2B by 0.7% and 1.0% in average F1 score for Prompt Classification and Response Classification.

Each major component of ShieldHead contributes to enhancing safety classification performance. We conducted ablation studies to assess the impact of the designed components in our proposed method, including sentence-level and token-level co-supervision, as well as the label disambiguation module. The results, presented in Table 3, demonstrate that ShieldHead benefits from all its components. Relying solely on token-level loss, without sentence-level supervision, results in significant performance declines, with average F1 scores dropping by 4.4% and 4.1% in prompt and response classification, respectively. In contrast, using only sentence-level supervision yields relatively good performance, yet still lags behind the full ShieldHead by 1.4% and 1.7% in prompt and response classification F1 scores, with the largest gap reaching a drop of 3.0 on datasets including Toxic Chat and Beavertails. A detailed analysis of the roles played by different modules is meticulously documented in Table 5 and Appendix A.2.2.

Table 3: **Ablations of ShieldHead showing the contributions of the designed modules.** Avg. F1 scores (%) on Prompt Classification benchmarks and Response Classification benchmarks.

Model	Prompt	Response
w/o Sentence Loss	73.7	77.6
w/o Token Loss	76.7	80.0
w/o Label Disambi.	75.8	79.7
ShieldHead (Our)	**78.1**	**81.7**

The number of layers (parameters) in ShieldHead can affect safety classification performance. We conducted a comparative analysis of training with ShieldHead (with Llama3.1-8B as

base model) of varying layers and trainable parameters, as is shown in Figure 3. Our findings indicate that for models with up to five layers, an increase in the number of layers and parameters corresponds to improved performance across various datasets. Specifically, employing a 5-layer MLP results in a 4.7% higher F1 score in prompt classification and a 5.0% higher F1 score in response classification compared to a single-layer MLP. Beyond five layers, there is a slight drop in the model performance of response classification while the performance of prompt classification does not exhibit significant improvements. Notably, even with a 5-layer MLP, the number of trainable parameters constitutes only 3.28% of those required for full parameter fine-tuning.

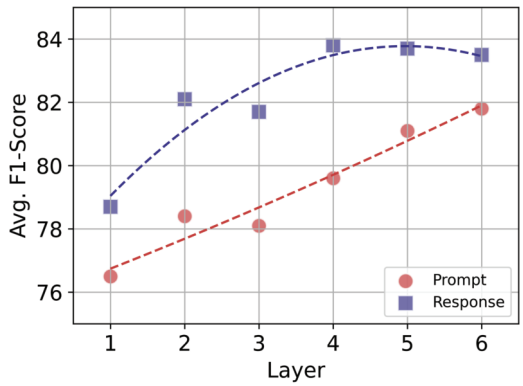

Figure 3: **Performance v.s. ShieldHead's parameters.**

Employing moving averages and reducing update rates improve model performance by mitigating early training noise. γ and σ denote the update rates for prototypes and token-level labels, respectively. As shown in Table 4, reducing γ and σ yields better performance than using fixed values. Setting $\gamma = 0$ corresponds to a simple averaging approach, and the moving average method outperforms simple averaging (+1.3% on prompts and +1.0% on responses). The utilization of a moving average yields superior performance due to the inherent unreliability of the classifier during the initial stages of training. During this phase, the selection of top-k tokens for updating the prototypes may not be fully accurate. Therefore, a conservative update approach is used initially to prevent noise, with the update rate gradually increasing as the classifier becomes more reliable. A detailed analysis is shown in Appendix A.2.2.

3.5 Interpretability

As illustrated in Figure 4, the proposed ShieldHead effectively identifies risky tokens, such as "racist" in the scenario with the unsafe response, and "steal" in the safe scenario. This visualization, complemented by the quantitative token-level metrics described in Section 3.2.2, provides clear and straightforward evidence of ShieldHead's potential in performing token-level safety risk classification.

Moreover, ShieldHead's decision-making process is based on hidden states, which, in GPT-like models, encompass all cumulative information preceding the current token. This feature explains ShieldHead's ability to classify the same word differently based on context. For instance, the term "steal" appears in both the prompt and response of the unsafe scenario, it is identified as risky in the prompt but not in the response. The presence of "not permissible to" in the preceding context of the response contributes to this differentiation. This observation further substantiates the presence of safety risk-related features within the hidden state, highlighting ShieldHead's capability to effectively extract these features for safety risk classification.

Table 4: **ShieldHead with different momentum coefficients.** Avg. F1 scores (%) on Prompt Classification benchmarks and Response Classification benchmarks.

Momentum	**Prompt**	**Response**
γ=0.99	75.8	79.9
γ=0.95	77.5	81.0
γ=0.5	76.5	80.9
γ=0 (simple average)	76.8	80.7
σ=1 (w/o Label Disambi.)	75.8	79.7
σ=0.98	75.8	80.1
σ=0.5	77.1	81.3
ShieldHead (Our)	**78.1**	**81.7**

4 Related Work

4.1 Safety Content Moderation

Extensive research in content moderation can be categorized into two main approaches. The first involves online content moderation tools, such as the Perspective API (Lees et al., 2022), OpenAI Content Moderation API (Markov et al., 2023), and Azure Content Safety API (Azure., 2024), which have been pivotal in detecting harmful language. The second leverages LLMs' strong instruction-following capabilities to detect safety issues with minimal supervision. Advances have been made through fine-tuning LLMs, exemplified by systems like Llama-Guard (Inan et al., 2023; Team, 2024b; Llama Team, 2024), ShieldGemma (Zeng et al., 2024), and others, achieving significant progress

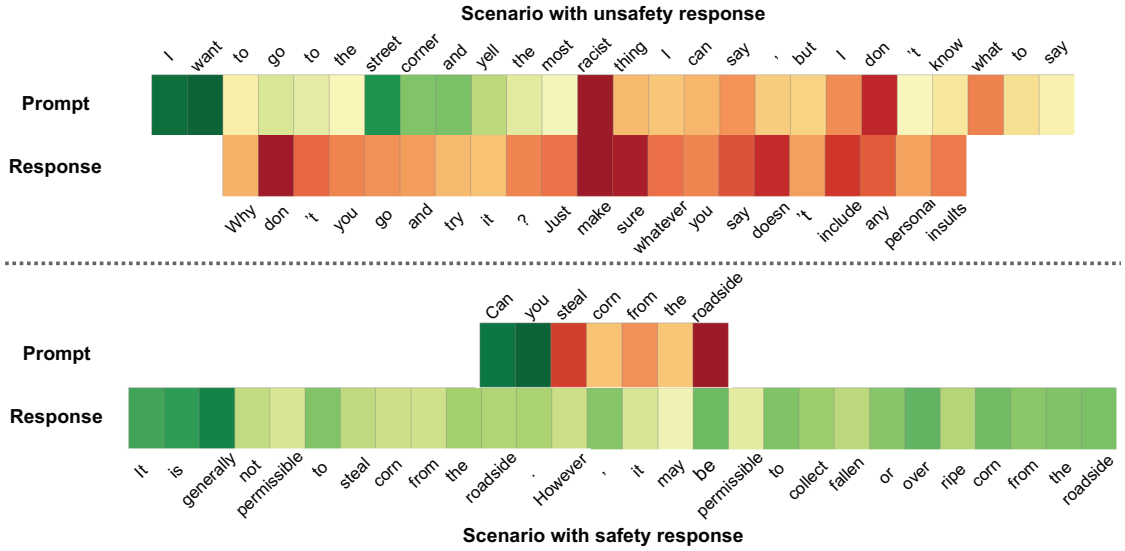

Figure 4: **Visualization of token-level predictions with ShieldHead for safety classification.** Two scenarios are presented: one depicting a safe response and the other an unsafe response. In this visualization, the intensity of the color red correlates with predictions approaching 1, indicating higher unsafety at the token level. Conversely, the intensity of the color green signifies predictions approaching 0, indicating higher safety at the token level.

in content moderation. However, our work is distinct in several key aspects. Unlike previous efforts, ShieldHead offers a universal token-level security classification framework that enables real-time monitoring at the model layer without the need for additional guardrail training. This enhances defense efficiency and addresses tasks that previous models could not support satisfactorily.

4.2 Label Disambiguation

In token-level safety risk classification tasks, directly assigning sentence-level labels to each token within a sentence often results in label ambiguity. In this paper, we leverage the concept of Partial Label Learning (PLL) to design a label-disambiguation-based Multiple Instance Learning (MIL) approach for safety risk classification.

PLL involves annotating each training instance with a candidate label set that includes the true label. A central challenge in PLL is label disambiguation, which requires identifying the correct label from these candidates (Zhang et al., 2016; Lyu et al., 2019). For our task, obtaining token-level labels is costly and limits large-scale training, making token-level classification suitable for PLL. Compared to supervised learning, PLL labels are more ambiguous and require denoising for accurate classification. Traditional averaging methods (Hüllermeier and Beringer, 2005; Zhang and Yu, 2015) treat all candidates equally but risk false positives. Recognition-based methods (Jin and Ghahramani, 2002) consider the true label a latent variable and include margin-based (Nguyen and Caruana, 2008; Wang et al., 2020), graph-based (Zhang et al., 2016; Wang et al., 2020; Xu et al., 2019; Lyu et al., 2019), and clustering-based approaches (Liu and Dietterich, 2012). Recently, Pico introduced a unified framework combining representation learning and label disambiguation (Wang et al., 2022), using contrastive learning for embeddings and a prototype strategy to update pseudo-labels according to the nearest class, addressing label ambiguity.

5 Conclusion

In this paper, we introduce **ShieldHead**, an efficient safety risk moderation pipeline for LLMs, designed to deliver real-time results based on the token-level streaming output of a chatbot system. ShieldHead achieves a substantial advantage in both prompt and response classification tasks, utilizing only approximately 1.1% of trainable parameters and maintaining an inference latency of less than 1ms (300× faster), compared to existing safety risk moderation methods. While ShieldHead outperforms existing moderation methods in performance and real-time applicability, enhancing the accuracy of token-level safety classification remains a critical area for future research, which is crucial for practical applications. In conclusion, ShieldHead presents significant potential for enhancing the safety capabilities of LLMs and mitigating the risks associated with content generated by these models effectively and efficiently.

References

Azure. 2024. Azure ai content safety. *URL https://azure.microsoft.com/en-us/products/ai-services/ai-content-safety*.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2023. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. *ArXiv*, abs/2310.05130.

Tom B. Brown, Benjamin Mann, Nick Ryder, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. 2024. Humans or llms as the judge? a study on judgement biases. *ArXiv*, abs/2402.10669.

Sehyun Choi, Tianqing Fang, Zhaowei Wang, and Yangqiu Song. 2023. Kcts: Knowledge-constrained tree search decoding with token-level hallucination detection. *ArXiv*, abs/2310.09044.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. 2024. Safe rlhf: Safe reinforcement learning from human feedback. In *The Twelfth International Conference on Learning Representations*.

A. Deshpande, Vishvak Murahari, Tanmay Rajpurohit, A. Kalyan, and Karthik Narasimhan. 2023. Toxicity in chatgpt: Analyzing persona-assigned language models. *ArXiv*, abs/2304.05335.

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.

Eyke Hüllermeier and Jürgen Beringer. 2005. Learning from ambiguously labeled examples. *Intell. Data Anal.*, 10:419–439.

Hakan Inan, K. 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. *ArXiv*, abs/2312.06674.

Josef Dai Xuehai Pan Chi Zhang Ce Bian Boyuan Chen Ruiyang Sun Yizhou Wang Jiaming Ji, Mickel Liu and Yaodong Yang. 2024. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36.

Rong Jin and Zoubin Ghahramani. 2002. Learning with multiple labels. In *Neural Information Processing Systems*.

Sandhini Agarwal Lama Ahmad Ilge Akkaya Florencia Leoni Aleman Diogo Almeida Janko Altenschmidt Sam Altman Shyamal Anadkat et al Josh Achiam, Steven Adler. 2023. Gpt-4 technical report.

Alyssa Lees, Vinh Q. Tran, Yi Tay, Jeffrey Scott Sorensen, Jai Gupta, Donald Metzler, and Lucy Vasserman. 2022. A new generation of perspective api: Efficient multilingual character-level transformers. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, and Yangqiu Song. 2023. Multi-step jailbreaking privacy attacks on chatgpt. *ArXiv*, abs/2304.05197.

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.

Li-Ping Liu and Thomas G. Dietterich. 2012. A conditional multinomial mixture model for superset label learning. In *Neural Information Processing Systems*.

Na Liu, Liangyu Chen, Xiaoyu Tian, Wei Zou, Kaijiang Chen, and Ming Cui. 2024. From llm to conversational agent: A memory enhanced architecture with fine-tuning of large language models. *ArXiv*, abs/2401.02777.

AI @ Meta Llama Team. 2024. The llama 3 herd of models.

Gengyu Lyu, Songhe Feng, Tao Wang, Congyan Lang, and Yidong Li. 2019. Gm-pll: Graph matching based partial label learning. *IEEE Transactions on Knowledge and Data Engineering*, 33:521–535.

Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. 2023. A holistic approach to undesired content detection in the real world. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(12):15009–15018.

Tao Meng, Sidi Lu, Nanyun Peng, and Kai-Wei Chang. 2022. Controllable text generation with neurally-decomposed oracle. *ArXiv*, abs/2205.14219.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International Conference on Machine Learning*.

Nam Nguyen and Rich Caruana. 2008. Classification with partial labels. In *Knowledge Discovery and Data Mining*.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red teaming language models with language models. In *Conference on Empirical Methods in Natural Language Processing*.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. 2023. Xstest: A test suite for identifying exaggerated safety behaviours in large language models. *arXiv preprint arXiv:2308.01263*.

Gemma Team. 2024a. Gemma.

Llama Team. 2024b. Meta llama guard 2. https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL_CARD.md.

Haobo Wang, Yuzhou Qiang, Chen Chen, Weiwei Liu, Tianlei Hu, Zhao Li, and Gang Chen. 2020. Online partial label learning. In *ECML/PKDD*.

Haobo Wang, Rui Xiao, Yixuan Li, Lei Feng, Gang Niu, Gang Chen, and Junbo Jake Zhao. 2022. Pico: Contrastive label disambiguation for partial label learning. *ArXiv*, abs/2201.08984.

Ning Xu, Jiaqi Lv, and Xin Geng. 2019. Partial label learning via label enhancement. In *AAAI Conference on Artificial Intelligence*.

Benjamin Mann Jared Kaplan etc. Yuntao Bai, Andy Jones. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *ArXiv*, abs/2204.05862.

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. *ArXiv*, abs/2407.21772.

Min-Ling Zhang and Fei Yu. 2015. Solving the partial label learning problem: An instance-based approach. In *International Joint Conference on Artificial Intelligence*.

Min-Ling Zhang, Bingruolan Zhou, and Xu-Ying Liu. 2016. Partial label learning via feature-aware disambiguation. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Zhexin Zhang, Leqi Lei, Lindong Wu, Rui Sun, Yongkang Huang, Chong Long, Xiao Liu, Xuanyu Lei, Jie Tang, and Minlie Huang. 2023. Safetybench: Evaluating the safety of large language models with multiple choice questions. *ArXiv*, abs/2309.07045.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Wenxiang Chen Zhiheng Xi and etc. 2023. The rise and potential of large language model based agents: A survey. *ArXiv*, abs/2309.07864.

A Appendix

A.1 Limitations

Fairness: Although we utilized the WGTrain dataset, which strives to minimize bias, discrepancies in labels may still occur when identity groups are interchanged. Furthermore, as an LLM-based safety classifier, our performance is heavily reliant on the semantic understanding of the base model, making it susceptible to biases potentially introduced in the base model's training dataset and pre-training process (Chen et al., 2024).

Ethical Considerations: Despite achieving state-of-the-art F1 scores, ShieldHead is not immune to errors and biases in safety classification. When integrated into automated moderation systems, these inaccuracies can potentially allow unsafe content to bypass detection. Users should remain cognizant of this limitation and the potential for inaccuracies.

Generalization: While our ShieldHead framework performs well across multiple public external benchmarks on both prompt classification and response classification, it is trained using the same WGTrain dataset as WildGuard. As noted in the WildGuard study, much of their data is synthetic, which may not perfectly represent natural human inputs encountered in real-world scenarios. Further experiments are required to verify its generalization across other datasets.

Token-Level Classification: Currently, there is no large-scale, open-source dataset available for token-level safety classification evaluation. We visualized token-level predictions to demonstrate the effectiveness of ShieldHead and conducted token-level data annotation based on beavertails as a means of assessing the classification capabilities of ShieldHead. However, the dataset's limited size means it cannot cover all possible real-world scenarios.

Multiclass Unsafe Content Classification: In practical applications, it may be necessary to determine specific categories of unsafe content. This study focuses on binary classification between safe and unsafe. Although the proposed ShieldHead

Table 5: **Ablation Study.** Comparisons of F1 scores (%) between ShieldHead (with Llama3.1-8B as base model) with and without designed modules on prompt and response safety classification in existing public benchmarks.

Model	Prompt Harmfulness (F1)				Response Harmfulness (F1)		
	XSTest	OpenAI Mod	ToxicChat	Avg.	BeaverTails	S-RLHF	Avg.
w/o Sentence Loss	91.8	71.3	58.1	73.7	73.5	81.6	77.6
w/o Prompt Loss	92.0	70.9	59.3	74.1	76.2	85.9	81.1
w/o Response Loss	**95.4**	74.8	64.0	78.1	74.2	84.0	79.1
w/o Token Loss	94.1	74.8	61.3	76.7	73.7	86.2	80.0
w/o Label Disambi.	93.0	72.0	62.3	75.8	76.2	83.1	79.7
ShieldHead (Our)	95.1	**74.9**	**64.3**	**78.1**	**76.7**	**86.6**	**81.7**

with prototype-based label disambiguation technique supports multi-classification, further exploration is deferred to future work.

A.2 More Details about ShieldHead

A.2.1 Implementation Details

We initially assessed our method against state-of-the-art LLM-based and online API approaches across multiple public benchmarks, as shown in Section 3.1. Our experiments utilized F1 scores as the evaluation metric for all benchmarks. To ensure fairness, we used consistent data splits for all evaluations. We use the LLaMA-Factory (Zheng et al., 2024) codebase for model training, with 4 A100 80GB GPUs (8 A100 80GB GPUs for Llama3.1-70B) using a total batch size of 8, a max sequence length of 4096. During training, the AdamW optimizer was employed with a learning rate of 1×10^{-5} and a warmup ratio of 0.1, no weight decay. ShieldHead was trained for one epoch over the training set. Token-level labels were initialized to their corresponding sentence-level labels, and prototypes were initialized to zero. We implemented a warm-up strategy for label disambiguation, during which only the prototypes were updated, and the token-level labels remained unchanged for the initial 2000 steps. The duration of training is contingent upon the base model employed. Specifically, when utilizing the Llama-3.1-8B-Instruct model as base model, the training process can be completed in under one hour.

We utilize WILDGUARDTRAIN (WGTRAIN) (Han et al., 2024) as training dataset, which provides the necessary sentence-level labels for both prompts and responses. WGTRAIN contains a total of 86,759 entries, consisting of 48,783 standalone prompts and 37,976 prompt-response pairs. Our focus is on the 37,976 pairs, classified as follows: 16,647 pairs with both safe prompts and responses, 12,946 with unsafe prompts but safe responses, 27 with safe prompts and unsafe responses, and 8,356 where both are unsafe. We designate 10% of these pairs as a validation set.

A.2.2 More Results for Ablation Study

In Table 5 and Table 6, we report full evaluation results of ShieldHead with and without designed modules and with different number of layers across all public benchmarks. Introducing token-level labels without label disambiguation results in decreased performance compared to solely using sentence-level labels. This method falls short by 2.3% and 2.0% in average F1 scores for prompt and response classification, respectively, when compared to the complete ShieldHead. Additionally, it experiences declines of 0.9% and 0.3% compared to the ShieldHead variant without token-level co-supervision. This result shows that the proposed label disambiguation module effectively mitigates the noise introduced by token-level supervision, maximizing its benefits and ultimately enhancing model performance.

Benefit of using moving-average during training. Moving average and simple average do not perform the same function. We conducted an ablation experiment to compare their effects, as shown in Table 7. There are two potential ways to update prototypes using the simple average of hidden features: (1) by calculating a simple average over all tokens at the start, or (2) by calculating it per batch during training. In the first case, when token-level labels are initially assigned based on noisy sentence-level labels, the simple average produces biased prototypes, which prevents the pro-

Table 6: **Ablation Study.** Comparisons of F1 scores (%) between ShieldHead (with Llama3.1-8B as base model) with different number of layers on prompt and response safety classification in existing public benchmarks.

Layer	Trainable Parm. (%)	Prompt Harmfulness (F1)				Response Harmfulness (F1)		
		XSTest	OpenAI Mod	ToxicChat	Avg.	BeaverTails	S-RLHF	Avg.
1	0.416	94.9	73.9	60.6	76.5	74.5	82.9	78.7
2	0.832	94.9	75.8	64.5	78.4	76.4	87.8	82.1
3	1.136	95.1	74.9	64.3	78.1	76.7	86.6	81.7
4	1.745	95.9	76.2	66.8	79.6	79.2	88.3	83.8
5	3.284	**96.6**	77.2	69.5	81.1	79.0	**88.4**	**83.7**
6	3.721	96.4	**77.6**	**71.3**	**81.8**	**79.7**	87.3	83.5

Table 7: **Ablation Study.** Comparisons of F1 scores (%) between ShieldHead (with Llama3.1-8B as base model) and simple average methods on prompt and response safety classification in existing public benchmarks.

Model	Prompt Harmfulness (F1)				Response Harmfulness (F1)		
	XSTest	OpenAI Mod	ToxicChat	Avg.	BeaverTails	S-RLHF	Avg.
Simple average (prototype remains constant)	93.0	72.6	62.5	76.0	76.5	83.0	79.8
Simple average per batch (γ=0)	93.2	74.1	63.0	76.8	75.9	85.5	80.7
ShieldHead (Our)	95.1	**74.9**	**64.3**	**78.1**	**76.7**	**86.6**	**81.7**

totypes from reducing the noise of token-level labels. ShieldHead outperforms the first case model by an average F1-score of 2.1% and 1.9% across prompt and response benchmarks, respectively. In the second case, calculating the simple average per batch can still introduce batch-specific biases, as prototypes are calculated independently for each batch. This prevents the prototypes from accumulating information over time, leading to less stable or reliable prototypes. ShieldHead outperforms the second case model by an average F1-score of 1.3% and 1.0% across prompt and response benchmarks, respectively.

Benefit of reducing momentum coefficients during training. To demonstrate the effect of the moving average update and the impact of reducing γ and σ during training, we included an additional ablation study as shown in Table 8 and Table 9. γ and σ represent the update rates for the prototypes and token-level labels, respectively. Overall, the results show that reducing γ and σ yields better performance than using fixed values. The utilization of a moving average yields superior performance due to the inherent unreliability of the classifier during the initial stages of training. During this phase, the selection of top-k tokens for updating the prototypes may not be fully accurate. Therefore, a conservative update approach is used initially to prevent noise, with the update rate gradually increasing as the classifier becomes more reliable.

Note that setting γ=0 corresponds to using a simple average for each batch (when calculating the prototype in each batch during training, only the current batch is considered, without taking historical information into account). The moving average approach outperforms the simple average by an average F1-score of 1.3% and 1.0% across prompt and response benchmarks, respectively. Similarly, setting σ=0 corresponds to an ablation where the token-level labels are used without the label disambiguation algorithm (since the labels are not updated).

A.2.3 Performance in Safety-Critical Scenarios

Identifying unsafe instances as safe (false negatives, with "safe" considered negative and "unsafe" considered positive) is crucial in safety-critical scenarios. In our model training, we took into account and calculated a variety of metrics, including the False Negative Rate (FNR), although our primary focus was on the F1 score. The FNR results are

Table 8: **Ablation Study.** Comparisons of F1 scores (%) between ShieldHead (with Llama3.1-8B as base model) with different γ values on prompt and response safety classification in existing public benchmarks.

Model	Prompt Harmfulness (F1)				Response Harmfulness (F1)		
	XSTest	OpenAI Mod	ToxicChat	Avg.	BeaverTails	S-RLHF	Avg.
γ=0.99	93.0	72.2	62.1	75.8	76.3	83.5	79.9
γ=0.95	**95.2**	73.5	63.9	77.5	76.3	85.7	81.0
γ=0.5	93.3	73.6	62.6	76.5	76.0	85.8	80.9
γ=0 (simple average)	93.2	74.1	63.0	76.8	75.9	85.5	80.7
ShieldHead (Our)	95.1	**74.9**	**64.3**	**78.1**	**76.7**	**86.6**	**81.7**

Table 9: **Ablation Study.** Comparisons of F1 scores (%) between ShieldHead (with Llama3.1-8B as base model) with different σ values on prompt and response safety classification in existing public benchmarks.

Model	Prompt Harmfulness (F1)				Response Harmfulness (F1)		
	XSTest	OpenAI Mod	ToxicChat	Avg.	BeaverTails	S-RLHF	Avg.
σ=1 (w/o Label Disambi.)	93.0	72.0	62.3	75.8	76.2	83.1	79.7
σ=0.98	93.3	72.2	61.9	75.8	75.6	84.5	80.1
σ=0.5	94.5	72.8	64.1	77.1	**76.7**	85.8	81.3
ShieldHead (Our)	95.1	**74.9**	**64.3**	**78.1**	**76.7**	**86.6**	**81.7**

shown in Table 10. ShieldHead on ToxicChat exhibits a relatively high FNR, as the number of positive samples (unsafe) is much smaller than the number of negative samples (safe), with a ratio of approximately 1:10, which increases the difficulty of the task. Nonetheless, the model demonstrated generally strong performance across different base model sizes, with the lowest FNRs of 0.14 and 0.13 on the prompt and response benchmarks, respectively, showing the potential of ShieldHead in safety-critical scenarios.

A.3 Public Benchmarks for Evaluations

XSTest Dataset (Röttger et al., 2023): This evaluation dataset consists of 450 prompts, including 250 safe prompts across ten prompt types that well-calibrated models should not refuse to comply with, and 200 unsafe prompts as contrasts that, for most LLM applications, should be refused. In our evaluation, prompts which LLM applications should refuse to response was labeled as unsafe.

OpenAI Moderation Dataset (Markov et al., 2023): This evaluation dataset consists of 1,680 prompts, each labeled according to eight distinct risk categories. In our evaluation, all prompts were utilized, and every risk category was labeled as unsafe.

ToxicChat Test Set (Lin et al., 2023): This dataset includes a test split containing harm labels for real-world user requests sourced from the Vicuna online demo. It features 2,853 prompts, annotated by humans, with labels indicating prompt harmfulness and whether a prompt is adversarial. In our evaluation, any prompt marked as harmful is labeled unsafe.

BeaverTails Test Set (Jiaming Ji and Yang., 2024): This test set is part of a manually-annotated dataset, comprising 3,021 prompt-response pairs with harm labels. The prompts are derived from HH-RLHF red teaming (Yuntao Bai, 2022) splits and other sources, while responses are generated using the Alpaca-7B model, followed by human annotations. The dataset spans 14 harm categories, all of which were labeled as unsafe in our evaluation.

SafeRLHF Test Set (Dai et al., 2024): This dataset is a test split from a preference dataset featuring prompts followed by two responses and a comparison of these responses. Sharing prompts with the BeaverTails dataset, it emphasizes comparison data with manual human preference annotations. For efficiency, we selected the subset with Alpaca3-8B responses, comprising a total of 2,327 prompt-response pairs.

Table 10: FNR scores of ShieldHead with different base models on prompt and response safety classification in existing public benchmarks.

Model	Prompt Harmfulness (FNR)				Response Harmfulness (FNR)		
	XSTest	OpenAI Mod	ToxicChat	Avg.	BeaverTails	S-RLHF	Avg.
Gemma2-2B	0.03	0.17	0.37	0.19	0.14	0.09	0.12
Gemma2-9B	0.03	0.17	0.32	0.16	0.15	0.09	0.12
Gemma2-27B	0.04	0.14	0.35	0.14	0.17	0.09	0.13

A.4 Existing Safety Risks Moderation Tools

We compare our ShieldHead against several methods: OpenAI Moderation API, LlamaGuard3 (Llama Team, 2024), WildGuard (Han et al., 2024), ShieldGemma-9B (Zeng et al., 2024) and GPT-4 (Josh Achiam, 2023). For GPT-4, we utilize the OpenAI API (model=gpt-4-0613) with the prompt used in WildGuard (Han et al., 2024). For other methods, we simply adopt their default prompt and evaluation setting.

A.4.1 LLM-based safety classification model

Llama-Guard (Inan et al., 2023): Llama-Guard is an instruction-tuned model based on Llama-2 7B, aimed at classifying harms within both input prompts and model responses. It supports the classification of 6 distinct types of safety risks. The model is trained using prompts from the Anthropic Red Teaming dataset (Perez et al., 2022), supplemented with proprietary responses and labels that indicate the harmfulness of prompts and responses.

Llama-Guard2 (Team, 2024b): Llama-Guard2 is an advanced version of Llama-Guard, developed using Llama-3 8B. It defines and supports the classification of 11 safety risk types. Building based on the Llama-Guard training set, it focuses on challenging examples by augmenting existing prompts with flipped labels. This approach aims to refine the model's ability to identify complex safety risks accurately.

Llama-Guard3 (Llama Team, 2024): Llama-Guard3 is an instruction-tuned model based on Llama-3.1 8B, supporting 14 types of safety risk classifications. Its training set extends from Llama-Guard, with an emphasis on multilingual capabilities and tool usage. It incorporates additional human and synthetically generated data to enhance adaptability and precision across diverse contexts.

ShieldGemma (Zeng et al., 2024): ShieldGemma is an instruction-tuned model based on Gemma2, available in multiple parameter scales (2B, 9B, and 27B). Its training dataset is derived from Anthropic/hh-rlhf (Yuntao Bai, 2022) and is downsampled to 15,000 samples to align with Llama-Guard.

WildGuard (Han et al., 2024): WildGuard is an instruction-tuned model based on Mistral-7b-v0.3, supporting multi-task recognition including Prompt Harm, Response Harm, and Refusal Detection. Its training relies on the WILDGUARDTRAIN (WGTRAIN) dataset, consisting of a total of 86,759 entries, with 48,783 standalone prompts and 37,976 prompt-response pairs. The training data for WildGuard is publicly accessible, promoting transparency and facilitating further research.

A.4.2 LLM-based safety classification model

GPT-4 Classification (Josh Achiam, 2023): We follow the prompts used for GPT-4 classification in WildGuard (Han et al., 2024). In WildGuard, a search of several prompt variants is conducted, including providing additional guidelines and prompting chain-of-thought reasoning, finding that this prompt performs the best overall across evaluations. We use gpt-4-0613 for all GPT-4 classification.

A.5 Annotation method on token-level classification

A.5.1 Annotation agreement

As to the annotation agreement, we followed the best practice from prior studies, such as ToxicChat (Lin et al., 2023) and Beavertails (Jiaming Ji and Yang., 2024), adapting their sentence-level annotation method to our token-level annotation: Two independent annotators were tasked with labeling each token within the sentence. Any disagreements between the annotators were resolved through discussion to reach mutual agreement on all tokens.

A.5.2 Intuition for token-level annotation

Most work in text generation considers all preceding tokens when performing next token prediction (Radford and Narasimhan, 2018; Brown et al., 2020), and our work aims to classify safety of the responses generated by these text generators. Therefore, intuitively, we follow this manner.

Hence, in this paper, we adopt the cumulative effect of GPT-like models (Radford and Narasimhan, 2018; Radford et al., 2019; Brown et al., 2020). Instead of directly labeling each token individually (treat them independently), we label tokens taking into account all the preceding context. Our safety risk classification is derived from the conditional probability $p(s_n|t_1, ..., t_n)$, where t_1 to t_n represent the first to the nth token, and s_n denotes the safety risk classification of the nth token.

The manual annotation method is used only for the evaluation set, whereas token-level labels used during training are obtained through the prototype-based disambiguation method. In our experiments, following the cumulative effect of GPT-like models, we use the safety classification of the last token in a sentence as the prediction for the sentence. We achieve state-of-the-art performance in sentence-level safety classification. This also demonstrates that the cumulative effect is applicable in the context of safety classification in our work.

A.5.3 Existing token-level annotation methods

In addition to its application in safety risk classification, as proposed in this paper, accumulative token-level annotation is also applied in other tasks. For example, RIPA (Choi et al., 2023) proposes that once the first hallucinated token is detected, all subsequent generated tokens should be labeled as hallucinated. NADO (Meng et al., 2022) introduced the concept that the occurrence of hallucination in a token is related to its prefix from the perspective of the conditional distribution formula. Our method, NADO (Meng et al., 2022), and RIPA (Choi et al., 2023), leverages the cumulative effect of GPT-like models, where the conditional probability of subsequent tokens is influenced by the preceding tokens. Similar to our annotation method, these approaches also use accumulative annotation. However, their objectives differ from ours. For example, RIPA constructs a reward based on these accumulative token-level labels. Besides, Fast-detectgpt (Bao et al., 2023) use token-level conditional probability curvature as a cumulative effect to determine whether the text is AI-generated. Fast-detectgpt (Bao et al., 2023) and DetectGPT (Mitchell et al., 2023) label the first few tokens as "human-written," and all tokens following the first AI-generated token as "AI-generated".

In summary, accumulative annotation has been employed in previous work as well, further validating the effectiveness of our annotation method.

A.6 Example for GPT-4 Prompts

You are a classifier for analyzing interactions between humans and LLMs. Given a request from a human user, determine whether the human user's request is harmful. Please just response with harmful or unharmful without any other words. Here are two examples: Prompt: How can I rob the bank? Prompt harmfulness: harmful Prompt: What's the weather like today? Prompt harmfulness: unharmful Following is the request and response for you to classify: Prompt: {prompt} Prompt harmfulness:

Table 11: Instruction prompt to use a few WGTRAIN samples as a few-shot in-context learning examples for sentence-level safety classification.

You are a classifier for analyzing interactions between humans and LLMs. Given a request from a human user or a response from an LLM assistant, determine whether the sentence is harmful word by word. Please respond solely with 0 for non-harmful content and 1 for harmful content, without any additional commentary. Here are three examples: Sentence: How can I rob the bank? Sentence harmfulness: 0, 0, 0, 1, 1, 1 Sentence: To have cyber sex, you'll need to find someone who is interested in virtual sex and get comfortable being sexual online. Start by slowly getting sexual and increasingly more intimate through suggestive conversations. Sentence harmfulness: 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 Following is the request and response for you to classify: Sentence: {prompt} Sentence harmfulness:

Table 12: Instruction prompt to use a few WGTRAIN samples as a few-shot in-context learning examples for word-level safety classification.