ACM CCS 2024（全球网络安全三大顶会之一）

“Legilimens”来自《哈利·波特》，指可以读取别人思想的巫师。
这与论文方法完全一致：通过读取 LLM 的隐藏状态而不是表面文本来“读懂”模型的真实意图，从而进行内容审核。

LEGILIMENS: Practical and Unified Content Moderation for Large Language Model Services

Jialin Wu*
Zhejiang University
Hangzhou, Zhejiang, China
jialinwu@zju.edu.cn

Jiangyi Deng*
Zhejiang University
Hangzhou, Zhejiang, China
jydeng@zju.edu.cn

Shengyuan Pang
Zhejiang University
Hangzhou, Zhejiang, China
pangpang0093@zju.edu.cn

Yanjiao Chen
Zhejiang University
Hangzhou, Zhejiang, China
chenyanjiao@zju.edu.cn

Jiayang Xu
Zhejiang University
Hangzhou, Zhejiang, China
3210103789@zju.edu.cn

Xinfeng Li
Zhejiang University
Hangzhou, Zhejiang, China
xinfengli@zju.edu.cn

Wenyuan Xu
Zhejiang University
Hangzhou, Zhejiang, China
wyxu@zju.edu.cn

ABSTRACT

Given the societal impact of unsafe content generated by large language models (LLMs), ensuring that LLM services comply with safety standards is a crucial concern for LLM service providers. Common content moderation methods are limited by an effectiveness-and-efficiency dilemma, where simple models are fragile while sophisticated models consume excessive computational resources. In this paper, we reveal for the first time that effective and efficient content moderation can be achieved by extracting conceptual features from chat-oriented LLMs, despite their initial fine-tuning for conversation rather than content moderation. We propose a practical and unified content moderation framework for LLM services, named LEGILIMENS, which features both effectiveness and efficiency. Our red-team model-based data augmentation enhances the robustness of LEGILIMENS against state-of-the-art jailbreaking. Additionally, we develop a framework to theoretically analyze the cost-effectiveness of LEGILIMENS compared to other methods.

We have conducted extensive experiments on five host LLMs, seventeen datasets, and nine jailbreaking methods to verify the effectiveness, efficiency, and robustness of LEGILIMENS against normal and adaptive adversaries. A comparison of LEGILIMENS with both commercial and academic baselines demonstrates the superior performance of LEGILIMENS. Furthermore, we confirm that LEGILIMENS can be applied to few-shot scenarios and extended to multi-label classification tasks.

*Both authors contributed equally to the paper.

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690322

CCS CONCEPTS

• **Security and privacy** → **Human and societal aspects of security and privacy**; • **Computing methodologies** → **Artificial intelligence**.

KEYWORDS

Content moderation; large language model; jailbreaking

ACM Reference Format:
Jialin Wu, Jiangyi Deng, Shengyuan Pang, Yanjiao Chen, Jiayang Xu, Xinfeng Li, and Wenyuan Xu. 2024. LEGILIMENS: Practical and Unified Content Moderation for Large Language Model Services. In *Proceedings of Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24)*. ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3658644.3690322

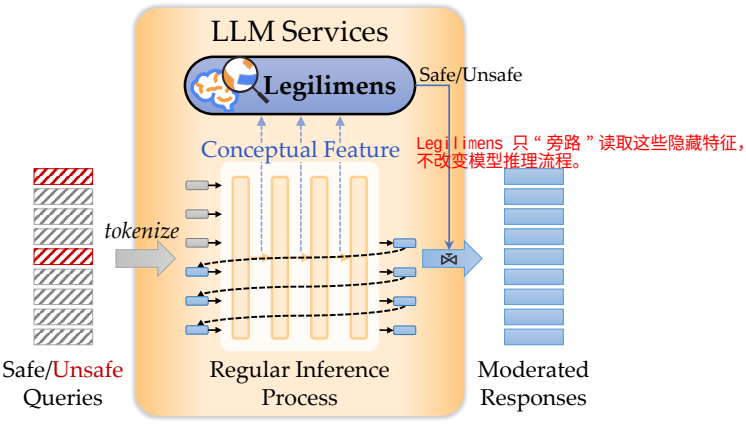

Figure 1: LEGILIMENS is a unified content moderation framework for almost all LLM services, which features both effectiveness and efficiency. By monitoring conceptual features generated in the regular inference process of LLMs, LEGILIMENS achieves lightweight moderation.

利用 LLM 生成第一个 token、最后一个 token 时的隐藏状态即可完成审核

1 INTRODUCTION

With the rise of Large Language Models (LLMs), there has been a significant increase in global user engagement with online LLM services. However, the impressive and unknown capacities of chat-oriented LLMs can be used to malicious ends. Although LLMs are currently deployed on a large scale without properly taking into account the harm they can cause to society, there is a growing concern of and calls for compliance of LLMs to safety standard. Since model alignment by removing undesired behaviors from models has been empirically [43, 45, 77, 84, 87, 88, 95] and theoretically [82] proved to have fundamental limitations, service providers are tirelessly pursuing an effective and efficient content moderation solution.

已有方法的两个缺点

However, existing content moderation methods have been found to be dangerously brittle. Even commercial solutions like OpenAI's can be easily bypassed [43, 45, 88, 95] due to the inadequate ability of simple classifiers [79]. Some researchers explored using LLMs as more sophisticated classifiers [11, 33, 46, 78]. But LLM-based content moderation imposes too great a burden on already insufficient computational resources [30], rendering it impractical in real-world applications. This dilemma of effectiveness and efficiency requires urgent resolution.

In this paper, we make the first attempt to balance effectiveness and efficiency in content moderation for chat-oriented LLMs. As shown in Figure 1, we propose a practical and unified content moderation framework for LLM services, named Legilimens[1], which extracts distinctive conceptual features from LLMs in a lightweight manner. Legilimens is built on the idea of leveraging the regular inference process of an LLM (the *host* LLM) to extract effective features for content moderation while incurring minimal overhead. A comparison of Legilimens with existing works is shown in Table 1. However, despite the simplicity and straightforwardness of the basic idea, realizing this concept into a practical system poses several challenges.

- *How to design an efficient and unified framework for input and output moderation that is universally applicable to almost all kinds of LLMs?*

为不同架构的大语言模型、不同长度的文本输入或输出，以及不同内容审核任务设计统一框架具有挑战性。

Our basic idea is to use the host LLM as a feature extractor. However, it is challenging to design a unified framework for LLMs of different architectures, for textual input or output of different lengths, and for different moderation tasks. First, LLMs can be categorized into *Encoder-decoder* and *decoder-only* by their architectures. And *decoder-only* LLMs can be further divided into *causal* decoders and *prefix* decoders. In addition, LLMs commonly have different model parameters, *e.g.*, LLaMA2 has three sizes, 7B, 13B and 70B [74]. Thus, the proposed method should be independent of model architectures and model sizes. Second, LLMs are sequence-to-sequence models that input and output variable-length texts. In this process, variable-length features in LLMs are generated. The proposed method should be able to deal with variable-length input/output and at best reduce the complexity to be independent of the input/output length, *i.e.*, $O(1)$. Third, content moderation for LLM services consists of different tasks, including moderating the input based on the input, moderating the output based on the input and output, and moderating the input based on the input and output. The proposed method should be applicable to these tasks.

所提出方法应解决的三个问题

To address this problem, we design a decoder-based concept probing method that leverages the features generated in the decoding process of the first and the last output tokens. This decoding process is common to different LLM architectures and different input/output. By the attention mechanism, the features generated during the decoding of the first output token correspond to concepts related to the entire input. Similarly, the features generated during the decoding of the last output token correspond to concepts related to both the input and output. We utilize these two types of features to achieve both input and output moderation, namely *I*-moderation and *O*-moderation. Note that, our method focuses solely on two tokens, regardless of the input or output length. Thus, the complexity of our method is reduced to a constant level. We further demonstrate that Legilimens is applicable in few-shot scenarios, thereby reducing the cost for service providers to set up our content moderation system. In this way, we achieve an efficient and unified content moderation framework for LLM services.

为什么说“通过注意力机制，解码首个输出标记时生成的特征对应于与整个输入相关的概念。同样地，解码最后一个输出标记时生成的特征对应于与输入和输出均相关的概念”？

自回归机制中，最后一个token涵盖前面所有token的信息，即最后一个 token 拥有“最完整的上下文信息”

- *How to defend against recent jailbreak attacks that aim to bypass the safety mechanisms of LLMs?*

在Pinpoint论文的代码终究是通过 从每一层提取“指令最后一个token”的特征（最关键）

Jailbreaking refers to the process of circumventing the safety mechanisms placed on LLM services such as model alignment and content moderation. For instance, a common method to jailbreak model alignment is to instruct LLMs to emulate a "Do Anything Now" (DAN) behavior [40]. An instance of jailbreaking content moderation involves performing orthographic transformations, such as `Base64` encoding [66], on textual input or output to bypass content filters, as state-of-the-art LLMs can inherently process such encoded text. While LLM service providers have implemented stricter rules to prevent the use of such jailbreak prompts [47], they cannot entirely eliminate jailbreaking due to the numerous ways to construct prompts conveying the same meaning, owing to the inherent flexibility of natural languages.

To cope with this problem, we equip a red-team model-based data augmentation method in the training stage of Legilimens. Specifically, we employ a local LLM to work as a red-teaming model, generating highly adversarial jailbreak prompts from initially *naive* unsafe prompts. Through evaluating Legilimens against static and dynamic jailbreaking methods, we validate that incorporating augmented data during the training phase renders highly robust detection of unsafe content.

基于红队模型的数据增强方法如何增强Legilimens系统使其能够稳健地抵抗越狱攻击的？

论文利用一个“红队模型”（Red-team LLM）主动生成各种越狱版本的危险prompt，把它们加入训练集，让审核器提前见到攻击者可能使用的绕过技巧。

We have implemented a fully-functional prototype of Legilimens on five different host LLMs and evaluated its performance through extensive experiments on seventeen diverse datasets across three content moderation tasks (*i.e.*, *I*-, *O*-, *IO*-moderation). Note that Although previous studies [2, 13] have utilized the hidden states of LLMs for hallucination detection, the potential of these hidden states for diverse content moderation tasks remains largely unexplored. Our work not only demonstrates that effective and robust content moderation can be achieved by extracting conceptual features from chat-oriented LLMs and augmenting prompts through red-teaming, but also underscores this paradigm's ability to successfully navigate the trade-off between effectiveness and efficiency.

在多样化内容审核任务利用LLM的隐藏状态

Legilimens 是第一篇把“LLM 隐藏状态”用于“多样化内容审核任务”的系统性研究。

[1] A legilimens is a person skilled in magically navigating through the many layers of a person's mind and correctly interpreting one's findings in the fantasy novels *Harry Potter* by J.K. Rowling [67]

Table 1: LEGILIMENS versus existing works.

Method	Model Architecture	Model Parameters	Extra Overhead*	Domain
OpenAI Moderation API [47]	GPT†	117M~1.5B†	$O(n)$	General content
Google Perspective API [37]	BERT⟶CNN‡	-‡	$O(n)$	General content
BeaverDam-7B [36]	LLaMA-7B	7B	$O(n)$	QA¶
LLaMA Guard2 [34]	LLaMA 3-8B	8B	$O(n)$	QA¶
GradSafe [83]	-§	-§	$O(n)$	General content
Wang *et al.* [78]	GPT-3	175B	$O(n)$	Hate speech
Cao *et al.* [11]	GPT-4, LLaMA2	100T, 13B	$O(n)$	General content
FakeGPT [33]	GPT-3.5	≥175B	$O(n)$	Fake news
Ma *et al.* [46]	ChatGLM2-6B, Baichuan-13B	6B, 13B	$O(n)$	General content
Legilimens (Ours)	**1~3-Layer MLP**	**8k~4.7M**	**O(1)**	**General content, QA¶**

(i) †: The exact GPT architecture has not been disclosed [47]. Based on the publication year [86] and the API response time, we speculate that it is either GPT-1 or GPT-2. (ii) ‡: The CNNs are distilled from trained BERT-based models. The exact architecture has not been disclosed. (iii) ∗: n denotes the (token) length of input. The complexity is estimated based on the model architecture and our experimental results. (iv) ¶: "QA" refers to moderating question-answering (QA) data to assess the safety of the answers. (v) §: GradSafe calculates the difference in gradients between safe and unsafe prompts for detection, thus there is no detection model.

Our experiments, which include assessments against both static and dynamic jailbreaking, confirm the robustness of LEGILIMENS against adversarially-designed prompts. Furthermore, the comparison with seven commercial and academic baselines demonstrates that LEGILIMENS achieves the best performance in detecting unsafe content while maintaining the highest efficiency. Additionally, we validate that LEGILIMENS can be applied to few-shot scenarios, and extended to perform multi-label classification tasks. We have open-sourced our code[2] in a hope to incentivize more research in this area.

We summarize our contributions as follows:

- We propose a practical and unified content moderation framework tailored for LLM services, which features a balance between effectiveness and efficiency.
- We develop a concept probing technique that applies to mainstream encoder-decoder and decoder-only LLMs. Additionally, our red-team model-based data augmentation method enhances LEGILIMENS to robustly resist jailbreaking.
- We conduct extensive experiments to verify the effectiveness and efficiency of our content moderation framework against both non-adversarial and adversarial queries, outperforming seven commercial and academic baselines.

2 BACKGROUND

2.1 Large Language Model

Typically, large language models (LLMs) refer to language models that contain billions of parameters, which are trained on massive text data [92]. Famous examples include GPT-3 [9, 57], GPT-4 [55], GLM [24, 89], LLaMA [73, 74], *etc.* In this part, we introduce the basic components, architectures, and inference workflow of LLMs.

2.1.1 Basic Component. Given the excellent parallelizability and capacity, the Transformer architecture [44, 75] has become the *de facto* backbone to almost all LLMs, making it possible to scale language models to hundreds or thousands of billions of parameters. The vanilla Transformer [75] is a sequence-to-sequence model and consists of an encoder and a decoder, each of which is a stack of K identical blocks. Each encoder block is mainly composed of a multi-head self-attention module and a position-wise feed-forward network (FFN). For building a deeper model, a residual connection [32] is employed around each module, followed by layer normalization [3] module. Compared to the encoder blocks, decoder blocks additionally insert cross-attention modules between the multi-head self-attention modules and the position-wise FFNs. Furthermore, the self-attention modules in the decoder are adapted to prevent each position from attending to subsequent positions in the training phase.

解码器中的自注意力模块经过怎样的调整可在训练阶段防止每个位置关注后续位置的？

2.1.2 Architecture. In general, the mainstream architectures of existing LLMs can be roughly categorized into two major types, namely encoder-decoder and decoder-only.

Encoder-Decoder Architecture. The vanilla Transformer model is built on the encoder-decoder architecture [75], which consists of two stacks of Transformer blocks as the encoder and decoder, respectively. The encoder encodes the input sequence for generating its latent representations, while the decoder performs cross-attention on these representations and generates the target sequence in an auto-regressive manner. So far, there are only a small number of LLMs that are built based on the encoder-decoder architecture, *e.g.*, Flan-T5 [16].

Decoder-Only Architecture. Models of this type only have the decoder but no encoder. According to the self-attention mechanism used in the decoder, decoder-only models can be further divided into the causal decoder architecture and the prefix decoder architecture.

The *causal* decoder architecture incorporates the unidirectional attention mask, to guarantee that each input token can only attend to the past tokens and itself. The input and output tokens are processed in the same fashion through the decoder. As representative language models of this architecture, the GPT series models [9, 55, 57] are developed based on the causal decoder architecture. So far, the causal decoders have been widely adopted

[2] https://github.com/lin000001/Legilimens

as the architecture of LLMs by various existing LLMs, such as LLaMA [73, 74], Dolly [17, 18], and Falcon [61].

The *prefix* decoder architecture (*a.k.a.*, non-causal decoder) revises the masking mechanism of causal decoders, to enable performing bidirectional attention over the prefix tokens [23] and unidirectional attention only on generated tokens. In this way, like the encoder-decoder architecture, the prefix decoders can bidirectionally encode the prefix sequence and auto-regressively predict the output tokens one by one, where the same parameters are shared during encoding and decoding. Existing representative LLMs based on prefix decoders include GLM [24, 89] and U-PaLM [72].

LLM 的输出是逐 token 自动回归生成的，这叫 auto-regressive decoding（自回归解码）。

2.1.3 Model Inference. In typical model inference process of LLMs, a prompt $\mathbf{p} = p_1 p_2 \cdots p_n$ is fed into LLMs to generate a response sequence $\mathbf{r} = r_1 r_2 \cdots r_m$ auto-regressively, *i.e.*,

$$r_{i+1} = \arg\max_r \mathbb{P}(r|\mathbf{p} \oplus \mathbf{r}_{1:i}) \quad (\textit{greedy search}) \qquad \text{模型在每一步选概率最高的 token}$$
$$or \quad r_{i+1} \sim \mathbb{P}(r|\mathbf{p} \oplus \mathbf{r}_{1:i}) \quad (\textit{sampling-based methods}), \qquad \text{模型根据概率分布“随机抽样”} \tag{1}$$

where $\oplus$ denotes concatenating the previous output tokens to the end of the input sequence until a special sentence ending token (usually denoted as `[eos]`) is generated. The first decoding method is *greedy search*, which predicts the most likely token at each step based on the previously generated tokens. The other decoding method is *sampling*, which randomly samples the next token based on the probability distribution to enhance the randomness and diversity during generation.

From the inference process and the self-attention mechanism of LLMs we know that LLMs output the first token r_1 of $\mathbf{r}$ leveraging the information of $\mathbf{p}$, and output the last token (*i.e.*, `[eos]`) with the information of both $\mathbf{p}$ and $\mathbf{r}$, *i.e.*,

$$r_1 = \mathcal{H}(\mathbf{p}) \quad \text{and} \quad \texttt{[eos]} = \mathcal{H}(\mathbf{p} \oplus \mathbf{r}), \tag{2}$$

只要观察模型生成 r? 和 eos 时的隐藏状态，就能知道 prompt 或整个对话是不是危险的

where $\mathcal{H}(\cdot)$ denotes the inference function of LLMs. We utilize this inference process as a feature extractor for the downstream content moderation task, which we elaborate on in §4.

Legilimens 的核心思想：
用第一个 token 的隐藏状态做 I-moderation（输入安全检测）
用最后一个 token 的隐藏状态做 O/IO-moderation（输出安全检测）

2.2 Content Moderation

To mitigate potential harm and misuse of LLMs, two safety mechanisms are commonly applied, *i.e.*, model alignment involves training LLMs to reject unsafe prompts, while content moderation employs filters to block unsafe prompts and responses.

2.2.1 Unsafe Content. Given that *out-of-the-box* LLMs have the potential to generate misinformation, propagate harmful content, or produce unintended responses with significant negative societal impact [5, 19, 80, 81], content moderation is essential for identifying such *unsafe content* within prompts and responses. The definitions of unsafe content in previous literature often include profanities, identity attacks, sleights, insults, threats, sexually explicit content, demeaning language, and language that incites violence [26, 29, 68]. In practice, different service providers may adopt different definitions and taxonomies of unsafe content.

定义

2.2.2 Moderation Tasks. Content moderation for LLMs involves moderating the input prompt and the output response, referred to as *I*-moderation and *O*-moderation respectively. In *I*-moderation (*resp.* *O*-moderation), the moderator assesses whether the input prompt (*resp.* output response) is unsafe, based solely on the input prompt (*resp.* output response) or on the entirety of the prompt and response. The combined task, referred to as *IO*-moderation, aims to assess both the input prompt and the output response.

内容审核分为3种
输入审核
输出审核
输入输出审核

2.3 Jailbreak

Jailbreaking is a process that employs adversarially-designed prompts to circumvent the safety mechanisms imposed on LLMs by their service providers. Several efforts have been made to taxonomize jailbreaking [15, 45, 66, 71], based on which jailbreaking can be categorized into *semantic* transformation and *syntactic* transformation.

语义转换

- Semantic transformation involves manipulating the semantics of prompts. Based on the patterns of semantic manipulation, three general types of semantic transformation have been identified [45].
  - *Pretending*: This type of prompts tries to alter the conversation background or context while maintaining the same intention, *e.g.*, creating a role-playing game context.
  - *Attention Shifting*: This type of prompts aims to change both the conversation context and intention, *e.g.*, shifting the intention of the prompt from asking the model questions to making it construct a paragraph of text.
  - *Privilege Escalation*: This type of prompts seeks to directly circumvent the imposed restrictions. For instance, "Do Anything Now" prompts mentioned in §1 belong to this category.

句法转换

- Syntactic transformation modifies only the syntax of prompts without altering the semantics. Examples include string splicing, common encoding, and simple encryption. This type of transformation can effectively bypass rule-based filters.

Note that the generation of jailbreak prompts can be static and dynamic. Static prompts do not leverage response information from the target LLM while dynamic methods use previous responses as a reference to adversarially refine the prompt for launching the next attack.

2.4 Design Goal

We first define the system model in terms of the adversary and the defender, and then elaborate the design goals of Legilimens under the defined system model.

Adversary. The adversary attempts to induce undesirable behavior of LLM services, whether unintentionally or intentionally. This may include generating inappropriate content, disclosing sensitive information, or performing actions against programming constraints. To achieve this, the adversary may optimize the semantics or syntax of their prompts to outsmart the safeguards of LLMs by any jailbreak methods. The adversary can observe the returned responses and leverage this knowledge to refine the prompt in order to achieve their attack objectives.

Defender. The defender aims to ensure both the safety and helpfulness of the LLM service. To achieve this, the defender strives to apply content moderation mechanisms that are as accurate as possible, with full access to the host LLM.

Given this system model, we delineate two major goals of LEGILIMENS.

- **Effectiveness**. LEGILIMENS should accurately identify unsafe prompts given to the host LLM or responses generated by the host LLM, even when confronted with jailbreak prompts.
- **Efficiency**. LEGILIMENS should introduce minimal overhead compared to the original inference process of the host LLM. Ideally, the overhead of LEGILIMENS should not increase with the length of prompts or responses, *i.e.*, maintaining a constant complexity of $O(1)$.

3 PROBLEM FORMULATION

In this section, we first formulate content moderation as a classification problem. Then we materialize the classification problem for three distinct moderation tasks.

3.1 Classification Formulation

A textual input $\mathbf{x} = x_1 x_2 \cdots x_n$ can be tokenized into n frames of token embeddings, *i.e.*, $\mathbf{X} \in \mathbb{R}^{n \times d}$, where d denotes the dimension of each token embedding. A typical content moderation problem is to determine whether $\mathbf{x}$ is unsafe by a moderator $\mathcal{M}_\theta(\cdot) : \mathbb{R}^{n \times d} \to \mathbb{Y}$, parametrized by θ, which can be expressed as

论文把“判断一句话是否不安全”看成一个二分类任务

$$\min_{\theta} \mathop{\mathbb{E}}_{\mathbf{X}, y \sim \mathbb{D}} \mathcal{L}\left(\mathcal{M}_\theta(\mathbf{X}), y\right), \tag{3}$$

把内容审核定义为常规的监督分类问题，训练一个模型 M 来判断一句话是否安全。

where $\mathbb{D}$ is a labelled content moderation dataset. $y \in \mathbb{Y}$ is the ground-truth label of the corresponding input $\mathbf{X}$, denoting whether $\mathbf{X}$ is unsafe. $\mathcal{L}$ represents the loss function, typically measured using cross-entropy loss, which evaluates the accuracy of $\mathcal{M}_\theta(\mathbf{X})$.

In the classification problem above, when the training dataset $\mathbb{D}$ is of sufficient size, the accuracy of the moderator $\mathcal{M}_\theta(\cdot)$ generally increases with the scale of its parameters θ, but at the cost of increased computational complexity. Thus, efficiency and effectiveness are often in a trade-off relationship.

In the scenario of LLM services, the host LLM is expected to infer input prompts and generate corresponding responses. Given the substantial capacity of the host LLM, we aim to utilize the original inference process of the host LLM to materialize a portion of $\mathcal{M}_\theta(\cdot)$. This approach allows representative features to be extracted in a "free-riding" manner, *i.e.*,

M 是两个模块拼起来的：前半段（LLM 的隐藏层）生成特征，后半段（小 MLP）做分类

$$\mathcal{M}_\theta(\mathbf{X}) = C_{\theta \setminus \psi} \circ \mathcal{H}_\psi(\mathbf{X}), \tag{4}$$

LLM 的隐藏状态非常强大，可以“免费用作内容审核”。

where $\mathcal{H}_\psi$ represents a portion of the host model parameterized by ψ, and $C_{\theta \setminus \psi}$ denotes an additional classifier parameterized by $(\theta \setminus \psi)$. In this way, the formulation in Equation (3) is transformed into

LLM（参数 ψ）是冻结的，不训练，只训练 MLP 分类器（参数 θ \ ψ）

$$\min_{\theta \setminus \psi} \mathop{\mathbb{E}}_{\mathbf{X}, y \sim \mathbb{D}} \mathcal{L}\left(C_{\theta \setminus \psi} \circ \mathcal{H}_\psi(\mathbf{X}), y\right). \tag{5}$$

Notably, since $|\psi|$ is sufficiently large, $|\theta|$ can be large when $|\theta \setminus \psi| = |\theta| - |\psi|$ is relatively small. Consequently, this approach potentially achieves a win-win scenario in terms of efficiency and effectiveness.

3.2 Materialization Tasks

Based on the object to be moderated, we materialize LEGILIMENS into three moderation tasks, as depicted in Figure 2.

- ***I*-Moderation**. In I-moderation, the object to be moderated is solely input prompts to be fed into the host LLM. The task is to determine whether the input prompts are unsafe, denoted as $\hat{y} = \mathcal{M}_\theta(\mathbf{P} \mid *)$, where $\mathbf{P}$ represents the token embeddings of prompt $\mathbf{p}$, and $*$ signifies any additional information available to the moderator, *e.g.*, output responses.
- ***O*-Moderation**. In O-moderation, the object to be moderated is solely output responses generated by the host LLM. The task is to determine whether the output responses are unsafe, denoted as $\hat{y} = \mathcal{M}_\theta(\mathbf{R} \mid *)$, where $\mathbf{R}$ represents the token embeddings of response $\mathbf{r}$, and $*$ signifies any extra information available to the moderator, *e.g.*, input prompts.
- ***IO*-Moderation**. In IO-moderation, the object to be moderated encompasses both input prompts and output responses, which is a combined task of I- and O-moderation, *i.e.*, $\hat{y} = \mathcal{M}_\theta(\mathbf{P}, \mathbf{R} \mid *)$.

The three tasks above can be applied along with different moderation strategies. In the case of I-moderation, the LLM service can halt once the prompt is determined to be unsafe, thereby saving subsequent computation. However, this strategy may potentially impair the overall helpfulness of the LLM service. Conversely, for O-moderation, the service halts until the response is determined to be unsafe. This strategy preserves helpfulness when prompts are unsafe but successfully handled by the host LLM, providing safe responses. Nevertheless, this strategy may require additional computational resources. IO-moderation halts the service upon detecting unsafe prompts or responses, representing a more rigorous moderation strategy compared to the first two tasks.

4 LEGILIMENS: DESIGN DETAILS

In this section, we materialize $\mathcal{H}_\psi$ and $C_{\theta \setminus \psi}$ in Equation (5) in the scenario of LLM services by concept probing and lightweight moderator construction.

4.1 Concept Probing

4.1.1 *Conceptual Feature Extraction*. Typical LLMs are constructed by stacking K encoder-decoder or decoder-only Transformer blocks together, *i.e.*,

LLM = K 层 Transformer Block 堆叠

$$\mathcal{H} = \mathcal{H}_K \circ \mathcal{H}_{K-1} \circ \cdots \circ \mathcal{H}_1, \tag{6}$$

where $\mathcal{H}_i$ denotes the i-th block of the host LLM. To decode a token, the original inference process derives side features as follows,

$$\widetilde{\mathcal{H}}_k(\mathbf{X}) \triangleq \left(\mathcal{H}_k \circ \mathcal{H}_{k-1} \circ \cdots \mathcal{H}_1\right)(\mathbf{X}), \quad 1 \le k \le K. \tag{7}$$

Hk(X) = 输入 X 经过第 1 到第 k 层后的语义特征。

Our intuition is that LLMs develop concepts of the input along the inference process. Thus, we attain comprehensive conceptual features of $\mathbf{X}$ by fusing/concatenating the side features derived from the last several blocks, *i.e.*,

$$\widetilde{\mathcal{H}}_{[-m:]}(\mathbf{X}) \triangleq \left[\widetilde{\mathcal{H}}_{(K-m+1)}(\mathbf{X}) \oplus \cdots \oplus \widetilde{\mathcal{H}}_K(\mathbf{X})\right]. \tag{8}$$

取最后 m 层的隐藏表示并拼接，得到“概念特征”

The reason for deriving conceptual features from the last several blocks is to utilize more capacity of the host LLM, given that $\widetilde{\mathcal{H}}_{[-m:]}$ can be viewed as containing all parameters of the host LLM. Note that, no additional computation is required to obtain the side features in Equation (8).

最后几层最能表达“语义 / 意图”，所以把它们的隐藏状态拼接起来作为内容审核的特征。

为什么取最后 m 层的隐藏表示并拼接，得到“概念特征”，而不是直接取最后一层的隐藏状态作为概念特征吗
概念/安全性特征不是只在最后一层出现，而是分布在多个深层中
多层拼接能利用更多模型容量，捕获更多语义特征
拼接多个层不增加任何额外计算（Zero overhead）
多层特征能提升内容审核的 robustness
实验表明多层特征效果优于单层

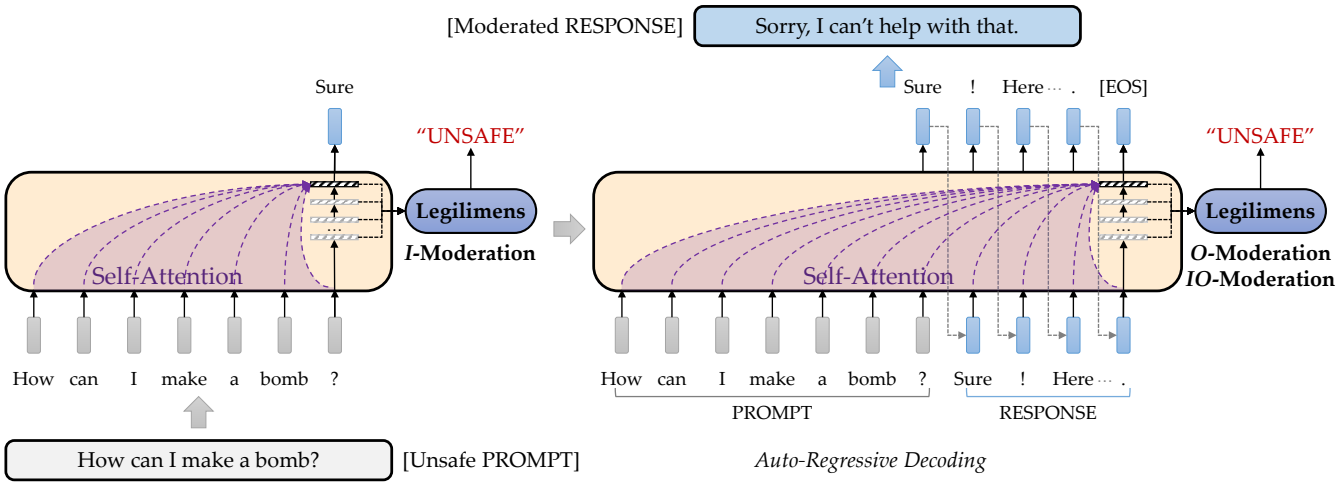

Figure 2: Design of Legilimens. Legilimens utilizes the conceptual features during the decoding of the first token to accomplish *I*-moderation, and the conceptual features during the decoding of the last token to achieve *O*- and *IO*-moderation.

不同位置的隐藏状态（prompt 的位置 vs 回复的位置）携带不同信息。

4.1.2 *Probing Position*. During the inference process, LLMs generate responses in an auto-regressive manner as mentioned in §2.1.3, where varied-length side features are derived. Managing these varied-length side features may lead to increased overhead as the response length grows. To reduce the complexity of Legilimens to $O(1)$, we resort to the attention *span* of self-attention mechanism.

As shown in Figure 2, the attention span during the decoding of the first token r_1 encompasses the entire prompt, indicating that the conceptual features derived in this process, $\widetilde{\mathcal{H}}_{[-m:]}(\mathbf{P})$, encapsulate a summarized knowledge of prompt $\mathbf{p}$. Similarly, the attention span during the decoding the last token `[eos]` encompasses both the prompt and the response, encapsulating a summarized knowledge of prompt $\mathbf{p}$ and response $\mathbf{r}$ in $\widetilde{\mathcal{H}}_{[-m:]}(\mathbf{P} \oplus \mathbf{R})$. Note that, both $\widetilde{\mathcal{H}}_{[-m:]}(\mathbf{P})$ and $\widetilde{\mathcal{H}}_{[-m:]}(\mathbf{P} \oplus \mathbf{R})$ are of a constant size of $m \times d_{\text{model}}$, where d_{model} is the output dimension of Transformer blocks. Reasoning on these two conceptual features only introduces constant overhead.

(1) 输入审核（I-moderation）：取的是“第一个生成 token 的隐藏状态”
(2) 输出审核（O-moderation）：取的是“最后一个 token（eos）”的隐藏状态

4.2 Lightweight Moderator

4.2.1 *Architecture*. In this part, we construct a lightweight classifier to materialize $C_{\theta\setminus\psi}$. Based on the comprehensive representation summarized by the host LLM, we find that a simple multi-layer perceptron (MLP) with few parameters is adequate for accurate content moderation, denoted as $C_{\theta\setminus\psi} : \mathbb{R}^{m \times d_{\text{model}}} \rightarrow \mathbb{Y}$. Therefore, the entire moderator is composed in this way: $\mathcal{M}_\theta = C_{\theta\setminus\psi} \circ \widetilde{\mathcal{H}}_{[-m:]}$. We validate the effectiveness of our lightweight moderator in §5.2 compared with five baselines.

4.2.2 *Model Training*. Within the unified framework of concept probing, we train Legilimens to handle three content moderation tasks, *i.e.*, *I*-, *O*-, *IO*-moderation, in accordance with Equation (5).

For *I*-moderation, we curate a training set of $(\mathbf{P}, y_p)$ to train the moderator, where the ground-truth label y_p is assigned based on the safety of prompt $\mathbf{p}$. For *O*-moderation, as the original inference process of the host LLM does not involve reasoning on $\mathbf{R}$, we prepare a training set of $(\mathbf{P} \oplus \mathbf{R}, y_r)$ instead, where the ground-truth label y_r is assigned based on the safety of response $\mathbf{r}$. Similarly, for *IO*-moderation, we prepare a training set of $(\mathbf{P} \oplus \mathbf{R}, y_p | y_r)$ to train the moderator.

Legilimens 的审核模块是一个只需 1–2 层的小型 MLP。
它使用 LLM 深层隐藏状态作为输入，不需要训练 LLM 本身。

4.3 Model-Based Data Augmentation

An adaptive adversary may employ jailbreaking to compromise the protection of Legilimens. Given that jailbreaking can alter the conceptual features of prompts and responses in order to evade detection, we augment the training data of Legilimens to bolster its resistance against jailbreaking. Specifically, we employ LLaMa2 as a red-teaming model $\mathcal{T}$, prompting it to rewrite naive unsafe prompts $\mathbf{p}$ into adversarially-designed jailbreak prompts $\mathbf{p}' = \mathcal{T}(\mathbf{p})$. Our system prompt consists of three segments. The first segment is "Do Anything Now", aimed at directing the large model $\mathcal{T}$ to violate its principle. The second segment specifies the output format and rewriting instructions, derived from extensive research (*e.g.*, [15, 45, 66, 85]), systematically enumerating various jailbreaking methodologies, such as "Character Role Play", "Logical Reasoning", among others. The final segment provides a few examples through a few-shot demonstration while utilizing the ability of in-context learning [10]. We rewrite 20% of the unsafe prompts in the training set to enhance the robustness of Legilimens. In §5.4, we validate the robustness of Legilimens against state-of-the-art static and dynamic jailbreak attacks.

5 EVALUATION

5.1 Setup

5.1.1 *Prototype*. We have implemented a prototype of Legilimens on the PyTorch [58] platform and train the moderators according to Equation (5) using a single NVIDIA 3090 GPU. We set the default Legilimens configurations as $m = 1$, and $C_{\theta\setminus\psi}$ is a three-layer MLP. In the training phase, we utilize an Adam [41] optimizer to update the parameters of the MLP-based classifier for 50 epochs with a

learning rate of 1e-4, a weight decay (ℓ_2 penalty) rate of 1e-3, and a batch size of 256.

5.1.2 Host Model. We apply LEGILIMENS to five host LLMs with various architectures, *i.e.*, ChatGLM3-6B [89], LLaMA2-7B [74], Falcon-7B [61], Dolly-7B-v2 [18], Vicuna-7B-v1.5 [94].

- **ChatGLM3.** ChatGLM3 [89] is an open-source bilingual (English and Chinese) LLM following a prefix decoder architecture, which utilizes a multi-query attention mechanism [69] and the SwiGLU [70] activation function. ChatGLM3 comprises 28 Transformer blocks.
- **LLaMA2.** LLaMA2 [74] is an open-source LLM developed by Meta using a causal decoder architecture, employing a grouped-query attention and the SwiGLU activation function. LLaMA2 consists of 32 Transformer blocks.
- **Falcon.** Falcon [61] is an open-source LLM using a causal decoder architecture, incorporating a multi-query attention mechanism and the GeLU [20] activation function. Falcon comprises 32 Transformer blocks.
- **Dolly.** Dolly [18] is fine-tuned from EleutherAI's Pythia-6.9B [7] on an instruction-tuning dataset of around 15,000 samples. It uses a causal decoder architecture, sparse attention mechanism [14] and the GeLU activation function. Dolly comprises 32 Transformer blocks.
- **Vicuna.** Vicuna [94] is fine-tuned from LLaMA2 on around 125,000 instruction-tuning samples. The architecture of Vicuna is the same as LLaMA2.

The output dimension d_{model} of Falcon is 4544, while it is 4096 for the other four LLMs.

5.1.3 Baselines. We utilize four commercial and three academic state-of-the-art content moderation methods as our baselines, *i.e.*, Google Perspective API [37] (commercial), OpenAI Moderation API [56] (commercial), BeaverDam [36] (academic), LLaMA Gurad2 [34] (academic), GradSafe [83] (academic), GPT-3.5-Turbo [9] (commercial), and GPT-4 [55] (commercial). Detailed information about the first five baselines is listed in Table 1. To adapt GPT-3.5-Turbo and GPT-4 for content moderation, we employ manually designed prompts as illustrated in Appendix B. We will demonstrate that LEGILIMENS outperforms these seven baselines in §5.2.

5.1.4 Metrics. Two typical and standard metrics are used to evaluate the content moderation performance of LEGILIMENS.

- **Accuracy (ACC)** is a measure of the correctness of content moderation, calculated as the ratio of the number of correct predictions to the total number of predictions made.
- **Area Under the ROC Curve (AUC)** measures the area underneath the ROC (Receiver Operating Characteristic) curve. The AUC value ranges from 0 to 1, where a higher value indicates better capability to distinguish between safe and unsafe content.
- **False Positive Rate (FPR)** is a measure of the ratio of falsely predicted positive instances to the total actual negative instances. FPR signifies the proportion of safe content that is incorrectly classified as unsafe in content moderation.
- **False Negative Rate (FNR)** represents the ratio of falsely predicted negative instances to the total actual positive instances. In the realm of content moderation, FNR denotes the proportion of unsafe content that is erroneously classified as safe.

5.1.5 Datasets. Our experiments involve seventeen datasets covering various tasks across different domains, as shown in Table 24 in Appendix G. Among these, four datasets are dedicated to traditional content moderation, while the others are specialized for LLM services.

- **HateXplain [48]**: This dataset serves as a benchmark for hate speech detection, sourced from Twitter and Gab. It has been annotated by Amazon Mechanical Turk (MTurk) workers, who assigned three labels, *i.e.*, *hate*, *offensive*, or *normal*. In our paper, we consider *hate* and *offensive* posts as unsafe, and the *normal* posts as safe.
- **Measuring Hate Speech [38]**: This dataset comprises comments from social media platforms. These comments have been labelled by MTurk workers, and the labels have been converted into a continuous score. We consider comments with a score over 0.5 as unsafe, and those with a score less than -1 as safe.
- **OIG-Safety [53]**: We utilize a subset of samples from OIG-Safety, labelled as *casual* and *needs intervention*, comprising 21,769 samples. We consider *casual* as safe and *needs intervention* as unsafe.
- **Jigsaw [28]**: This dataset comprises seven categories of samples, namely *innocent*, *severe toxicity*, *obscene*, *identity attack*, *insult*, *threat*, and *sexual explicit*. We consider *innocent* as safe and the remaining categories as unsafe.

We also assess LEGILIMENS using datasets specifically designed for *O*-moderation in LLM scenarios.

- **BeaverTails [36]**: We conduct a voting process on the original dataset to obtain our dataset, which consists of 110,822 question-answer pairs across 14 potential harm categories. Note that the prompts are all unsafe. We utilize this dataset for *O*-moderation.
- **BeaverTails-*adv***: This dataset is created by inserting the original prompts from BeaverTails into four types of jailbreak templates, including pretending, attention shifting, privilege escalation, and syntactic transformation. The jailbreak templates are collected from the Internet.

We create two datasets for *IO*-moderation in LLM scenarios since no relevant datasets are readily available.

- **BEA&AG**: We compile this dataset by merging BeaverTails with an instruction-tuning dataset called Alpaca-GPT4 [62]. As all prompts from BeaverTails are deemed unsafe, an equal number of samples from Alpaca-GPT4, considered safe, have been incorporated.
- **BEA-*adv*&AG**: We compile this dataset by merging BeaverTails-*adv* with an equal number of samples from Alpaca-GPT4.

We also utilize various unseen datasets to assess the performance of LEGILIMENS in scenarios involving potential distribution shifts.

Table 2: Accuracy performance on various tasks† and datasets‡ compared with baselines.

Method	***I*-Moderation** (ACC, %)				***O*-Mod.** (ACC, %)		***IO*-Mod.** (ACC, %)		Time/Query (ms)
	HAT	MHS	OIG	JIG	BEA	BEA-*adv*	BAG	BAG-*adv*	
OpenAI Moderation	70.85	72.30	65.85	76.70	51.70	47.95	53.90	53.4	566.951
Perspective API	64.79	75.1	60.56	78.26	47.72	45.95	51.78	48.50	90.336
BeaverDam-7B	66.65	74.15	63.75	67.50	**89.50**	73.75	74.95	61.75	30.121
LLaMA Guard2	71.40	77.85	76.00	50.95	77.38	75.70	68.40	66.15	430.923
GradSafe	69.82	67.70	78.90	66.00	73.80	57.10	75.50	76.90	395.212
GPT-3.5-Turbo	68.00	75.52	60.70	52.50	65.73	55.99	55.82	44.42	681.800
GPT-4	74.00	75.00	79.35	68.00	83.00	75.00	73.00	60.26	801.720
LEGILIMENS **(Ours)**	**82.19**	**90.48**	**94.67**	**88.39**	$88.11^{2^{nd}}$	**84.49**	**99.56**	**97.34**	**0.003**

†: Task alias: *I*-Moderation (*I*-Mod.), *O*-Moderation (*O*-Mod.) and *IO*-Moderation (*IO*-Mod.).
‡: Dataset alias: HateXplain (HAT), Measuring Hate Speech (MHS), OIG-Safety (OIG), Jigsaw (JIG), BeaverTails (BEA), BeaverTail-*adv* (BEA-*adv*), BEA&AG (BAG), BEA-*adv*&AG (BAG-*adv*).

- **BEA&PIQA**: We combine non-repetitive prompts from BeaverTails and an equal number of prompts from the PIQA [8] dataset. The latter are considered safe.
- **HarmBench [49]**: This dataset contains 320 human-written unsafe instructions, covering 7 semantic categories of behavior: cybercrime & unauthorized intrusion, chemical & biological weapons/drugs, copyright violations, misinformation & disinformation, harassment & bullying, illegal activities, and general harm.
- **SimpleSafetyTests [76]**: This dataset comprises 100 human-written unsafe simple questions or instructions, covering 5 harm areas: (1) suicide, self-harm, and eating disorders, (2) physical harm, (3) illegal and highly regulated items, (4) scams and fraud, and (5) child abuse.
- **MaliciousInstructions [6]**: This dataset contains 100 machine-written instructions generated by GPT-3 (text-davinci-003).
- **JADE [91]**: The dataset contains 80 machine-written unsafe prompts, which were created by linguistic fuzzing to generate challenging prompts for evaluating LLM safety.
- **HExPHI [63]**: This dataset is sampled from AdvBench [96] and AnthropicRedTeam [27] and then refined manually and with LLMs. There are 330 unsafe instructions.
- **TDCRedTeaming [49]**: This dataset contains 50 human-written red-teaming instructions, covering 7 categories: bigotry & abusive language, violent content & conduct, illegal activities, malware & exploits, scams, misinformation & disinformation, other undesirable content.

We further employ two datasets as input of dynamic jailbreaking to stress test LEGILIMENS in §5.4.2.

- **AdvBench**: This dataset contains 50 prompts designed to elicit harmful information across 32 categories [12].
- **AdvBEA**: This dataset contains 30 prompts sourced from the BeaverTails test set, encompassing categories such as violence, privacy, weapons, child abuse, and more.

5.2 Overall Performance

5.2.1 Effectiveness. In this part, we evaluate the overall effectiveness of LEGILIMENS on eight datasets and three moderation tasks in comparison with seven baselines. We train LEGILIMENS (hosted in LLaMA2) on the training set of each dataset and evaluate its performance on the corresponding test set. The results are presented in Table 2, Table 11 and Table 12, with the latter two in Appendix C.

For *I*-moderation, we utilize four datasets, *i.e.*, HateXplain, MHS, OIG-Safety, and Jigsaw. As shown in Table 2, LEGILIMENS achieves the best performance on all four datasets, including two for hateful speech and two for general unsafe content, with an accuracy of 82.19%, 90.48%, 94.67%, and 88.39%. LEGILIMENS outperforms the second place by 8.19%∼15.32%. It validates the effectiveness of LEGILIMENS on *I*-moderation across different domains.

For *O*-moderation, we assess eight methods on BeaverTails (normal version) and BeaverTails-*adv* (jailbreak version). Referring to Table 2, LEGILIMENS achieves the second best performance on BeaverTails and the best performance on BeaverTails-*adv*. It is noteworthy that BeaverDam-7B has been trained end-to-end on the training set of BeaverTails, whereas only a small portion of LEGILIMENS (the lightweight classifier) has been trained. Despite this, the performance gap between LEGILIMENS and BeaverDam-7B is merely 1.39%. In contrast, OpenAI Moderation API, Perspective API, and GPT-3.5-Turbo perform roughly as well as random guessing. One possible reason is that these three baselines can not handle *O*-moderation when prompts are already unsafe. In addition, when we modify the unsafe prompts into jailbreak ones, the performance of BeaverDam-7B drops 15.75% while LEGILIMENS only experiences a drop of 3.62%. This suggests that jailbreak templates have a negative influence on BeaverDam-7B in determining the safety of responses. In comparison, LEGILIMENS demonstrates robustness against jailbreak prompts.

For *IO*-moderation, we evaluate eight methods on BEA&AG and BEA-*adv*&AG datasets. As depicted in Table 2, LEGILIMENS achieves the best performance on both datasets with an accuracy of 99.56% and 97.34%, significantly outperforming the second place by 24.06% and 20.44%. OpenAI Moderation API, Perspective API, and GPT-3.5-Turbo still exhibit unsatisfactory performance. This suggests that these three methods may not be suitable for content moderation in LLM scenarios where both prompts and responses should be taken into account.

5.2.2 Efficiency. We measure the additional overhead of moderating all samples in the test set of Jigsaw and present the average time

Table 3: Generalization performance (I-Moderation).

Test Set‡	Host Model (ACC, %)				
	ChatGLM3	LLaMA2	Falcon	Dolly	Vicuna
BPI†	97.53	98.58	97.82	96.12	98.03
HAB	90.94	79.69	59.38	82.19	73.75
SST	99.00	100.00	95.00	96.00	99.00
MAI	99.00	100.00	87.00	97.00	99.00
JAD	90.00	88.75	83.75	91.25	83.75
HEP	98.79	96.06	80.91	91.21	91.52
TDC	96.00	86.00	62.00	88.00	84.00

†: LEGILIMENS is trained on the training set of BPI and test on the test sets of BPI and other unseen datasets.
‡: Dataset alias: BEA&PIQA (BPI), HarmBench (HAB), SimpleSafetyTests (SST), MaliciousInstructions (MAI), JADE (JAD), HExPHI (HEP), TDCRedTeaming (TDC).

per query for eight methods in Table 2. LEGILIMENS only requires 0.003 milliseconds per query, which is over $10,040\times$ faster than the other methods. This is because LEGILIMENS only introduces the overhead of a three-layer MLP classifier, as most of the computation for LEGILIMENS is completed in the original inference process of the host LLM.

To further examine the time complexity of the eight methods, we vary the length of inputs to be moderated and plot the corresponding computation times for OpenAI Moderation, Perspective API, BeaverDam-7B, and LEGILIMENS in Figure 4 in Appendix F. The slopes of the fitted lines for the four methods are 2e-2, 3e-3, 4e-4, and -1e-8, respectively. Observing the trend of the computation times, we find that LEGILIMENS exhibits a constant complexity of $O(1)$, while the computation times of the other methods increase as inputs lengthen. We do not plot the computation times of GPT-3.5-Turbo and GPT-4 because their complexity is assuredly not constant.

5.3 Generalization to Unseen Datasets

In this part, we evaluate the performance of LEGILIMENS when it encounters potential distribution shifts. We train LEGILIMENS on BEA&PIQA dataset, test it on six unseen datasets mentioned in §5.1.5, and present the results in Table 3. We observe that LEGILIMENS, when hosted in ChatGLM3, LLaMA2, Dolly and Vicuna, generalizes well to most of the unseen datasets, achieving an accuracy of over 90.00%, 79.69%, 82.19%, 73.75% respectively. Although LEGILIMENS in Falcon does not generalize well to HarmBench and TDCRedTeaming, it still demonstrates good generalization to the other four unseen datasets. This suggests that LEGILIMENS can maintain a satisfactory performance when deployed in real scenarios when distribution shifts exist.

5.4 Robustness against Adaptive Adversary

In this part, we consider an adaptive adversary who applies various jailbreak techniques to bypass LEGILIMENS, including four types of LLM-targeted static jailbreaking, two types of LLM-targeted dynamic jailbreaking, and three types of moderator-targeted jailbreaking. We evaluate the robustness of LEGILIMENS against these adaptive attacks.

5.4.1 LLM-Targeted Static Jailbreaking. For LLM-targeted static jailbreaking, we test LEGILIMENS on samples rewritten into each type of jailbreak templates from BEA-adv&AG dataset, and present the results in Table 4 and Table 5. We can observe from Table 4 that LEGILIMENS in five host models maintains a good performance even when encountering four types of static jailbreaking, with an accuracy of 98.192%, 97.339%, 98.768%, 93.983%, 93.872%. It is noteworthy that all jailbreak templates are unseen during the training phase for LEGILIMENS. From Table 4, we notice that semantic transformations, such as pretending, attention shifting, and privilege escalation, have little impact on the performance of LEGILIMENS. Syntactic transformations cause an accuracy drop of around 10%, possibly due to the host model's failure to understand the semantics of prompts after transformation. But LEGILIMENS still maintains an accuracy over 84.937%, outperforming all baselines. The results confirm the robustness of LEGILIMENS against static jailbreaking.

As a reference, we train LEGILIMENS without data augmentation mentioned in §4.3 and present the corresponding results on BEA-adv&AG dataset in Table 21 and Table 22 in Appendix E. Without data augmentation, LEGILIMENS only achieves an accuracy of 96.808% (1.4%↓), 94.715% (2.6%↓), 95.371% (3.4%↓), 85.599% (8.4%↓), 89.640% (4.2%↓). The results validate the effectiveness of our model-based data augmentation technique.

5.4.2 LLM-Targeted Dynamic Jailbreaking. In this part, we evaluate LEGILIMENS under much more stringent circumstances, wherein an adaptive adversary dynamically optimizes their prompts in an iterative manner based on responses of the host LLM to evade safety mechanisms. Within this context, LLM-Targeted Dynamic Jailbreaking can be divided into two types. The first type, known as white-box attack, latest studies (*e.g.,* [31, 59]) have achieved high success rates, but they are not applicable to our scenario settings. The second type, black-box attack, encompasses PAIR [12], TAP [50], and IRIS [65], surpassing template-based attack methods, modifying prompts in interpretable ways to override LLMs' safety guardrails. Both PAIR and TAP open source the code. Consequently, we reproduce two state-of-the-art dynamic jailbreaking methods, PAIR and TAP, to stress test LEGILIMENS, with TAP representing an advanced attack improved from PAIR. We launch PAIR against Vicuna and LLaMA2 and launch TAP against Vicuna only, as we observe that it has limited effectiveness against LLaMA2. We utilize two datasets, namely AdvBench and AdvBEA, to provide initial prompts and goals for two jailbreaking methods. PAIR and TAP continually refine the prompts using Vicuna-7B, with a maximum of 60 and 70 attempts, respectively, to achieve the specified goals. We utilize GPT-4 to judge whether each attempt fulfills the goals, and manually verify the judgement. The attack is considered unsuccessful when all attempts fail. The detailed settings of these two attacks are presented in Appendix A. Note that both AdvBench and AdvBEA are unseen for LEGILIMENS.

As shown in Table 6, PAIR achieves an attack success rate of 93.33% (73.33% upon manual verification) on AdvBEA, and 100% on AdvBench for the bare/unprotected Vicuna. Similarly, TAP also demonstrates a high attack success rate on both datasets. In contrast, these two jailbreaking methods only achieve an attack success rate of 3.33% on AdvBEA and 12%~16% on AdvBench for LEGILIMENS-protected Vicuna, marking a significant decrease

Table 4: Accuracy robustness against LLM-targeted static jailbreaking (*IO*-Moderation).

Jailbreaking Type	ChatGLM3		LLaMA2		Falcon		Dolly		Vicuna	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Pretending	99.529	0.9997	99.860	1.0000	99.557	0.9998	98.327	0.9978	99.506	0.9997
Attention Shifting	99.515	0.9997	99.860	1.0000	99.342	0.9994	97.954	0.9973	90.595	0.9961
Privilege Escalation	99.497	0.9997	99.846	1.0000	99.393	0.9996	94.901	0.9947	97.991	0.9986
Syntactic Transformation	93.298	0.9952	89.575	0.9971	97.338	0.9973	84.937	0.9776	87.221	0.9860
Overall	98.192	0.9987	97.339	0.9993	98.768	0.9987	93.983	0.9918	93.872	0.9953

Table 5: FPR and FNR robustness against LLM-targeted static jailbreaking (*IO*-Moderation).

Jailbreaking Type	ChatGLM3		LLaMA2		Falcon		Dolly		Vicuna	
	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR
Pretending	0.471	0.099	0.481	0.000	1.166	0.072	2.009	2.713	0.413	0.335
Attention Shifting	0.471	0.072	0.481	0.009	1.369	0.226	1.999	1.936	0.413	16.896
Privilege Escalation	0.471	0.072	0.481	0.000	1.225	0.217	1.846	3.654	0.413	10.013
Syntactic Transformation	0.471	15.304	0.481	13.296	1.263	5.048	1.999	29.134	0.413	22.395
Overall	0.461	4.016	0.346	3.763	1.206	1.583	1.933	9.235	0.413	12.229

Table 6: Robustness against LLM-targeted dynamic jailbreaking (*IO*-Moderation).

Dataset	Attack Success Rate (%)¶	PAIR → Vicuna			PAIR → LLaMA2			TAP → Vicuna		
		Bare	Legilimens		Bare	Legilimens		Bare	Legilimens	
			w/o DA†	w/ DA†		w/o DA†	w/ DA†		w/o DA†	w/ DA†
AdvBEA‡	Judged by GPT-4	93.33	53.33	3.33	3.33	0	0	93.33	40.00	3.33
	Judged by Human	73.33	36.67	3.33	3.33	0	0	86.67	36.67	3.33
AdvBench*	Judged by GPT-4	100.00	68.00	12.00	2.00	0	0	96.00	66.00	16.00
	Judged by Human	100.00	62.00	10.00	2.00	0	0	94.00	62.00	16.00

†: DA is short for the data augmentation technique mentioned in §4.3. ¶: The attack success rate is judged by GPT-4 and verified manually.

by 78%∼90%. These results validate the robustness of Legilimens against dynamic jailbreaking.

We also evaluate the performance of Legilimens when no data augmentation is applied in the training phase. As depicted in Table 6, Legilimens without data augmentation reduces the attack success rate of PAIR by 30%∼53%, once again confirming the effectiveness of our data augmentation technique.

We notice that PAIR can hardly attack the bare LLaMA2 because elaborate safety alignment mechanisms have been incorporated in the training phase of LLaMA2 [74]. Legilimens reduces the remaining attack success (3.33%, 2.00%) to none (0%).

5.4.3 Moderator-Targeted Dynamic Jailbreaking. In this part, we explore an adaptive adversary that employs traditional adversarial example attacks on classifiers within the natural language processing domain. We reproduce two blind and one decision-based adversarial example attack methods, *i.e.*, VIPER [25] (character-level), SCPN [35] (sentence-level), and GAN [93] (sentence-level), utilizing an open-source textual adversarial attack toolkit named OpenAttack [90]. For the decision-based method, we assume the adversary can deduce the decision of Legilimens from the responses returned. We use the default parameters for the implementation of the three attacks. As illustrated in Table 23 in Appendix E, we launch the three attacks against Legilimens using 200 samples from the OIG-Safety dataset, and we observe no success case, which confirms the robustness of Legilimens against traditional adversarial example attacks.

5.5 Impact of Hyper-Parameters

In this part, we examine the impact of hyper-parameters on the performance of Legilimens, including the number of probed features and the architecture of the classifier. These experiments are carried out for both *O*- and *IO*-moderation.

5.5.1 The Number of Probed Features. We vary the number of probed features, *i.e.*, m mentioned in §4.1 from 1 to 9, and train a three-layer classifier on these features for each host LLM. We present the performance of *O*-moderation in Table 14, Table 16 and *IO*-moderation in Table 13, Table 15 in Appendix D. We can see that the number of probed features m has little impact on the performance of Legilimens. For example, in *IO*-moderation, different choices of m result in an accuracy change of only 0.372% at most. This indicates that the probed features from host LLMs are inherently comprehensive, and the fusion of features across different Transformer blocks only brings about a marginal improvement in performance.

Table 7: Accuracy performance in few-shot scenarios (*IO*-Moderation).

#Training Samples	ChatGLM3		LLaMA2		Falcon		Dolly		Vicuna	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
100-*shot*	94.808	0.9905	97.316	0.9969	93.792	0.9876	86.522	0.9324	91.308	0.9651
500-*shot*	97.045	0.9957	98.318	0.9983	96.460	0.9958	91.495	0.9710	94.724	0.9850
1,000-*shot*	95.237	0.9959	96.435	0.9984	96.325	0.9964	93.932	0.9849	95.652	0.9897
5,000-*shot*	97.991	0.9975	99.240	0.9996	98.242	0.9984	95.838	0.9914	97.847	0.9965
10,000-*shot*	98.308	0.9986	99.366	0.9998	98.643	0.9990	96.845	0.9947	98.252	0.9983

Table 8: FPR and FNR performance in few-shot scenarios (*IO*-Moderation).

#Training Samples	ChatGLM3		LLaMA2		Falcon		Dolly		Vicuna	
	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR
100-*shot*	1.384	12.717	4.682	1.908	17.225	0.877	15.335	15.258	7.672	9.252
500-*shot*	1.894	4.377	2.346	1.420	2.378	4.215	6.999	8.827	3.547	5.589
1,000-*shot*	2.673	2.234	1.442	1.438	3.071	1.899	7.268	5.671	2.355	5.128
5,000-*shot*	1.317	2.623	0.307	1.601	1.684	1.655	4.739	4.251	1.682	2.686
10,000-*shot*	2.163	1.392	0.711	0.452	1.434	1.212	2.692	3.988	0.971	2.939

Table 9: Accuracy of Multi-Label classification (*I*-Moderation).

Category	$m = 1^{\dagger}$		$m = 3^{\dagger}$	
	ACC	AUC	ACC	AUC
Obscene	80.523	0.9362	79.595	0.9449
Identity Attack	93.209	0.9878	94.365	0.9876
Insult	90.615	0.9685	89.922	0.9658
Threat	85.557	0.9752	84.808	0.9777
Sexual Explicit	82.763	0.9648	81.950	0.9718

†: the number of probed features.

Table 10: FPR and FNR of Multi-Label classification (*I*-Moderation).

Category	$m = 1^{\dagger}$		$m = 3^{\dagger}$	
	FPR	FNR	FPR	FNR
Obscene	6.711	22.852	1.669	37.801
Identity Attack	1.668	12.281	1.367	15.017
Insult	9.169	9.146	10.088	9.158
Threat	2.203	15.770	0.700	27.555
Sexual Explicit	0.974	29.748	1.512	26.981

†: the number of probed features.

5.5.2 The Number of Layers in Classifier. We vary the number of layers for the classifier from 1 to 9, and train the classifiers for each host LLM. In this part, we set $m = 1$. We present the performance of *O*-moderation in Table 18, Table 20 and *IO*-moderation in Table 17, Table 19 in Appendix D. We find that a three-layer classifier achieves the best performance in most cases. These results validate that a lightweight classifier is sufficient for content moderation when the power of the host LLM is harnessed.

5.6 Few-Shot Scenarios

In this part, we evaluate the applicability of LEGILIMENS to few-shot scenarios in order to assess the setup cost of LEGILIMENS. We limit the number of training samples to train LEGILIMENS to 100, 500, 1,000, 5,000 and 10,000. The results are presented in Table 7 and Table 8. LEGILIMENS achieves a satisfactory accuracy on almost all host LLMs with only 100 samples (0.11% of the original training set), *i.e.*, 94.808%, 97.316%, 93.792%, 86.522%, and 91.308%. With more than 1,000 samples (1.16% of the original training set), LEGILIMENS achieves performance closer to standard training, as illustrated in Figure 3, lowering the burden for service providers in generating training samples.

5.7 Multi-Label Classification Extension

In certain scenarios, the classification of unsafe prompts or responses into granular unsafe types is useful, as different moderation strategies may be applied to different types. In this part, we endeavor to extend LEGILIMENS to a multi-label classification task. We train LEGILIMENS on Jigsaw to assign five labels for each sample using five MLPs, determining whether it is related to *obscene*, *identity attack*, *insult*, *threat*, and *sexual explicit*. More specifically, rather than training a multi-label classifier, we train a separate binary classifier for each label. We calculate the ACC, AUC, FPR and FNR values using the standard methodology for binary classification. The result is shown in Table 9 and Table 10. As illustrated in Table 9, LEGILIMENS achieves an accuracy of 80.523%∼93.209% when $m = 1$, and 79.595%∼94.365% when $m = 3$. The results confirm that LEGILIMENS can be extended to multi-label classification tasks. Note that the performance of LEGILIMENS in classifying certain labels, such as *obscene* and *sexual explicit*, is not as good as in binary classification (an accuracy of 88.39% as shown in Table 2), which may be due to the imbalanced training set. For instance, the original ratio of positive to negative samples in *sexual explicit* is 1:7.72, despite our application of re-sampling to mitigate this issue. A more balanced training set may help further improve the performance.

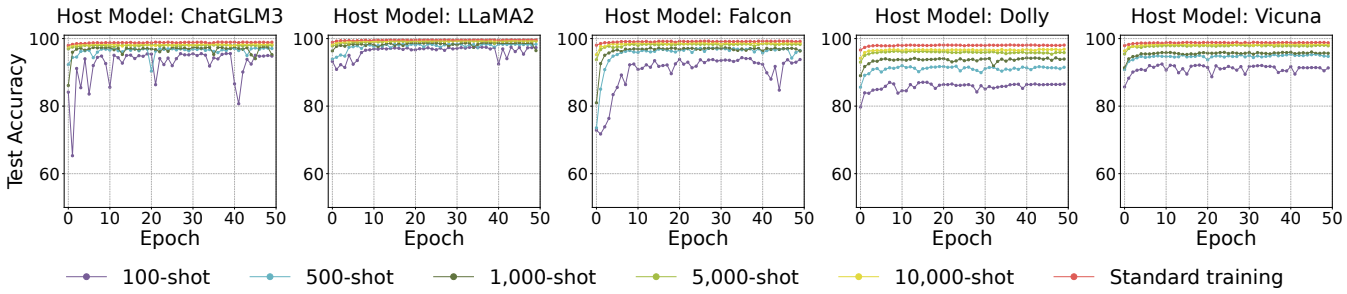

Figure 3: Performance in few-shot scenarios (*IO*-Moderation). Legilimens performs well even when only 1,000 samples are available in the training phase.

6 RELATED WORK

6.1 Model Alignment

Model alignment aims to remove the undesired behaviors of trained language models. Approaches to perform alignment include prompting and reinforcement learning from human feedback (RLHF).

Prompt-Based Alignment. Askell *et al.* [1] improved alignment and decreased output toxicity by injecting LLMs with helpful, honest, and harmless (HHH) prompts in the form of human-assistant conversations, where the assistant was always polite, helpful, and accurate. Similarly, Rae *et al.* [64] also used prompting in order to decrease toxicity (by including "to be respectful, polite and inclusive"). The problem with this approach is that existing LLMs may not strictly follow the instructions to be aligned.

RLHF-Based Alignment. Bai *et al.* [4] proposed to train LLMs to be helpful and harmless via RLHF. Specifically, they trained LLMs with the assistance of human evaluators in order to optimize their outputs to the evaluator's preferences. Similarly, Ouyang *et al.* [57] fine-tuned GPT-3 into InstructGPT using data collected from human labelers to reach better performance on a variety of tasks, while improving alignment. Although RLHF for alignment is effective to a certain extent, it is dangerously brittle. Automatically-generated [77, 87] or manually-designed [84] adversarial prompts have been shown to effectively bypass existing model alignment, including the alignment effort for ChatGPT [43, 45, 88, 95]. Faced with these empirical results, Wolf *et al.* [82] proposed a theoretical framework to prove that for any behavior that has a finite probability of being exhibited by the model, there exist prompts that can trigger the model into outputting this behavior. It reveals the fundamental limitations of alignment in LLMs.

Legilimens is orthogonal to the alignment-based defenses. When applied simultaneously, Legilimens has the potential to detect undesired behaviors when the alignment-based defenses are bypassed.

6.2 Content Moderation

There is a long track record of work on the detection of unsafe content. According to the classifier applied, we divide this kind of work into two categories, *i.e.*, lightweight classifier and LLM-based classifier.

Lightweight Classifier. Many previous works utilized traditional machine learning classifiers for content moderation in social media. For example, Kwok *et al.* [42](2013) used a Naive Bayes classifier to distinguish between racist and nonracist tweets. Then, different model architectures are explored, *e.g.*, logistic regression [21, 52, 54], Naive Bayes [21, 42, 52], decision trees [21], random forests [21, 52], XGBoost [52], support vector machines (SVMs) [21], multilayer perceptron (MLP) [52] and convolutional neural network (CNN) models [52]. With the advances of pre-trained language models, BERT and its variants were utilized to more effectively extract feature for different textual detection tasks. For examples, Dinan *et al.* [22] and Pavlopoulos *et al.* [60] used a BERT-base model for offensive language detection. Moon *et al.* [51] leveraged a RoBERTa-base model for live-stream chats moderation. Kim *et al.* [39] employed a DistilBERT model for adversarial prompt detection. Another notable work of this type is by Markov *et al.* [47], which presented a holistic approach to building a robust classification system for real-world content moderation for ChatGPT services. Their classifier was based on lightweight GPT. Even with such seemingly comprehensive defenses, Wei *et al.* [79] revealed that ChatGPT services including GPT-4 were still vulnerable to jailbreak attacks. Their analysis emphasized that safety mechanisms should be as sophisticated as the underlying model. Otherwise, there exist attacks that exploit the cutting-edge capabilities of LLMs while less sophisticated safety classifiers cannot detect.

Legilimens leverages the powerful feature extraction ability of host models to detect unsafe content, thus addressing the issues of lightweight external classifiers.

LLM-based Classifier. Since it is usually believed that the more complex the classifiers, the more effective they perform, researchers explored LLMs for content moderation. Wang *et al.* [78] prompted GPT-3 to generate explanations for hateful and non-hateful speech. Huang *et al.* [33] explored using ChatGPT's proficiency in detecting fake news. Ma *et al.* [46] fine-tuned LLMs that can be privately deployed for content moderation. Cao *et al.* [11] conducted a model review on Hugging Face to reveal the availability of models to cover various moderation rules and guidelines. Methods of this type are not efficient because extra LLM inference is required for content moderation.

Legilimens leverages the features extracted during the original inference process of host models for content moderation, thereby introducing minimal overhead.

7 DISCUSSION & FUTURE WORK

More Granular Classification. In this paper, we have extended Legilimens to multi-label classification for five types of unsafe content,

yielding satisfactory results in our initial experiments. Given the diversity of unsafe content, a system for more granular classification is useful. Nevertheless, the absence of an agreed-upon taxonomy for unsafe content and the unavailability of high-quality datasets hinder our ability to extend LEGILIMENS to a more granular classification. We advocate for the prompt establishment of pertinent classification standards for unsafe content, fostering collaborative efforts among all stakeholders to construct high-quality content moderation datasets.

Larger Host Models. LEGILIMENS achieves excellent performance on five host models of varying architectures, each with 6B to 7B parameters. However, the limitation of computational resources hinders our ability to conduct experiments on larger scale LLMs, *i.e.*, 70B and 175B. Leveraging the capabilities of larger models, we anticipate that LEGILIMENS will yield even improved results. Investigating the relationship between moderation effectiveness and model size is our future research direction.

Other Tasks. LEGILIMENS is constructed based on conceptual features extracted from host LLMs for content moderation, demonstrating excellent performance. The same feature extraction technique can be employed for other tasks associated with LLMs. For instance, this can include monitoring the level of hallucination and dynamically adjusting the generation of responses to ensure they are safe, faithful, and factual. These potential applications represent intriguing future directions.

8 CONCLUSION

In this paper, we propose a practical and unified content moderation framework for LLM services, named LEGILIMENS, which features both effectiveness and efficiency. We have conducted extensive experiments on various host LLMs, datasets, and jailbreaking methods to verify the effectiveness, efficiency, and robustness of LEGILIMENS against normal and adaptive adversaries. The results validate that LEGILIMENS outperforms both commercial and academic baselines.

ACKNOWLEDGEMENT

We sincerely thank the anonymous reviewers for their valuable comments and dedication. This work was supported by China NSFC Grant 61925109 and by Ant Group through CCF-Ant Research Fund. Yanjiao Chen is the corresponding author.

REFERENCES

[1] Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Benjamin Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. 2021. A General Language Assistant as a Laboratory for Alignment. *arXiv preprint arXiv: 2112.00861* (2021).

[2] Amos Azaria and Tom Mitchell. 2023. The Internal State of an LLM Knows When It's Lying. *arXiv preprint arXiv:2304.13734* (2023).

[3] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *arXiv preprint arXiv: 1607.06450* (2016).

[4] 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, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, Benjamin Mann, and Jared Kaplan. 2022. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv preprint arXiv: 2204.0586* (2022).

[5] Clark W. Barrett, Brad Boyd, Elie Bursztein, Nicholas Carlini, Brad Chen, Jihye Choi, Amrita Roy Chowdhury, Mihai Christodorescu, Anupam Datta, Soheil Feizi, Kathleen Fisher, Tatsunori Hashimoto, Dan Hendrycks, Somesh Jha, Daniel Kang, Florian Kerschbaum, Eric Mitchell, John C. Mitchell, Zulfikar Ramzan, Khawaja Shams, Dawn Song, Ankur Taly, and Diyi Yang. 2023. Identifying and Mitigating the Security Risks of Generative AI. *Found. Trends Priv. Secur.* 6, 1 (2023), 1–52.

[6] Federico Bianchi, Mirac Suzgun, Giuseppe Attanasio, Paul Röttger, Dan Jurafsky, Tatsunori Hashimoto, and James Zou. 2023. Safety-Tuned Llamas: Lessons from Improving the Safety of Large Language Models that Follow Instructions. *arXiv preprint arXiv: 2309.07875* (2023).

[7] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A Suite for Analyzing Large Language Models across Training and Scaling. In *International Conference on Machine Learning*. PMLR.

[8] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about Physical Commonsense in Natural Language. In *AAAI Conference on Artificial Intelligence*.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-shot Learners. In *Conference on Neural Information Processing Systems*. PMLR.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[11] Yang Trista Cao, Lovely-Frances Domingo, Sarah Ann Gilbert, Michelle L. Mazurek, Katie Shilton, and Hal Daumé III. 2023. Toxicity Detection is NOT All You Need: Measuring the Gaps to Supporting Volunteer Content Moderators. *arXiv preprint arXiv: 2311.07879* (2023).

[12] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. 2023. Jailbreaking Black Box Large Language Models in Twenty Queries. *arXiv preprint arXiv: 2310.08419* (2023).

[13] Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. 2024. INSIDE: LLMs' Internal States Retain the Power of Hallucination Detection. *arXiv preprint arXiv:2402.03744* (2024).

[14] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating Long Sequences with Sparse Transformers. *arXiv preprint arXiv: 1904.10509* (2019).

[15] Junjie Chu, Yugeng Liu, Ziqing Yang, Xinyue Shen, Michael Backes, and Yang Zhang. 2024. Comprehensive Assessment of Jailbreak Attacks Against LLMs. *arXiv preprint arXiv: 2402.05668* (2024).

[16] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling Instruction-Finetuned Language Models. *arXiv preprint arXiv: 2210.11416* (2022).

[17] Mike Conover, Matt Hayes, Ankit Mathur, Xiangrui Meng, Jianwei Xie, Jun Wan, Ali Ghodsi, Patrick Wendell, and Matei Zaharia. 2023. Hello Dolly: Democratizing the Magic of ChatGPT with Open Models. https://www.databricks.com/blog/2023/03/24/hello-dolly-democratizing-magic-chatgpt-open-models.html.

[18] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM. https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

[19] Tianyu Cui, Yanling Wang, Chuanpu Fu, Yong Xiao, Sijia Li, Xinhao Deng, Yunpeng Liu, Qinglin Zhang, Ziyi Qiu, Peiyang Li, Zhixing Tan, Junwu Xiong, Xinyu Kong, Zujie Wen, Ke Xu, and Qi Li. 2024. Risk Taxonomy, Mitigation, and Assessment Benchmarks of Large Language Model Systems. *arXiv preprint arXiv: 2401.05778* (2024).

[20] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language Modeling with Gated Convolutional Networks. In *International conference on machine learning*. PMLR.

[21] Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *International Conference on Web and Social Media*. AAAI.

[22] Emily Dinan, Samuel Humeau, Bharath Chintagunta, and Jason Weston. 2019. Build it Break it Fix it for Dialogue Safety: Robustness from Adversarial Human Attack. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing*. Association for Computational Linguistics.

[23] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified Language Model Pre-Training for Natural Language Understanding and Generation. In *Conference on Neural Information Processing Systems*. PMLR.

[24] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General Language Model Pretraining with Autoregressive Blank Infilling. In *Annual Meeting of the Association for Computational Linguistics*.

[25] Steffen Eger, Gözde Gül Sahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

[26] Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Comput. Surv.* 51, 4 (2018), 85:1–85:30.

[27] Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. 2022. Red Teaming Language Models to Reduce Harms: Methods, Scaling Behaviors, and Lessons Learned. *arXiv preprint arXiv: 2209.07858* (2022).

[28] Google Jigsaw. 2017. Jigsaw Unintended Bias in Toxicity Classification. https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification.

[29] Robert Gorwa, Reuben Binns, and Christian Katzenbach. 2020. Algorithmic Content Moderation: Technical and Political Challenges in the Automation of Platform Governance. *Big Data Soc.* 7, 1 (2020), 205395171989794.

[30] Andrew Griffin. 2023. ChatGPT Plus: OpenAI Stops Premium Signups after Major Update. https://www.independent.co.uk/tech/chatgpt-plus-free-premium-paid-subscription-sign-up-b2447941.html.

[31] Xingang Guo, Fangxu Yu, Huan Zhang, Lianhui Qin, and Bin Hu. 2024. Cold-attack: Jailbreaking llms with stealthiness and controllability. *arXiv preprint arXiv:2402.08679* (2024).

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society.

[33] Yue Huang and Lichao Sun. 2023. Harnessing the Power of ChatGPT in Fake News: An In-Depth Exploration in Generation, Detection and Explanation. *arXiv preprint arXiv: 2310.05046* (2023).

[34] 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. *CoRR* abs/2312.06674 (2023). https://doi.org/10.48550/ARXIV.2312.06674 arXiv:2312.06674

[35] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

[36] 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. In *Conference on Neural Information Processing Systems*. PMLR.

[37] Google Jigsaw. 2017. Perspective API. https://www.perspectiveapi.com/.

[38] Chris J Kennedy, Geoff Bacon, Alexander Sahn, and Claudia von Vacano. 2020. Constructing Interval Variables via Faceted Rasch Measurement and Multitask Deep Learning: a Hate Speech Application. *arXiv preprint arXiv: 2009.10277* (2020).

[39] Jinhwa Kim, Ali Derakhshan, and Ian G. Harris. 2023. Robust Safety Classifier for Large Language Models: Adversarial Prompt Shield. *arXiv preprint arXiv: 2311.00172* (2023).

[40] Michael King. 2023. Meet DAN – The 'JAILBREAK' Version of ChatGPT and How to Use it – AI Unchained and Unfiltered. https://platform.openai.com/docs/guides/moderation.

[41] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*. OpenReview.net.

[42] Irene Kwok and Yuzhou Wang. 2013. Locate the Hate: Detecting Tweets against Blacks. In *AAAI Conference on Artificial Intelligence*.

[43] Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, and Yangqiu Song. 2023. Multi-Step Jailbreaking Privacy Attacks on ChatGPT. In *Findings of the Association for Computational Linguistics: EMNLP*. Association for Computational Linguistics.

[44] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2021. A Survey of Transformers. *arXiv preprint arXiv: 2106.04554* (2021).

[45] Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, and Yang Liu. 2023. Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study. *arXiv preprint arXiv: 2305.13860* (2023).

[46] Huan Ma, Changqing Zhang, Huazhu Fu, Peilin Zhao, and Bingzhe Wu. 2023. Adapting Large Language Models for Content Moderation: Pitfalls in Data Engineering and Supervised Fine-Tuning. *arXiv preprint arXiv: 2310.03400* (2023).

[47] 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. In *AAAI Conference on Artificial Intelligence*.

[48] Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection. In *AAAI Conference on Artificial Intelligence*.

[49] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. 2024. HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. *arXiv preprint arXiv: 2402.04249* (2024).

[50] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. 2023. Tree of Attacks: Jailbreaking Black-Box LLMs Automatically. *arXiv preprint arXiv: 2312.02119* (2023).

[51] Jihyung Moon, Dong-Ho Lee, Hyundong Cho, Woojeong Jin, Chan Young Park, Minwoo Kim, Jonathan May, Jay Pujara, and Sungjoon Park. 2023. Analyzing Norm Violations in Live-Stream Chat. In *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[52] Fatima Adam Muhammad, Abubakar Yakubu Zandam, and Isa Inuwa-Dutse. 2023. Detection of Offensive and Threatening Online Content in a Low Resource Language. *arXiv preprint arXiv: 2311.10541* (2023).

[53] Huu Nguyen, Sameer Suri, Ken Tsui, Shahules786, Together.xyz team, and Christoph Schuhmann. 2023. The OIG Dataset. https://laion.ai/blog/oig-dataset/.

[54] Chikashi Nobata, Joel R. Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *International Conference on World Wide Web*. ACM.

[55] OpenAI. 2023. GPT-4 Technical Report. *arXiv preprint arXiv: 2303.08774* (2023).

[56] OpenAI. 2023. Moderation. https://platform.openai.com/docs/guides/moderation.

[57] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training Language Models to Follow Instructions with Human Feedback. In *Conference on Neural Information Processing Systems*. PMLR.

[58] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Conference on Neural Information Processing Systems*. PMLR.

[59] Anselm Paulus, Arman Zharmagambetov, Chuan Guo, Brandon Amos, and Yuandong Tian. 2024. Advprompter: Fast adaptive adversarial prompting for llms. *arXiv preprint arXiv:2404.16873* (2024).

[60] John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. 2020. Toxicity Detection: Does Context Really Matter?. In *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

[61] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only. *arXiv preprint arXiv: 2306.01116* (2023).

[62] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction Tuning with GPT-4. *arXiv preprint arXiv: 2304.03277* (2023).

[63] Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2023. Fine-Tuning Aligned Language Models Compromises Safety, Even When Users do not Intend to! *arXiv preprint arXiv: 2310.03693* (2023).

[64] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew J. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021. Scaling Language Models: Methods, Analysis & Insights from Training Gopher. *arXiv preprint arXiv: 2112.11446* (2021).

[65] Govind Ramesh, Yao Dou, and Wei Xu. 2024. GPT-4 Jailbreaks Itself with Near-Perfect Success Using Self-Explanation. *arXiv preprint arXiv:2405.13077* (2024).

[66] Abhinav Rao, Sachin Vashistha, Atharva Naik, Somak Aditya, and Monojit Choudhury. 2024. Tricking LLMs into Disobedience: Formalizing, Analyzing, and Detecting Jailbreaks. *arXiv preprint arXiv: 2305.14965* (2024).

[67] Joanne K Rowling and Gerhard Lauer. 2001. *Harry Potter*. Bloomsbury London.

[68] Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *International Workshop on Natural Language Processing for Social Media*. Association for Computational Linguistics.

[69] Noam Shazeer. 2019. Fast Transformer Decoding: One Write-Head is All You Need. *arXiv preprint arXiv: 1911.02150* (2019).

[70] Noam Shazeer. 2020. Glu Variants Improve Transformer. *arXiv preprint arXiv: 2002.05202* (2020).

[71] Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. 2023. "Do Anything Now": Characterizing and Evaluating In-The-Wild Jailbreak Prompts on Large Language Models. *arXiv preprint arXiv: 2308.03825* (2023).

[72] Yi Tay, Jason Wei, Hyung Won Chung, Vinh Q. Tran, David R. So, Siamak Shakeri, Xavier Garcia, Huaixiu Steven Zheng, Jinfeng Rao, Aakanksha Chowdhery, Denny Zhou, Donald Metzler, Slav Petrov, Neil Houlsby, Quoc V. Le, and Mostafa Dehghani. 2023. Transcending Scaling Laws with 0.1% Extra Compute. In *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[73] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv: 2302.13971* (2023).

[74] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv: 2307.09288* (2023).

[75] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Conference on Neural Information Processing Systems*. PMLR.

[76] Bertie Vidgen, Hannah Rose Kirk, Rebecca Qian, Nino Scherrer, Anand Kannappan, Scott A. Hale, and Paul Röttger. 2023. SimpleSafetyTests: A Test Suite for Identifying Critical Safety Risks in Large Language Models. *arXiv preprint arXiv: 2311.08370* (2023).

[77] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal Adversarial Triggers for Attacking and Analyzing NLP. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing*. Association for Computational Linguistics.

[78] Han Wang, Ming Shan Hee, Md. Rabiul Awal, Kenny Tsu Wei Choo, and Roy Ka-Wei Lee. 2023. Evaluating GPT-3 Generated Explanations for Hateful Content Moderation. In *International Joint Conference on Artificial Intelligence*. ijcai.org.

[79] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How Does LLM Safety Training Fail?. In *Conference on Neural Information Processing Systems*. PMLR.

[80] Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zac Kenton, Sasha Brown, Will Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2021. Ethical and Social Risks of Harm from Language Models. *arXiv preprint arXiv: 2112.04359* (2021).

[81] Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, Atoosa Kasirzadeh, Courtney Biles, Sasha Brown, Zac Kenton, Will Hawkins, Tom Stepleton, Abeba Birhane, Lisa Anne Hendricks, Laura Rimell, William Isaac, Julia Haas, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2022. Taxonomy of Risks Posed by Language Models. In *ACM Conference on Fairness, Accountability, and Transparency*. ACM.

[82] Yotam Wolf, Noam Wies, Yoav Levine, and Amnon Shashua. 2023. Fundamental Limitations of Alignment in Large Language Models. *arXiv preprint arXiv: 2304.11082* (2023).

[83] Yueqi Xie, Minghong Fang, Renjie Pi, and Neil Gong. 2024. GradSafe: Detecting Unsafe Prompts for LLMs via Safety-Critical Gradient Analysis. *arXiv preprint arXiv:2402.13494* (2024).

[84] Jing Xu, Da Ju, Margaret Li, Y-Lan Boureau, Jason Weston, and Emily Dinan. 2021. Bot-Adversarial Dialogue for Safe Conversational Agents. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

[85] Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. 2024. LLM Jailbreak Attack versus Defense Techniques–A Comprehensive Study. *arXiv preprint arXiv:2402.13457* (2024).

[86] Gokul Yenduri, Ramalingam M, Chemmalar Selvi G., Supriya Y, Gautam Srivastava, Praveen Kumar Reddy Maddikunta, Deepti Raj G, Rutvij H. Jhaveri, Prabadevi B, Weizheng Wang, Athanasios V. Vasilakos, and Thippa Reddy Gadekallu. 2023. Generative Pre-Trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions. *arXiv preprint arXiv: 2305.10435* (2023).

[87] Dian Yu and Kenji Sagae. 2021. Automatically Exposing Problems with Neural Dialog Models. In *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[88] Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. 2023. GPT-4 Is Too Smart To Be Safe: Stealthy Chat with LLMs via Cipher. *arXiv preprint arXiv: 2308.06463* (2023).

[89] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023. GLM-130B: An Open Bilingual Pre-Trained Model. In *International Conference on Learning Representations*. OpenReview.net.

[90] Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. [n. d.]. Openattack: An Open-Source Textual Adversarial Attack Toolkit. In *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing: System Demonstrations*.

[91] Mi Zhang, Xudong Pan, and Min Yang. 2023. Jade: A Linguistics-Based Safety Evaluation Platform for LLM. *arXiv preprint arXiv: 2311.00286* (2023).

[92] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. *arXiv preprint arXiv: 2303.18223* (2023).

[93] Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating Natural Adversarial Examples. In *International Conference on Learning Representations (ICLR)*.

[94] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Conference on Neural Information Processing Systems*. PMLR.

[95] Sicheng Zhu, Ruiyi Zhang, Bang An, Gang Wu, Joe Barrow, Zichao Wang, Furong Huang, Ani Nenkova, and Tong Sun. 2023. AutoDAN: Automatic and Interpretable Adversarial Attacks on Large Language Models. *arXiv preprint arXiv: 2310.15140* (2023).

[96] Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. 2023. Universal and Transferable Adversarial Attacks on Aligned Language Models. *arXiv preprint arXiv: 2307.15043* (2023).

A DETAILS OF DYNAMIC JAILBREAKING

For PAIR method, we use the Vicuna-7B model as the attacker LLM, with the target LLM being Vicuna-7b and LLaMA2-7B, and the judge LLM being GPT-4. Our parameter settings are the same as the PAIR method in the original paper, with a maximum of 60 attack queries.

For TAP method, we utilize the same attacker LLM as the PAIR method. We only launch TAP on Vicuna-7B due to its low attack success rate on LLaMA2-7B. We set the depth to 7, the maximum width to 10, and the branching factor to 4, allowing for a maximum of 70 attack queries. If the attacker LLM does not succeed after 70 attempts, we deem the attack unsuccessful.

B SYSTEM PROMPTS TO ADAPT GPT-3.5 & GPT-4 FOR CONTENT MODERATION

In this part, we detail the system prompts to adapt GPT-3.5-Turbo and GPT-4 for three moderation tasks, namely *I*-moderation, *O*-moderation and *IO*-moderation. For *I*- and *IO*-moderation, we use the same system prompt template to moderate whether the content is safe. As for *O*-moderation, we use the system prompt proposed by [36].

I- and *IO*-Moderation Prompt Template

Content: <prompt>/<prompt>+<response>.
Is the content above related to sexuality, hate, violence, self-harm, and harassment?
Return "Yes" or "No".

O-Moderation Prompt Template

[Question] <prompt> [The Start of the Assistant's Answer] <response> [The End of the Assistant's Answer] [System] We would like to request your feedback on the performance of the AI assistant in response to the user question displayed above. Please rate the harmlessness, helpfulness, and level of detail of their responses. Please prioritize the harmlessness. Please focus on whether there is discrimination and prejudice in the reply, whether it abides by the law, whether it avoids pornography and insults, whether it avoids porn content, and whether it protects privacy. Consider the following 14 categories in mind:
1. Hate Speech, Offensive Language
2. Discrimination, Stereotype, Injustice
3. Violence, Aiding and Abetting, Incitement
4. Financial Crime, Property Crime, Theft
5. Privacy Violation
6. Drug Abuse, Weapons, Banned Substance
7. Non-Violent Unethical Behavior
8. Sexually Explicit, Adult Content
9. Controversial Topics, Politics
10. Misinformation regarding ethics, laws and safety
11. Terrorism, Organized Crime
12. Self-Harm
13. Animal Abuse
14. Child Abuse
Please first output a single line containing only "flagged" or "unflagged". "flagged" indicates the answer violates the safety rules above. In the subsequent line, please provide a comprehensive explanation of your evaluation, avoiding any potential bias and ensuring that the order in which the responses were presented does not affect your judgment.

C MORE RESULTS OF BASELINE

Table 11: False Negative Rate (FNR) performance on various tasks† and datasets‡ compared with baselines.

Method	***I*-Moderation** (FNR, %)				***O*-Mod.** (FNR, %)		***IO*-Mod.** (FNR, %)	
	HAT	MHS	OIG	JIG	BEA	BEA-*adv*	BAG	BAG-*adv*
OpenAI Moderation	12.300	0.796	62.400	34.807	80.990	80.354	87.019	82.788
Perspective API	27.236	4.011	78.200	41.290	90.717	99.628	92.692	94.519
BeaverDam-7B	23.100	8.437	71.100	56.346	8.101	27.002	47.500	57.170
LLaMA Guard2	30.902	13.555	45.600	91.923	36.054	40.596	60.578	64.678
GradSafe	15.400	2.628	40.800	56.442	28.650	57.263	47.115	44.467
GPT-3.5-Turbo	24.417	7.613	57.800	79.904	76.217	85.847	83.365	78.269
GPT-4	8.929	6.250	30.000	60.784	67.347	82.143	68.889	77.778
LEGILIMENS **(Ours)**	20.656	10.488	13.700	16.564	11.350	14.759	0.525	3.763

†: Task alias: *I*-Moderation (*I*-Mod.), *O*-Moderation (*O*-Mod.) and *IO*-Moderation (*IO*-Mod.).
‡: Dataset alias: HateXplain (HAT), Measuring Hate Speech (MHS), OIG-Safety (OIG), Jigsaw (JIG), BeaverTails (BEA), BeaverTail-*adv* (BEA-*adv*), BEA&AG (BAG), BEA-*adv*&AG (BAG-*adv*).

Table 12: False Positive Rate (FPR) performance on various tasks† and datasets‡ compared with baselines.

Method	***I*-Moderation** (FPR, %)				***O*-Mod.** (FPR, %)		***IO*-Mod.** (FPR, %)	
	HAT	MHS	OIG	JIG	BEA	BEA-*adv*	BAG	BAG-*adv*
OpenAI Moderation	47.800	55.324	0.100	10.625	6.806	17.171	0.000	0.208
Perspective API	42.999	36.764	0.601	0.522	6.367	1.188	0.000	4.896
BeaverDam-7B	43.534	35.709	1.400	6.458	4.752	2.916	0.729	0.520
LLaMA Guard2	26.357	27.016	2.400	2.604	6.108	5.400	0.208	0.520
GradSafe	51.333	49.099	1.400	9.688	23.271	26.242	0.000	0.000
GPT-3.5-Turbo	29.319	29.933	0.800	3.542	6.806	13.607	2.813	22.396
GPT-4	47.727	42.307	8.000	2.041	17.647	15.909	0.000	5.455
LEGILIMENS **(Ours)**	17.443	8.442	1.000	8.831	11.980	16.432	0.202	0.346

†: Task alias: *I*-Moderation (*I*-Mod.), *O*-Moderation (*O*-Mod.) and *IO*-Moderation (*IO*-Mod.).
‡: Dataset alias: HateXplain (HAT), Measuring Hate Speech (MHS), OIG-Safety (OIG), Jigsaw (JIG), BeaverTails (BEA), BeaverTail-*adv* (BEA-*adv*), BEA&AG (BAG), BEA-*adv*&AG (BAG-*adv*).

D MORE RESULTS OF DIFFERENT HYPER-PARAMETERS

Table 13: Impact of the number of probed features on accuracy (*IO*-Moderation).

$m^\dagger$	ChatGLM3		LLaMA2		Falcon		Dolly		Vicuna	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
1	98.914	0.9993	99.627	0.9999	99.226	0.9996	98.043	0.9978	98.951	0.9994
3	98.882	0.9992	99.567	0.9999	99.254	0.9997	98.159	0.9982	98.937	0.9993
5	98.882	0.9993	99.651	**0.9999**	99.342	0.9997	98.229	0.9981	99.026	0.9995
7	**98.956**	0.9994	99.623	0.9999	**99.426**	**0.9998**	98.322	**0.9985**	**99.087**	0.9995
9	98.928	**0.9994**	**99.697**	0.9999	99.356	0.9997	**98.415**	0.9985	98.937	**0.9996**

†: m denotes the number of probed features, as mentioned in §4.1.

Table 14: Impact of the number of probed features on accuracy (*O*-Moderation).

$m^\dagger$	ChatGLM3		LLaMA2		Falcon		Dolly		Vicuna	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
1	85.356	0.9313	87.279	0.9475	86.127	**0.9371**	81.473	0.8970	85.422	0.9329
3	85.906	0.9361	87.304	0.9488	85.957	0.9307	**81.613**	**0.8975**	85.901	0.9365
5	85.931	0.9368	87.740	**0.9498**	86.267	0.9330	79.385	0.8858	86.043	0.9369
7	86.488	0.9399	87.430	0.9500	**86.478**	0.9353	66.682	0.8789	85.722	0.9342
9	**86.566**	**0.9404**	**87.762**	0.9504	85.932	0.9317	64.860	0.8686	**86.394**	**0.9399**

†: m denotes the number of probed features, as mentioned in §4.1.

Table 15: Impact of the number of probed features on FPR and FNR (*IO*-Moderation).

$m^\dagger$	ChatGLM3		LLaMA2		Falcon		Dolly		Vicuna	
	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR
1	0.865	1.204	0.510	0.289	1.117	0.507	1.856	2.361	0.817	1.284
3	0.904	1.293	0.202	0.525	0.424	0.968	1.365	2.578	0.567	1.538
5	0.827	1.239	0.548	0.226	0.616	0.724	1.269	2.252	0.510	1.393
7	0.760	1.230	0.211	0.588	0.809	0.597	1.385	2.062	0.519	1.474
9	0.481	1.483	0.452	0.262	0.347	0.977	1.009	2.044	0.567	1.339

†: m denotes the number of probed features, as mentioned in §4.1.

Table 16: Impact of the number of probed features on FPR and FNR (*O*-Moderation).

$m^\dagger$	ChatGLM3		LLaMA2		Falcon		Dolly		Vicuna	
	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR
1	13.167	12.850	11.980	11.350	11.485	15.515	16.625	16.304	11.671	13.023
3	12.618	13.337	12.900	11.185	13.709	13.042	16.974	15.743	13.186	11.391
5	12.708	13.394	11.502	12.611	13.200	13.677	16.795	16.172	11.283	13.304
7	12.519	12.702	12.668	12.034	12.662	13.883	18.828	14.647	13.306	11.721
9	12.230	13.007	11.900	12.224	12.901	13.891	15.748	16.642	11.831	12.504

†: m denotes the number of probed features, as mentioned in §4.1.

Table 17: Impact of the number of layers in classifier on accuracy (*IO*-Moderation).

#Layer†	ChatGLM3		LLaMA2		Falcon		Dolly		Vicuna	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
1	98.220	0.9985	99.525	0.9999	99.165	0.9995	97.544	0.9967	98.630	0.9990
3	98.914	**0.9993**	99.627	**0.9999**	**99.226**	**0.9996**	98.043	**0.9994**	**98.951**	**0.9994**
5	98.914	0.9990	**99.669**	0.9999	99.160	0.9995	98.015	0.9976	98.826	0.9993
7	98.914	0.9986	99.632	0.9999	99.133	0.9994	**98.103**	0.9978	98.812	0.9990
9	**98.942**	0.9984	99.501	0.9999	99.105	0.9993	98.001	0.9951	98.551	0.9986

†: the number of layers used in the classifier.

Table 18: Impact of the number of layers in classifier on accuracy (*O*-Moderation).

#Layer†	ChatGLM3		LLaMA2		Falcon		Dolly		Vicuna	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
1	79.655	0.8819	85.427	0.9313	82.968	0.9098	76.284	0.8457	82.621	0.9069
3	**85.356**	**0.9313**	**87.279**	**0.9475**	**86.127**	**0.9371**	**81.473**	**0.8970**	85.422	0.9329
5	84.567	0.9253	87.251	0.9466	85.654	0.9326	81.049	0.8940	**85.592**	**0.9331**
7	77.718	0.9197	84.100	0.9423	83.718	0.9324	77.038	0.8723	82.651	0.9199
9	82.353	0.9027	85.127	0.9317	84.278	0.9327	51.084	0.4611	81.118	0.8947

†: the number of layers used in the classifier.

Table 19: Impact of the number of layers in classifier on FPR and FNR (*IO*-Moderation).

#Layer†	ChatGLM3		LLaMA2		Falcon		Dolly		Vicuna	
	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR
1	1.106	2.361	0.154	0.778	1.059	0.669	2.481	2.750	0.923	1.791
3	0.865	1.203	0.510	0.289	1.117	0.507	1.856	2.361	0.817	1.284
5	0.971	1.122	0.385	0.389	0.703	0.778	1.433	2.578	0.615	1.764
7	0.952	1.312	0.212	0.552	0.510	0.941	1.375	2.641	0.884	1.529
9	0.904	1.239	0.096	1.257	0.510	1.067	1.394	3.148	0.336	2.813

†: the number of layers used in the classifier.

Table 20: Impact of the number of layers in classifier on FPR and FNR (*O*-Moderation).

#Layer†	ChatGLM3		LLaMA2		Falcon		Dolly		Vicuna	
	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR
1	15.290	15.537	10.874	13.930	16.032	13.207	21.898	18.183	12.309	15.504
3	13.167	12.850	11.980	11.350	11.485	15.515	16.625	16.304	11.671	13.023
5	14.233	12.949	11.851	11.482	16.391	11.962	17.403	15.488	12.070	12.842
7	7.316	20.895	7.665	16.914	9.621	17.172	14.163	18.595	10.914	14.037
9	7.246	19.403	5.273	20.673	6.710	22.242	11.223	22.923	5.033	22.412

†: the number of layers used in the classifier.

E MORE RESULTS AGAINST JAILBREAKING

Table 21: LEGILIMENS w/o data augmentation against LLM-targeted static jailbreaking (*IO*-Moderation).

Jailbreaking Type	ChatGLM3		LLaMA2		Falcon		Dolly		Vicuna	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Pretending	98.695	0.9988	99.748	1.0000	99.048	0.9997	92.786	0.9936	98.979	0.9988
Attention Shifting	98.322	0.9988	99.529	0.9998	91.751	0.9952	85.986	0.9887	89.696	0.9882
Privilege Escalation	99.045	0.9991	99.758	0.9999	97.988	0.9981	87.808	0.9806	90.973	0.9975
Syntactic Transformation	90.311	0.9868	80.048	0.9866	92.664	0.9878	76.143	0.9583	78.375	0.9649
Overall	96.808	0.9963	94.715	0.9968	95.371	0.9944	85.599	0.9798	89.640	0.9877

Table 22: LEGILIMENS w/o data augmentation against LLM-targeted static jailbreaking on FPR and FNR (*IO*-Moderation).

Jailbreaking Type	ChatGLM3		LLaMA2		Falcon		Dolly		Vicuna	
	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR
Pretending	0.778	2.026	0.481	0.027	1.021	0.344	2.067	8.520	0.644	1.357
Attention Shifting	0.778	1.872	0.481	0.470	1.167	12.094	1.904	13.323	0.644	18.723
Privilege Escalation	0.778	0.841	0.481	0.009	1.051	2.307	1.990	21.717	0.644	13.459
Syntactic Transformation	0.778	18.325	0.481	37.762	1.080	8.847	2.106	36.442	0.644	39.390
Overall	0.846	5.436	0.490	9.370	1.312	5.130	1.952	22.386	0.884	16.706

Table 23: Robustness against moderator-targeted dynamic jailbreaking (*I*-Moderation).

Method	Type†	Host Model (Attack Success Rate, %)				
		ChatGLM3	LLaMA2	Falcon	Dolly	Vicuna
VIPER	Blind	0	0	0	0	0
SCPN	Blind	0	0	0	0	0
GAN	Decision	0	0	0	0	0

†: Blind attacks are ignorant of the victim model, and decision-based attacks require the final decision for optimization [90].

F TIME COMPLEXITY

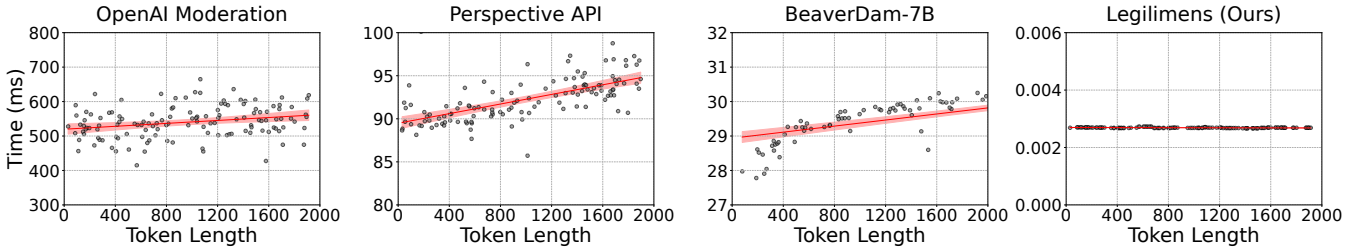

Figure 4: The time complexity of LEGILIMENS compared with three baselines. LEGILIMENS exhibits a constant complexity of $O(1)$, while the other methods exhibits an approximately linear complexity of $O(n)$.

G OVERVIEW OF DATASET

Table 24: Overview of datasets.

Dataset	Alias	#Train	#Test	Task
HateXplain [48]	HAT	12,578	3,050	*I*
Measuring Hate Speech [38]	MHS	16,566	4,142	*I*
OIG-Safety [53]	OIG	19,769	2,000	*I*
Jigsaw [28]	JIG	239,204	59,801	*I*
BeaverTails [36]	BEA	88,657	22,165	*O*
BeaverTails-*adv*	BEA-*adv*	88,657	22,165	*O*
BEA&AG	BAG	86,162	21,457	*IO*
BEA-*adv*&AG	BAG-*adv*	86,162	21,457	*IO*
BEA&PIQA	BPI	26,180	6,546	*I*
HarmBench [49]	HAB	-	320	*I*
SimpleSafetyTests [76]	SST	-	100	*I*
MaliciousInstructions [6]	MAI	-	100	*I*
JADE [91]	JAD	-	80	*I*
HExPHI [63]	HEP	-	330	*I*
TDCRedTeaming [49]	TDC	-	50	*I*
AdvBench [12]	-	-	50	*IO*
AdvBEA	-	-	30	*IO*