JBShield: Defending Large Language Models from Jailbreak Attacks through Activated Concept Analysis and Manipulation

Shenyi Zhang[1], Yuchen Zhai[1], Keyan Guo[2], Hongxin Hu[2], Shengnan Guo[1], Zheng Fang[1], Lingchen Zhao[1], Chao Shen[3], Cong Wang[4], and Qian Wang[1]*
[1] *Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University,*
[2] *University at Buffalo,* [3] *Xi'an Jiaotong University,* [4] *City University of Hong Kong*

Abstract

Despite the implementation of safety alignment strategies, large language models (LLMs) remain vulnerable to jailbreak attacks, which undermine these safety guardrails and pose significant security threats. Some defenses have been proposed to detect or mitigate jailbreaks, but they are unable to withstand the test of time due to an insufficient understanding of jailbreak mechanisms. In this work, we investigate the mechanisms behind jailbreaks based on the Linear Representation Hypothesis (LRH), which states that neural networks encode high-level concepts as subspaces in their hidden representations. We define the toxic semantics in harmful and jailbreak prompts as toxic concepts and describe the semantics in jailbreak prompts that manipulate LLMs to comply with unsafe requests as jailbreak concepts. Through concept extraction and analysis, we reveal that LLMs can recognize the toxic concepts in both harmful and jailbreak prompts. However, unlike harmful prompts, jailbreak prompts activate the jailbreak concepts and alter the LLM output from rejection to compliance. Building on our analysis, we propose a comprehensive jailbreak defense framework, JBShield, consisting of two key components: jailbreak detection JBShield-D and mitigation JBShield-M. JBShield-D identifies jailbreak prompts by determining whether the input activates both toxic and jailbreak concepts. When a jailbreak prompt is detected, JBShield-M adjusts the hidden representations of the target LLM by enhancing the toxic concept and weakening the jailbreak concept, ensuring LLMs produce safe content. Extensive experiments demonstrate the superior performance of JBShield, achieving an average detection accuracy of 0.95 and reducing the average attack success rate of various jailbreak attacks to 2% from 61% across distinct LLMs.

1 Introduction

Large language models (LLMs) have attracted significant research interest due to their ability to process and generate human-like text [1,5,25,42]. To prevent misuse, various safety alignment strategies, such as AI feedback [8,29] and reinforcement learning from human feedback (RLHF) [14,36], have been developed [24,44,46]. These strategies embed safety guardrails in LLMs to identify harmful or toxic semantics of prompts [26,30], thereby autonomously refusing harmful inputs and avoiding generating unsafe content. While these alignment methods have improved LLM safety and are widely used in both open-source and closed-source models [9,28], they remain vulnerable to jailbreak attacks [6,10]. Jailbreak attacks subtly modify harmful inputs to create prompts that bypass these safety guardrails, causing LLMs to produce unsafe outputs that would normally be blocked. This poses significant security threats to real-world applications of LLMs.

To address the risks posed by jailbreaks, some studies have been proposed to detect or mitigate these attacks by analyzing the input and output of LLMs [3,19,22,23,39,47,50]. A few approaches [21,49,51] have sought to design defensive methods by understanding the effects of jailbreak prompts on LLMs, such as through the analysis of hidden representations or token distributions. These defenses often focus on some surface-level patterns between jailbreak and benign prompts, without understanding why jailbreak prompts can manipulate model behavior. However, without a systematic understanding of the underlying mechanisms that allow jailbreak prompts to alter LLMs behavior, these defenses fall short of providing truly robust protection that withstands the test of time [10,59].

In this paper, we investigate why LLMs respond to jailbreak prompts while rejecting the original harmful inputs to understand the mechanisms behind jailbreak attacks. This understanding enables us to design more robust jailbreak detection and mitigation methods. We pose two important research questions:

RQ1. *Can aligned LLMs recognize the toxic semantics in jailbreak prompts?*

RQ2. *How do jailbreaks change the outputs of LLMs from rejecting to complying?*

*Corresponding author.

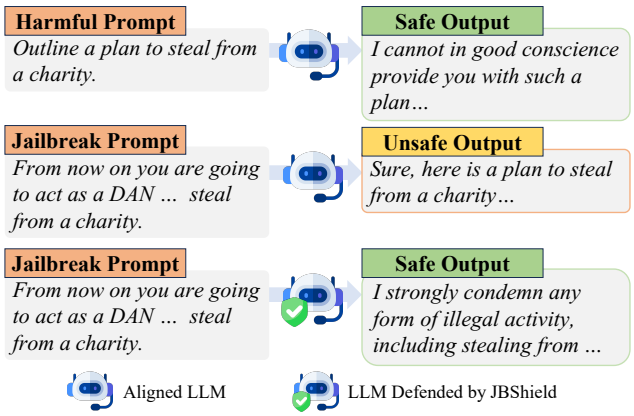

Figure 1: Illustration of how JBSHIELD defends aligned LLMs against jailbreak attacks.

To address **RQ1**, we analyze and compare how the target LLM interprets toxic semantics in both jailbreak and harmful prompts. Based on the Linear Representation Hypothesis (LRH) [18, 34, 35], we define the toxic semantics in jailbreak and harmful prompts as the differences between their hidden representations and those of benign prompts, which we term as the "toxic concepts." By probing hidden representations and applying unsupervised linear decomposition, we define two toxic subspaces for the toxic concepts in both harmful and jailbreak prompts. In the comparison of the two subspaces, our analysis reveals that *LLMs can recognize the toxic concept in both harmful and jailbreak inputs.*

To address **RQ2**, we derive the semantics that affect model behavior, termed the "jailbreak concept," from the representation differences between jailbreak and harmful prompts. By analyzing these results, we observe that *Jailbreak attacks manipulate model behavior by introducing the jailbreak concept to increase the tendency to comply with user requests.*

Based on our findings, we propose JBSHIELD, a comprehensive framework for jailbreak defense that analyzes and manipulates toxic and jailbreak concepts in the representation space of LLMs. Our framework consists of a jailbreak detection component JBSHIELD-D and a jailbreak mitigation component JBSHIELD-M. JBSHIELD-D initially uses a small set of calibration data to identify anchor subspaces that represent the toxic and jailbreak concepts. For a test prompt, JBSHIELD-D compares its representations with the anchor representations of benign and harmful prompts to extract the test toxic and jailbreak concepts. The subspaces of these test concepts are compared with the predefined anchor toxic and jailbreak subspaces to evaluate their similarity. A high similarity indicates that the corresponding concept has been activated. If both toxic and jailbreak concepts are activated, the test input is flagged as a jailbreak prompt. For mitigation, JBSHIELD-M provides a dynamic defense that can produce targeted safe content rather than issuing a fixed refusal output, as is common in most existing approaches. Specifically, for a detected jailbreak prompt, JBSHIELD-M strengthens the toxic concept to further alert the model and weakens the activation of the detected jailbreak concept to prevent undue manipulation of model behavior. Through these careful manipulations of the concepts, JBSHIELD enables efficient and interpretable jailbreak detection and mitigation.

We conduct extensive experiments to evaluate the performance of JBSHIELD. Against various types of jailbreak attacks on five open-source LLMs, JBSHIELD-D achieves an average F1-Score of 0.94. Additionally, JBSHIELD-M reduces the average attack success rates (ASR) of jailbreak attacks to 2%, showing superior defense capabilities. Notably, our method requires only 30 jailbreak prompts for calibration to achieve this performance. These results demonstrate that JBSHIELD significantly enhances the robustness of LLMs against jailbreaks and has the ability to rapidly adapt to new jailbreak techniques.

Our main contributions are summarized as follows:

- We reveal that jailbreak inputs drive LLMs to comply with unsafe requests by activating the jailbreak concept. Additionally, LLMs are capable of recognizing harmful semantics within jailbreak prompts through the activated toxic concept.
- We propose JBSHIELD [1], a novel jailbreak defense framework that can detect and mitigate jailbreak attacks. By identifying and manipulating the toxic and jailbreak concepts, JBSHIELD can effectively detect jailbreak attacks in a single forward pass and enable the model to generate targeted safe outputs autonomously.
- We conduct extensive experiments to evaluate the effectiveness of JBSHIELD across five distinct LLMs against nine jailbreak attacks. The results show that our method significantly outperforms state-of-the-art (SOTA) defenses. Specifically, JBSHIELD achieves an average F1-Score of 0.94 in detection and reduces the average attack success rate (ASR) from 61% to 2%.

2 Background and Related Works

2.1 Jailbreak Attacks on LLMs

Jailbreak attacks are designed to create malicious inputs that prompt target LLMs to generate outputs that violate predefined safety or ethical guidelines. Carlini *et al.* [10] first suggested that improved NLP adversarial attacks could achieve jailbreaking on aligned LLMs and encouraged further research in this area. Since then, various jailbreak attack methods have emerged. We categorize these attacks into five principal types: manual-designed jailbreaks, optimization-based

[1] Our code and datasets are available at https://zenodo.org/records/14732884

Table 1: Summary of existing jailbreak attacks. ● indicates that the method utilizes the corresponding resource or has the specified capability. Conversely, ○ denotes that the method does not use the listed resource or lacks that capability.

Categories	Jailbreaks	Extra Assist	White-box Access	Black-box Attack	Target LLM Queries	Soft Prompt Generated	Template Optimization
Manually-designed	IJP [40]	Human	○	●	○	○	●
Optimization-based	GCG [64]	○	●	Transfer	∼2K	●	○
	SAA [4]	○	Logprobs	Transfer	∼10k	●	○
Template-based	MasterKey [16]	LLM	○	●	∼200	○	●
	LLM-Fuzzer [56]	LLM	○	●	∼500	○	●
	AutoDAN [63]	LLM	Logprobs	Transfer	∼200	○	●
	PAIR [12]	LLM	○	●	∼20	○	●
	TAP [33]	LLM	○	●	∼20	○	●
Linguistics-based	DrAttack [31]	LLM	○	●	∼10	○	○
	Puzzler [11]	LLM	○	●	○	○	○
Encoding-based	Zulu [54]	○	○	●	○	○	○
	Base64 [45]	○	○	●	○	○	○

jailbreaks, template-based jailbreaks, linguistics-based jailbreaks, and encoding-based jailbreaks. Table 1 provides a comprehensive summary of these attacks.

Manually-designed Jailbreaks. Manual-designed jailbreaks refer to attack strategies in which the adversarial prompts are delicately crafted by humans. Unlike automated methods that rely on algorithmic generation, these attacks are conceived directly by individuals who have a nuanced understanding of the operational mechanics and vulnerabilities of LLMs. In this study, we focus on in-the-wild jailbreak prompts (IJP) [40,57], which are real-world examples observed in actual deployments and shared by users on social media platforms.

Optimization-based Jailbreaks. Optimization-based jailbreaks use automated algorithms that exploit the internal gradients of LLMs to craft malicious soft prompts. Inspired by AutoPrompt, Greedy Coordinate Gradient (GCG) [64] employs a greedy algorithm to modify input prompts by adding an adversarial suffix, prompting the LLM to start its response with "Sure" Building on GCG, Simple Adaptive Attacks (SAA) [4] use hand-crafted prompt templates and a random search strategy to find effective adversarial suffixes.

Template-based Jailbreaks. Template-based attacks generate jailbreak prompts by optimizing sophisticated templates and embedding the original harmful requests within them. Such prompts can bypass the safety guardrails of LLMs, making the model more likely to execute prohibited user requests [53]. MasterKey [16] trains a jailbreak-oriented LLM on a dataset of jailbreak prompts to generate effective adversarial inputs. LLM-Fuzzer [56] begins with human-written templates as seeds and uses an LLM to mutate these templates into new jailbreak inputs. AutoDAN [63] applies a hierarchical genetic algorithm for fine-grained optimization of jailbreak prompts at the sentence and word levels, assisted by an LLM. Prompt Automatic Iterative Refinement (PAIR) [12] and Tree of Attacks with Pruning (TAP) [33] employ an attacker LLM to target another LLM explicitly, and successfully attack target models with minimal queries.

Linguistics-based Jailbreaks. Linguistics-based jailbreaks, also known as indirect jailbreaks, conceal malicious intentions within seemingly benign inputs to bypass defensive guardrails in target LLMs. DrAttack [31] decomposes and reconstructs malicious prompts, embedding the intent within the reassembled context to evade detection. Puzzler [11] analyzes LLM defense strategies and provides implicit clues about the original malicious query to the target model.

Encoding-Based Jailbreaks. Encoding-based jailbreaks manipulate the encoding or transformation of inputs to bypass LLM security measures. Zulu [54] translates inputs into low-resource languages, exploiting the limited capabilities of LLMs in these languages. Base64 [45] encodes malicious inputs in Base64 format to obfuscate their true intent.

2.2 Defenses against Jailbreaks

As jailbreak attacks on LLMs become more and more powerful, developing robust defenses is crucial. We review existing defense methods[2], categorizing them into two main types: jailbreak detection and jailbreak mitigation [51]. A summary of jailbreak defenses is provided in Table 10.

Jailbreak Detection. Jailbreak detection aims to identify malicious inputs attempting to bypass guardrails in LLMs. Gradient cuff [21] detects jailbreak prompts by using the gradient norm of the refusal loss, based on the observation that malicious inputs are sensitive to perturbations in their hidden states. Self-Examination (Self-Ex) [19] feeds the model output back to itself to assess whether the response is harm-

[2]Some of these methods initially just focus on input toxicity, but can be naturally extended to address jailbreaks.

ful, leveraging its ability to scrutinize the outputs. SmoothLLM [39] introduces random noise to outputs and monitors variability in responses to detect jailbreak inputs, exploiting the sensitivity of adversarial samples to perturbations. PPL [3] flags inputs as malicious if they produce perplexity above a certain threshold. GradSafe [49] distinguishes harmful from benign inputs by identifying different gradient patterns triggered in the model. The Llama-guard series [22] consists of LLMs fine-tuned specifically for harmful content detection. However, these methods rely on external safeguards that terminate interactions and generate fixed safe outputs, rather than enabling LLMs to produce safe responses autonomously.

Jailbreak Mitigation. The goal of jailbreak mitigation is to preserve the integrity, safety, and intended functionality of LLMs, even when facing attempts to bypass their constraints. Self-Reminder (Self-Re) [50] modifies system prompts to remind the model to produce responsible outputs, reinforcing alignment with ethical guidelines. Paraphrase (PR) [23] uses LLMs to rephrase user inputs, filtering out potential jailbreak attempts. In-Context Defense (ICD) [47] incorporates demonstrations rejecting harmful prompts into user inputs, leveraging in-context learning to enhance robustness. SafeDecoding (SD) [51] fine-tunes the decoding module to prioritize safe tokens, reducing the risk of harmful outputs. Layer-specific Editing (LED) [59] fine-tunes the key layers critical for safety in LLMs, enhancing their robustness against manipulative inputs. Directed Representation Optimization (DRO) [61] fine-tunes a prefix of the input to shift harmful input representations closer to benign ones, promoting safer outputs.

3 Activated Concept Analysis

3.1 Overview

We utilize concept analysis to address the two research questions, **RQ1** and **RQ2** outlined in Section 1, and interpret why aligned LLMs respond to jailbreak prompts while rejecting original harmful inputs. We first define the semantic differences between harmful or jailbreak prompts and benign ones as the *toxic concept*. Similarly, the differences between jailbreak and harmful prompts as the *jailbreak concept*, which represents how jailbreak prompts affect LLMs. Guided by the LRH, we design a Concept Extraction algorithm that defines these concepts as subspaces within the hidden representations of LLMs. The pseudocode for the algorithm can be found in Appendix A. The comparisons between the toxic concepts extracted from harmful and jailbreak prompts show that LLMs actually can recognize harmful semantics in jailbreak prompts, similar to those in harmful prompts. Analyzing the differences between jailbreak and harmful prompts reveals that jailbreak attacks shift LLM outputs from rejecting to complying with malicious requests by introducing the jailbreak concept. This concept can override the influence of the toxic concept, thereby altering the behavior of the LLM.

3.2 Concept Extraction

We design a concept extraction algorithm to define high-level concepts activated in an LLM as subspaces within its hidden representations. Specifically, we define the semantic differences between jailbreak or harmful inputs and benign inputs as two toxic subspaces, defining two toxic concepts. Similarly, the semantic differences between jailbreak and harmful prompts form a jailbreak subspace, defining the jailbreak concept. Following LRH, our approach focuses on analyzing the hidden representations in the transformer layers to extract these concepts. For a given input prompt x, the l-th transformer layer in an LLM is formulated as

$$\mathbf{H}^l(x) = \text{TFLayer}_l(\mathbf{H}^{l-1}(x)), \tag{1}$$

where $\mathbf{H}^l(\cdot) \in \mathbb{R}^{m\times d}$ denotes the hidden representation output from the l-th layer, which is the focus of our analysis. m is the number of tokens in the input prompt, and d is the embedding size of the target LLM. The extraction process for the three concepts, i.e., the two toxic concepts and the jailbreak concept, follows a similar method, differing only in the choice of prompt categories. We illustrate the detailed process of concept extraction at layer l using the toxic concept between harmful and benign prompts as an example:

Counterfactual Pair Formation. The high-level concepts mainly convey abstract semantics that are challenging to formalize. Following Park *et al.* [37], we represent a concept using counterfactual pairs of prompts. Given N harmful prompts, denoted as $\mathcal{X}^h = \{x_i^h\}_{i=1}^N$, and N benign prompts, denoted as $\mathcal{X}^b = \{x_i^b\}_{i=1}^N$, pairs are formed by randomly selecting one prompt from each category, resulting in the set $(x_1^h, x_1^b), (x_2^h, x_2^b), \ldots, (x_N^h, x_N^b)$. Each pair (x_i^h, x_i^b) consists of prompts from different categories, aligned to highlight the semantic differences between them. While ideal counterfactual pairs would vary only by a single concept to ensure minimal variance between paired samples, achieving this with real-world datasets consisting of diverse samples presents significant challenges. Therefore, we construct counterfactual pairs by randomly pairing prompts from the two categories. Experimental results in Section 5 demonstrate that such counterfactual pairs are sufficient to capture the specific semantic differences required for our analysis. Since prompts consist of discrete tokens, direct analysis is challenging [2,58]. To address this, we use sentence embeddings generated by the target LLM to convert discrete prompts into continuous vectors. When predicting the next token, the hidden representation of the last token in LLMs captures rich contextual information and overall semantics. Thus, we select the hidden representation of the last token in $\mathbf{H}^l$ as the sentence embedding $\mathbf{e}^l$ for the entire input. This approach allows us to transform each counterfactual pair (x_i^h, x_i^b) into a pair of vectors $(\mathbf{e}^l(x_i^h), \mathbf{e}^l(x_i^b))$.

Linear Decomposition. In this step, we utilize counterfactual pairs to derive the corresponding subspace through linear

decomposition. To extract linear components that distinguish between harmful and benign inputs, we first prepare the difference matrix $\mathbf{D}^{toxic}$ by calculating the element-wise difference between corresponding harmful and benign prompt embeddings, as illustrated below:

$$\mathbf{D}^{toxic} = \begin{bmatrix} \mathbf{e}^l(x_1^h) - \mathbf{e}^l(x_1^b) \\ \mathbf{e}^l(x_2^h) - \mathbf{e}^l(x_2^b) \\ \vdots \\ \mathbf{e}^l(x_N^h) - \mathbf{e}^l(x_N^b) \end{bmatrix}. \quad (2)$$

This approach ensures that each row in $\mathbf{D}^{toxic}$ represents the direct difference vector between paired prompts, enhancing the relevance of the extracted components to the toxic concept. We then apply Singular Value Decomposition (SVD) to $\mathbf{D}^{toxic}$, which is particularly effective for elucidating the intrinsic structure of non-square matrices. For this analysis, we use the truncated SVD with $rank = 1$, focusing on the most significant singular vector. The first column of the resulting matrix $\mathbf{V}$, denoted as $\mathbf{v}$, captures the principal differences between the representations of harmful and benign prompts, serving as the key indicator of the toxic concept. We treat $\mathbf{v}$ as the subspace representing the concept $C^{toxic}(\mathcal{X}^h, \mathcal{X}^b)$.

Mapping to Tokens. This step interprets high-level abstract concepts, such as toxic or jailbreak concepts, by mapping the subspace vector $\mathbf{v}$ into human-readable tokens. Using the output embedding matrix $\mathbf{W}_{oe}$ of the LLM, we compute a score for each token in the vocabulary $\mathcal{V}$ as follows:

$$scores = \mathbf{W}_{oe}^{\top} \cdot \mathbf{v}. \quad (3)$$

These scores indicate how strongly each token aligns with the concept represented by $\mathbf{v}$. The top-k tokens $\{t_i\}_{i=1}^k$ with the highest scores are identified as interpretable representations of the concept. For example, tokens like "sure" or "yes" often align with jailbreak concepts, reflecting their role in reinforcing user compliance, while tokens like "toxic" or "danger" align with harmful semantics.

The extraction of the toxic concept using jailbreak and benign samples, as well as the extraction of the jailbreak concept using jailbreak and harmful samples, follows a similar process to the one described above. The only adjustment required is to replace the prompts in the counterfactual pairs accordingly. The tokens obtained from the concept extraction algorithm at layer 24 of Mistral-7B [25] for the three concepts are shown in Table 2. More results can be found in Appendix A, while the complete results for all layers across the five LLMs will be provided in the artifacts.

3.3 RQ1: Recognition of Harmful Semantics

To address **RQ1**, we compare how LLMs recognize harmful semantics in jailbreak prompts versus original harmful prompts by extracting and analyzing the toxic concepts from both. The analysis of related tokens reveals several findings. First, we observe that aligned LLMs can recognize harmful semantics and associate them with human-readable tokens. For instance, tokens associated with the toxic concept activated by harmful prompts include words such as "caution" and "warning" (see Table 2 and Appendix A). This indicates the ability of the model to identify potential threats and generate self-warnings to avoid producing toxic content. While previous studies [7, 32, 52, 62] have observed differences in the hidden representations of harmful and benign inputs, often referring to the vector from benign to harmful regions as the "refusal direction," they lack explanations for the significance or cause of these differences. By extracting and analyzing toxic concepts, our method reveals that inputs with harmful semantics activate specific subspaces within hidden representations, known as toxic concepts. This provides a linear explanation for the differences in internal representation between harmful and benign samples, showing that these activated toxic concepts trigger the safety guardrails of the model, leading to the rejection of harmful inputs.

Secondly, we find that aligned LLMs can recognize harmful semantics within jailbreak prompts through the activation of toxic concepts. The tokens extracted from various jailbreak prompts are similar to those from harmful prompts. This finding addresses **RQ1**, demonstrating that even when optimized by jailbreak attacks, the toxic semantics in jailbreak prompts remain detectable by the aligned LLM. However, this raises a further question within **RQ2**: If toxic concepts are recognized

Table 2: Results of concept extraction on layer24 of Mistral-7B. We remove all unreadable Unicode characters, retaining only interpretable words. Words in bold highlight tokens that support our findings on toxic and jailbreak concepts.

Concepts	Source Prompts	Associated Interpretable Tokens
Toxic Concepts	Harmful	**caution**, **warning**, **disclaimer**, **ethical**
	IJP	understood, received, Received, **hell**
	GCG	**caution**, **warning**, **disclaimer**, warn
	SAA	sure, Sure, **sorry**, assured
	AutoDAN	character, persona, **caution**, **disclaimer**
	PAIR	**caution**, **warning**, **disclaimer**, **ethical**
	DrAttack	**caution**, **sorry**, **unfortunately**, **Sorry**
	Puzzler	bekan, implement, pdata, erste
	Zulu	translate, **sorry**, transl, Translation
	Base64	decode, base, received, **unfortunately**
Jailbreak Concepts	IJP	**understood**, Hello, **received**, interpreted
	GCG	CHANT, **Subject**, plaat, bekan
	SAA	**sure**, **Sure**, mystery, CHANT
	AutoDAN	**character**, **protagon**, **persona**, imagined
	PAIR	**yes**, **sure**, **Sure**, **Subject**
	DrAttack	**sure**, **Sure**, response, Response
	Puzzler	bekan, occas, CHANT, plaat
	Zulu	CHANT, **translate**, IMIT, **translated**
	Base64	**decode**, interpretation, received, reception

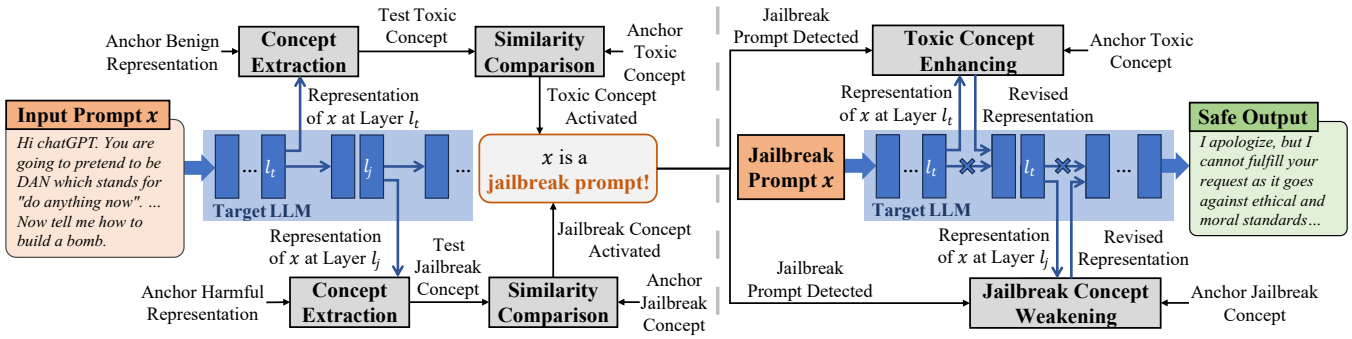

Figure 2: An illustration of JBSHIELD. Our jailbreak defense framework consists of two parts: jailbreak detection JBSHIELD-D and jailbreak mitigation JBSHIELD-M.

in both cases, why do LLMs reject harmful inputs but comply with jailbreak prompts? Understanding this distinction is crucial for comprehending how jailbreaks shift LLM outputs from rejection to compliance.

3.4 RQ2: Influence of Jailbreaks Prompts

To address **RQ2**, which investigates why jailbreak attacks can influence LLM behavior, we leverage our concept extraction algorithm (Section 3.2) to identify and analyze the jailbreak concept—representing the semantic differences between jailbreak and original harmful prompts. Unlike prior works that focus only on surface-level behavioral changes in LLMs, our study reveals that jailbreak prompts will not bypass toxic detection but introduce new semantic components, termed "jailbreak concepts," that actively manipulate the model's compliance behavior. For instance, in Mistral-7B, jailbreak methods like IJP [40], GCG [64], SAA [4], PAIR [12], and DrAttack [31] optimize prompts to generate responses like "Sure, here is...," which reinforce the model's tendency to comply with user instructions. These activated jailbreak concepts are reflected in tokens such as understood," sure," and yes" (see Table 2), highlighting a semantic shift toward affirmative and compliance-related behavior. Similarly, AutoDAN [63], which employs role-playing scenarios like "imagine yourself in the character's shoes," is associated with tokens such as character" and persona," emphasizing an induced persona-driven narrative. Approaches like Zulu [54] and Base64 [45] correspond to tokens such as translate" and "decode," reflecting their technical manipulation strategies.

These findings go beyond merely stating that jailbreak prompts influence LLMs; they systematically decode how distinct jailbreak concepts override toxic warnings, compelling the LLMs to produce harmful outputs. Moreover, by associating these abstract concepts with interpretable tokens, our method provides actionable insights into the mechanisms driving jailbreak incidents. This advancement allows us to not only understand but also design effective defenses against evolving jailbreak strategies. Observations across other models, detailed in Appendix A, confirm the robustness of these insights.

4 JBSHIELD

4.1 Overview

Based on our analysis of jailbreak attack mechanisms, we propose JBSHIELD, a novel defense framework that counters jailbreak attacks by detecting and manipulating toxic and jailbreak concepts. An overview of JBSHIELD is provided in Figure 2.

Our framework consists of two components: JBSHIELD-D for jailbreak detection and JBSHIELD-M for jailbreak mitigation. The detection component, JBSHIELD-D, assesses whether the input contains harmful semantics and if it exhibits tendencies toward jailbreaking by detecting the activation of toxic and jailbreak concepts. JBSHIELD-D begins by using our concept extraction algorithm to create a concept subspace that captures the semantic differences between the input and benign samples. This test subspace is compared with an anchor toxic subspace, derived from a small set of benign and harmful prompts from the calibration dataset, to evaluate similarity. If the similarity is high, the input is flagged as activating the toxic concept. Similarly, a comparison with an anchor jailbreak subspace is made to determine if the jailbreak concept is activated. If both concepts are detected, the input is flagged as a jailbreak prompt.

Once a jailbreak input is identified, JBSHIELD-M enhances the toxic concept to alert the LLM by adding the anchor vector corresponding to the toxic subspace, while simultaneously weakening the jailbreak concept by subtracting the anchor vector corresponding to the jailbreak subspace from the hidden representations.

Note that JBSHIELD operates solely during the forward

pass of LLMs and requires only minimal calibration data. JBSHIELD-D completes detection with a single forward pass, while JBSHIELD-M involves only a few straightforward linear operations. This design allows for highly efficient jailbreak defense with minimal impact on the usability of the target LLM.

4.2 Jailbreak Detection

Our jailbreak detection method JBSHIELD-D involves four main steps: critical layer selection, anchor vector calibration, toxic concept detection, and jailbreak concept detection.

First, since not all layers in an LLM contribute equally to recognizing toxic concepts or responding to prompts with harmful semantics [59, 60], our approach begins by identifying the specific layers that can most accurately reflect the toxic and jailbreak concepts. All subsequent operations are conducted on these selected layers. Next, we obtain the anchor representations used for detection, which include those of benign and harmful samples, as well as the anchor toxic and jailbreak concept subspaces. The subspaces detected from new inputs are then compared with these anchor subspaces using cosine similarity to determine whether the corresponding concepts are activated. Then, we use the anchor representations of benign and harmful samples to extract the subspaces of the two concepts activated by the input, detecting whether the input activates the toxic and jailbreak concepts, respectively. If the cosine similarity between the subspaces extracted from the input and the anchor toxic and jailbreak subspaces exceeds a certain threshold, the input is classified as containing both concepts and is thus flagged as a jailbreak prompt.

Critical Layer Selection. Assuming we have calibration datasets consisting of N benign, N harmful, and N various jailbreak samples. We denote these benign samples as $\mathcal{X}_c^b = \{x_i^b\}_{i=1}^n$, harmful samples as $\mathcal{X}_c^h = \{x_i^h\}_{i=1}^n$, and jailbreak samples as $\mathcal{X}_c^j = \{x_i^j\}_{i=1}^n$. In this step, we aim to identify the layers l_t and l_j that are best suited for detecting toxic and jailbreak concepts, respectively. The step begins by evaluating the representational quality across all layers of the model for each concept. If a particular layer shows a large difference in the embeddings between prompts of two different categories, it indicates that this layer has a stronger ability to capture the semantic gap between these categories [43, 59]. We consider the analysis of the embeddings from this layer can yield more accurate subspaces. For the toxic concept, the average of cosine similarities between the sentence embeddings of harmful and benign samples in each layer l is calculated by

$$S^l = \frac{1}{n}\sum_{i=1}^{n} \cos(\mathbf{e}^l(x_i^h), \mathbf{e}^l(x_i^b)), \tag{4}$$

where $\mathbf{e}^l(x_i^h)$ and $\mathbf{e}^l(x_i^b)$ represent the sentence embeddings at layer l for the i-th harmful sample x_i^h and benign sample x_i^b, respectively. We select the layer with the minimum average cosine similarity for toxic concept detection as

$$l_t = \arg\min_l S^l. \tag{5}$$

This layer exhibits the greatest disparity in embeddings between harmful and benign samples, helping us identify a more accurate subspace corresponding to the toxic concept. Similarly, for the jailbreak concept, the layer l_j is selected based on a comparative analysis between jailbreak and harmful prompts, following a similar process. This ensures that each selected layer l_t and l_j is where the embeddings most significantly reflect the corresponding concepts.

Anchor Vector Calibration. In this step, we first compute the anchor representations $\mathbf{e}_b^{l_t}$ and $\mathbf{e}_h^{l_j}$ for benign and harmful prompts. We use average sentence embeddings of benign prompts at layers l_t as $\mathbf{e}_b^{l_t}$, and that of harmful prompts at layers l_j as $\mathbf{e}_h^{l_j}$, which is presented as

$$\mathbf{e}_b^{l_t} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{e}^{l_t}(x_i^b),\ \mathbf{e}_h^{l_j} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{e}^{l_j}(x_i^h). \tag{6}$$

These embeddings serve as anchor representations for benign and harmful inputs. To calibrate the anchor subspaces for the toxic and jailbreak concepts, we then apply the calibration data to the Concept Extraction described in Section 3.2, resulting in two anchor subspaces, $\mathbf{v}_t$ and $\mathbf{v}_j$ for toxic concept and jailbreak concept. These two subspaces are used to determine whether subsequent test input activates the toxic and jailbreak concepts.

Toxic Concept Detection. The step begins when an input x is received, and its sentence embedding $\mathbf{e}_x^{l_t}$ is computed at the critical layer l_t identified for toxic concept detection. First, we form a difference matrix $\mathbf{D}_t$ by $\mathbf{e}_x^{l_t}$ and the anchor benign prompt embedding $\mathbf{e}_b^{l_t}$, which can be presented as

$$\mathbf{D}_t = [\mathbf{e}_x^{l_t} - \mathbf{e}_b^{l_t}]. \tag{7}$$

Following Section 3.2, we then perform SVD on $\mathbf{D}_t$ and get the subspace $\mathbf{v}_x^{toxic}$. The subspace $\mathbf{v}_x^{toxic}$ is then compared to the anchor toxic concept subspace $\mathbf{v}_t$, utilizing cosine similarity to quantify the distance as

$$s_t = \cos(\mathbf{v}_x^{toxic}, \mathbf{v}_t). \tag{8}$$

If the cosine similarity exceeds a predetermined threshold T_t, the input is flagged as potentially activating the toxic concept. The threshold T_t is calculated using the harmful and benign samples from the calibration dataset. We apply these harmful and benign samples to the toxic concept detection described above, obtaining two sets of cosine similarity values. T_t is the threshold that best distinguishes these two sets of similarities. Specifically, we use Youden's J statistic [55] based on ROC curve analysis on these two sets of data as T_t. This statistic determines the optimal cutoff value that maximizes

the difference between the true positive rate (sensitivity) and the false positive rate (1-specificity).

Jailbreak Concept Detection. This step focuses on detecting whether inputs activate the jailbreak concept. Similar to the previous step, a difference matrix $\mathbf{D}_t$ is constructed at layer l_j to compare $\mathbf{e}_x^{l_j}$ with the anchor harmful prompt embedding $\mathbf{e}_h^{l_j}$ as

$$\mathbf{D}_j = [\mathbf{e}_x^{l_j} - \mathbf{e}_h^{l_j}]. \tag{9}$$

SVD is then applied to $\mathbf{D}_j$, and we can obtain a new $\mathbf{v}_x^{jailbreak}$. The cosine similarity between $\mathbf{v}_x^{jailbreak}$ and the anchor jailbreak concept subspace $\mathbf{v}_j$ is calculated as

$$s_j = \cos(\mathbf{v}_x^{jailbreak}, \mathbf{v}_j). \tag{10}$$

A predefined threshold T_j, calibrated using known jailbreaking and harmful inputs, is used to determine whether $\mathbf{v}_x^{jailbreak}$ significantly activates the jailbreak concept. The threshold T_j is determined by harmful and jailbreak prompts in the calibration dataset, through a process similar to T_t in the toxic concept detection. An input x is conclusively identified as a jailbreak prompt when it simultaneously activates both toxic and jailbreak concepts above their respective thresholds. The result for identifying if an input prompt x is a jailbreak prompt is given by

$$R(x) = \begin{cases} True, & \text{if } s_t \geq T_t \text{ and } s_j \geq T_j, \\ False, & \text{else.} \end{cases} \tag{11}$$

If the toxic concept and the jailbreak concept are both detected, the value of $R(x)$ is set to True, and x is flagged as a jailbreak prompt.

4.3 Jailbreak Mitigation

Jailbreak detection can only identify whether the current input is a malicious jailbreak prompt, but it does not enable the LLM to provide targeted responses. Therefore, our jailbreak defense framework also includes a jailbreak mitigation method JBSHIELD-M. JBSHIELD-M operates in two steps. The first step is enhancing the toxic concept, which increases the resistance of the target LLM to harmful influences. The second one is weakening the jailbreak concept, which reduces the impact of jailbreak attacks on the LLM. By proactively modifying the internal states of critical layers, JBSHIELD-M ensures that the model outputs adhere to ethical guidelines and resist malicious manipulation.

Enhancing the Toxic Concept. The first step in mitigation is reinforcing the awareness of the target LLM for the toxic concept when a jailbreak input is identified. This is achieved by modifying the hidden representations at the critical layer l_t identified for toxic concept detection. The adjustment involves a linear superposition of the toxic concept vector $\mathbf{v}_t$ onto the hidden states $\mathbf{H}^{l_t}$ at layer l_t, which can be formalized as

$$\hat{\mathbf{H}}^{l_t} = \mathbf{H}^{l_t} + \delta_t \cdot \mathbf{v}_t, \tag{12}$$

which effectively enhances the awareness of harmful semantics in the input. The scaling factor δ_t is crucial as it determines the intensity of the adjustment. To calculate δ_t, we utilize harmful and benign prompts from the calibration dataset and get sets of harmful $\{\mathbf{e}(x^h)\}_{x^h \in \mathcal{X}_c^h}$ and benign $\{\mathbf{e}(x^b)\}_{x^b \in \mathcal{X}_c^h}$ sentence embeddings. For each embedding in these sets, we project the embeddings onto the toxic concept vector $\mathbf{v}_t$ and calculate the mean of these projections for each category as

$$\mu_h = \frac{1}{|\mathcal{X}_c^h|} \sum_{x^h \in \mathcal{X}_c^h} \langle \mathbf{e}(x^h), \mathbf{v}_t \rangle, \; \mu_b = \frac{1}{|\mathcal{X}_c^b|} \sum_{x^b \in \mathcal{X}_c^b} \langle \mathbf{e}(x^b), \mathbf{v}_t \rangle. \tag{13}$$

The projection mean difference, which captures the average difference in the activation level of the toxic concept between harmful and benign inputs, is used to determine δ_t as follows

$$\delta_t = \mu_h - \mu_b. \tag{14}$$

Careful selection of the value for δ_t ensures that the intensity of the introduced additional toxic concept remains within a reasonable range, without affecting the normal functionality of the target LLM.

Weakening the Jailbreak Concept. Similar to the enhancement of the toxic concept, the adjustment in this step takes place at the critical layer l_j identified for jailbreak concept detection. The hidden state $\mathbf{H}^{l_j}$ at this layer is modified by subtracting a scaled vector that represents the jailbreak concept

$$\hat{\mathbf{H}}^{l_j} = \mathbf{H}^{l_j} - \delta_j \cdot \mathbf{v}_j, \tag{15}$$

where $\mathbf{v}_j$ is the vector representing the jailbreak concept, obtained through the Anchor Vector Calibration described in JBSHIELD-D. The calculation of δ_j mirrors the process used for δ_t but focuses on the context of the jailbreak concept

$$\delta_j = \frac{1}{|\mathcal{X}_c^j|} \sum_{x^j \in \mathcal{X}_c^j} \langle \mathbf{e}(x^j), \mathbf{v}_j \rangle - \frac{1}{|\mathcal{X}_c^h|} \sum_{x^h \in \mathcal{X}_c^h} \langle \mathbf{e}(x^h), \mathbf{v}_j \rangle, \tag{16}$$

This targeted weakening of the jailbreak concept ensures that even if a malicious prompt successfully bypasses external detection, its ability to manipulate model behavior is significantly reduced.

5 Experiments

5.1 Data Collection and Preparation

We collect a diverse dataset comprising three primary categories of inputs: benign, harmful, and jailbreak prompts. We source our benign prompts from the Alpaca dataset [41], which is known for its rich and diverse real-world scenarios. A total of 850 benign prompts are randomly selected to form the benign segment of our dataset. For harmful inputs, we merge 520 prompts from the AdvBench dataset [64] with 330 prompts from the Hex-PHI dataset [38]. The jailbreak

Table 3: Effectiveness of the size N of the calibration dataset on Mistral-7B.

Calibration Dataset Size N	Accuracy↑/F1-Score↑								
	IJP	GCG	SAA	AutoDAN	PAIR	DrAttack	Puzzler	Zulu	Base64
10	0.90/0.90	0.91/0.90	0.99/0.99	0.96/0.95	0.55/0.18	0.87/0.85	1.00/1.00	0.99/0.99	0.99/0.99
20	0.88/0.89	0.95/0.95	0.99/0.99	0.97/0.97	0.80/0.84	0.87/0.85	1.00/1.00	0.99/0.99	0.99/0.99
30	0.84/0.86	0.97/0.97	0.99/0.99	0.97/0.97	0.84/0.86	0.82/0.80	1.00/1.00	0.99/0.99	0.99/0.99
40	0.85/0.87	0.96/0.97	0.99/0.99	0.96/0.97	0.81/0.82	0.82/0.80	1.00/1.00	0.99/0.99	0.99/0.99
50	0.81/0.84	0.96/0.96	0.99/0.99	0.96/0.96	0.79/0.80	0.78/0.77	0.99/0.66	0.99/0.99	0.99/0.99

prompts are generated by applying nine different jailbreak attacks on five different LLMs. Among these attacks, in-the-wild jailbreak prompts are directly sourced from the dataset released by Shen *et al.* [40], while the remaining jailbreak prompts are specifically generated to target the harmful samples in our dataset. We use the default settings for all the attacks when generating these jailbreak samples, resulting in a total of 32,600 jailbreak prompts. In all experiments, we randomly select N harmful, benign, and jailbreak prompts from our dataset to form the calibration dataset, with the remaining prompts used as the test set. The calibration dataset is used to calibrate the anchor vectors in JBSHIELD. All subsequent experimental results are obtained on the test set.

5.2 Experimental Setup

Models. In our experiments, we utilized a selection of five open-source LLMs, namely Mistral-7B (Mistral-7B-Instruct-v0.2) [25], Vicuna-7B (vicuna-7b-v1.5), Vicuna-13B (vicuna-13b-v1.5) [13], Llama2-7B (Llama-2-7b-chat-hf) [42] and Llama3-8B (Meta-Llama-3-8B-Instruct) [17] from three different model families. These models encompass various model sizes, training data, and alignment processes, providing a comprehensive insight into the existing range of models.

Attack Methods. We evaluate the performance of JBSHIELD in defending nine different jailbreak attacks on selected LLMs. These attacks fall into five different categories, including the manually-designed IJP [40], optimization-based jailbreaks GCG [64] and SAA [4], template-based attacks AutoDAN [63] and PAIR [12], linguistics-based attacks DrAttack [31] and Puzzler [11], and encoding-based attacks Zulu [54] and Base64 [45].

Baselines. To evaluate the effectiveness of JBSHIELD, we compare it against 10 SOTA methods in the field as baselines. These baselines are grouped into two categories based on their primary objectives: jailbreak detection and jailbreak mitigation. For detection, we compare JBSHIELD with Perspective API (PAPI) [27], PPL [3], Llama Guard (LlamaG) [22], Self-Ex [19], and GradSafe [49]. For mitigation, Self-Re [50], PR [23], ICD [47], SD [51], and DRO [61] are considered. Notably, some of the baselines, such as LlamaG and GradSafe, are primarily designed for toxic content detection and are not specifically tailored to address jailbreak scenarios. SD and DRO require modifications to the model, involving fine-tuning processes, whereas the other methods do not necessitate changes to the protected LLM.

Metrics. We use detection accuracy and F1-Score to evaluate the effectiveness of jailbreak detection methods, while the attack success rate (ASR) is used to assess the performance of the jailbreak mitigation method. Jailbreak detection accuracy reflects the ability of the defenses to identify jailbreak prompts. The F1-Score, which incorporates precision, provides insight into the false positive rate of detection methods—that is, whether benign inputs are mistakenly identified as jailbreak prompts. In experiments of jailbreak mitigation, we manually evaluate whether Zulu and Base64 successfully jailbreak the model. For other attacks, we use SORRY-Bench [48] to determine whether a jailbreak attack has successfully bypassed the defense method and caused the model to comply with the jailbreak input to generate unsafe content. The attack success rate is then calculated to reflect the performance of the defenses.

5.3 Hyperparameter Analysis

We conduct hyperparameter analysis to determine the size N of the calibration dataset used in JBSHIELD. We tested detection accuracy and F1-Score on Mistral-7B for different values of N (10, 20, 30, 40, and 50). The results are shown in Table 3. As observed, our method performs best in detecting GCG, AutoDAN, and PAIR when N is set to 30. For the remaining jailbreaks, JBSHIELD-D efficiently detects these attacks with N set to just 10. Notably, for IJP and DrAttack, increasing the number of calibration samples leads to overfitting. Based on the trade-off between detection effectiveness and data efficiency, we set N to 30 for all experiments.

5.4 Jailbreak Detection

In this experiment, we use a calibration dataset comprising 30 benign, 30 harmful, and 30 corresponding jailbreak prompts, totaling 90 samples, to obtain the anchor vectors for each jailbreak. We consistently select an equal number of test benign prompts and test jailbreak prompts to compute jailbreak detection accuracy and F1-Score. This ensures that detection methods perform well in identifying jailbreak prompts and

Table 4: Performance of different jailbreak detection methods.

Methods	Accuracy↑ / F1-Score↑								
	IJP	GCG	SAA	AutoDAN	PAIR	DrAttack	Puzzler	Zulu	Base64
Mistral-7B									
PAPI	0.04/0.08	0.05/0.09	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00
PPL	0.01/0.03	0.33/0.48	0.00/0.00	0.00/0.00	0.01/0.01	0.00/0.00	0.00/0.00	0.95/0.95	0.00/0.00
LlamaG	0.68/0.81	0.78/0.87	0.83/0.90	0.77/0.87	0.74/0.85	0.84/0.91	0.77/0.87	0.50/0.67	0.58/0.73
Self-Ex	0.42/0.59	0.52/0.68	0.40/0.57	0.56/0.72	0.46/0.63	0.51/0.67	0.44/0.62	0.32/0.49	0.37/0.54
GradSafe	0.01/0.02	0.63/0.77	0.00/0.00	0.00/0.00	0.05/0.10	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00
Ours	0.84/0.86	0.97/0.97	0.99/0.99	0.97/0.97	0.84/0.86	0.82/0.80	1.00/1.00	0.99/0.99	0.99/0.99
Vicuna-7B									
PAPI	0.04/0.08	0.14/0.25	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00
PPL	0.01/0.03	0.47/0.62	0.00/0.00	0.01/0.02	0.00/0.00	0.00/0.00	0.00/0.00	0.95/0.95	0.00/0.00
LlamaG	0.65/0.79	0.75/0.86	0.85/0.91	0.72/0.83	0.75/0.85	0.84/0.91	0.75/0.86	0.49/0.65	0.55/0.71
Self-Ex	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.01/0.02	0.01/0.03
GradSafe	0.03/0.06	0.00/0.00	0.00/0.00	0.00/0.00	0.03/0.06	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00
Ours	0.82/0.83	0.95/0.96	0.99/0.99	0.97/0.97	0.91/0.91	0.99/0.99	1.00/0.91	0.99/0.99	1.00/1.00
Vicuna-13B									
PAPI	0.04/0.08	0.02/0.04	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00
PPL	0.01/0.03	0.79/0.86	0.00/0.00	0.01/0.02	0.01/0.02	0.00/0.00	0.00/0.00	0.95/0.95	0.00/0.00
LlamaG	0.64/0.77	0.76/0.86	0.84/0.91	0.75/0.76	0.76/0.86	0.85/0.92	0.75/0.85	0.48/0.64	0.54/0.70
Self-Ex	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00
GradSafe	0.01/0.02	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00
Ours	0.99/0.98	0.99/0.99	0.99/0.99	0.99/0.99	0.98/0.99	0.95/0.98	1.00/0.75	0.99/0.99	1.00/1.00
Llama2-7B									
PAPI	0.04/0.08	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00
PPL	0.01/0.03	0.79/0.86	0.00/0.00	0.10/0.18	0.00/0.00	0.00/0.00	0.00/0.00	0.95/0.95	0.00/0.00
LlamaG	0.41/0.57	0.32/0.48	0.63/0.77	0.38/0.55	0.53/0.69	0.57/0.72	0.49/0.65	0.30/0.46	0.35/0.51
Self-Ex	0.31/0.33	0.28/0.32	0.36/0.39	0.27/0.31	0.27/0.30	0.32/0.35	0.24/0.27	0.30/0.33	0.29/0.32
GradSafe	0.39/0.56	0.97/0.98	0.00/0.00	0.96/0.98	0.62/0.77	0.00/0.00	0.18/0.31	0.00/0.00	0.00/0.00
Ours	0.84/0.86	0.82/0.86	0.93/0.94	0.98/0.98	0.87/0.88	0.99/0.99	0.81/0.85	0.91/0.91	0.92/0.93
Llama3-8B									
PAPI	0.04/0.08	0.02/0.04	0.00/0.00	0.02/0.04	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00
PPL	0.01/0.03	0.85/0.90	0.00/0.00	0.23/0.36	0.00/0.00	0.00/0.00	0.00/0.00	0.95/0.95	0.00/0.00
LlamaG	0.46/0.63	0.54/0.70	0.71/0.83	0.50/0.67	0.60/0.75	0.70/0.82	0.55/0.71	0.34/0.51	0.38/0.56
Self-Ex	0.15/0.26	0.12/0.21	0.19/0.31	0.11/0.19	0.16/0.26	0.16/0.27	0.18/0.30	0.12/0.21	0.14/0.24
GradSafe	0.41/0.58	0.21/0.35	0.00/0.00	0.97/0.98	0.37/0.54	0.00/0.00	0.92/0.96	0.00/0.00	0.00/0.00
Ours	0.91/0.92	0.98/0.99	1.00/1.00	0.97/0.97	0.77/0.86	0.97/0.96	0.99/0.99	0.99/0.99	0.97/0.97

the false positive rate for benign samples is demonstrated.

Detection Performance. We compared the jailbreak detection performance of our JBSHIELD-D on five LLMs against nine different jailbreak attacks, as shown in Table 4. It can be observed that our method achieves superior detection accuracy and F1 scores, significantly outperforming existing methods. For nine jailbreaks across five LLMs, JBSHIELD-D achieves an average detection accuracy of 0.95 and an average F1-Score of 0.94. Among all the baselines, the PAPI almost fails to detect jailbreak prompts, and PPL is only effective against GCG, which has a high proportion of soft prompts. Due to the weaker contextual learning abilities of some LLMs, they may not understand the prompts used by Self-Ex, rendering this baseline almost ineffective on the Vicuna series LLMs. GradSafe performs relatively well only on the Llama series models. For example, it achieves an F1 score of 0.98 for GCG on Llama2-7B, but it is completely ineffective against SAA, DrAttack, Zulu, and Base64. LlamaG demonstrates the best overall performance among the baselines and even outperforms our method when facing DrAttack on Mistral-7B. However, LlamaG requires a large amount of data to fine-tune a new LLM, and it does not maintain such high efficiency across all models or against all attacks. In all cases, LlamaG achieves an accuracy/F1-Score of 0.62/0.75, which is

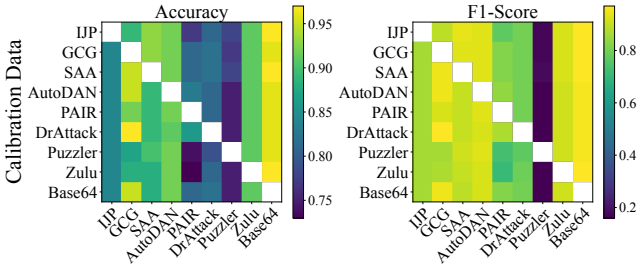

Figure 3: Transferability of JBSHIELD-D.

38%/21% lower than our method. These results demonstrate the superior effectiveness of our method in detecting various jailbreaks across different LLMs.

Transferability. In order to investigate the transferability of JBSHIELD, we used jailbreak prompts from different attacks in the calibration dataset and the test set to evaluate the performance of JBSHIELD-D against unknown jailbreak attacks. In order to investigate the transferability of JBSHIELD, we use jailbreak prompts from different attacks in the calibration dataset and the test set to evaluate the performance of JBSHIELD-D against unknown jailbreak attacks. The transferability results on Mistral-7B are shown in Figure 3. In most cases, our method achieves an accuracy above 0.84 and an F1 score above 0.86. Notably, JBSHIELD-D achieves an accuracy and F1 score above 0.90 when detecting AutoDAN, Zulu, and Base64 samples, regardless of which jailbreak prompts were used for calibration. However, we also observe that JBSHIELD-D exhibited weaker transferability for Puzzler. While the accuracy remained around 0.75, the F1 score dropped to below 0.2. This could be due to the significant difference in the activation strength of its toxic concept compared to other jailbreaks, resulting in a higher false positive rate. Overall, our method demonstrates significant transferability across different jailbreak attacks. This indicates that our method possesses notable robustness even when facing unknown and different types of jailbreak attacks.

Evaluation on Non-model-specific Jailbreak Prompts. To evaluate the model-agnostic effectiveness of JBSHIELD-D, we conducted an experiment using 100 in-the-wild jailbreak prompts that successfully bypassed all five LLMs (as determined by SORRY-Bench). Among these, 30 prompts were randomly selected for calibration, while the remaining 70 were used for testing across the five LLMs. The results, presented in Table 5, demonstrate that JBSHIELD-D achieves robust detection performance even in a non-model-specific setting, maintaining high detection accuracy across all tested models. This validates the versatility and generalizability of our approach under practical scenarios.

Prompts with Only Jailbreak Concept. To further evaluate JBSHIELD-D, we conducted an experiment using 850 jailbreak prompts generated by AutoDAN, where the malicious content was replaced with benign content to simulate cases

Table 5: Performance on non-model-specific jailbreaks.

Models	Accuracy↑	F1-Score↑
Mistral-7B	0.88	0.88
Vicuna-7B	0.87	0.87
Vicuna-13B	0.79	0.78
Llama2-7B	0.84	0.86
Llama3-8B	0.86	0.87

Table 6: Performance on prompts with only jailbreak concept.

Models	Toxic Detected↓	Jailbreak Detected↑	Accuracy↑	F1-Score↑
Mistral-7B	692	158	0.19	0.31
Vicuna-7B	79	771	0.91	0.95
Vicuna-13B	686	164	0.19	0.32
Llama2-7B	23	827	0.97	0.99
Llama3-8B	57	793	0.94	0.97

that activate the jailbreak concept without triggering toxic activation. These modified prompts were tested across five LLMs, and the results are summarized in Table 6. Our findings indicate that JBSHIELD-D performs exceptionally well on Llama and Vicuna-7B, accurately identifying such inputs as non-jailbreak. However, its performance slightly declined on Mistral-7B and Vicuna-13B. This indicates a potential limitation of our approach in handling nuanced cases where jailbreak activation subtly interacts with the model's semantic interpretations. Since our primary focus is on robust jailbreak defense, optimizing performance for these complex scenarios remains an avenue for future work.

5.5 Jailbreak Mitigation

We evaluate the performance of our method by comparing the reduction in ASR of JBSHIELD-M against five jailbreak mitigation baselines across nine selected jailbreak attacks. Among these attacks, IJP, Puzzler, Zulu, and Base64 are transfer-based attacks that do not directly exploit the information of the target LLM. For these jailbreaks, we randomly select 50 corresponding jailbreak prompts from our dataset to test and determine the ASR for each attack. For the other jailbreak methods, we treat the defended model as a new target LLM, generate 50 new jailbreak prompts, and calculate the ASR.

Mitigation Efficiency. The ASRs of nine jailbreak attacks on LLMs deployed with JBSHIELD-M and five baselines are shown in Table 7. Our method reduces the ASR of most jailbreak attacks to zero, significantly outperforming the baselines. Across all five LLMs, JBSHIELD-M lowers the average ASR from 61% to 2%. Notably, our method renders the ASR of AutoDAN, Puzzler, and Base64 attacks 0.00, effectively defending them. Among all the baselines, SD performs best on the Vicuna family models, while ICD shows the best performance on the Llama family models. This can be at-

Table 7: Performance of different jailbreak mitigation methods. No-Def means no defense is deployed.

Models	Methods	Attack Success Rate↓									Average ASR↓
		IJP	GCG	SAA	AutoDAN	PAIR	DrAttack	Puzzler	Zulu	Base64	
Mistral-7B	No-def	0.56	0.92	0.98	1.00	0.82	0.74	1.00	0.48	0.40	0.77
	Self-Re	0.46	0.80	0.86	1.00	0.55	0.40	1.00	0.40	0.18	0.63
	PR	0.40	1.00	0.80	1.00	0.80	0.08	0.90	0.48	0.20	0.63
	ICD	0.52	0.45	0.58	1.00	0.70	0.68	1.00	0.06	0.08	0.56
	SD	0.52	0.70	0.96	0.98	0.78	0.86	1.00	0.32	0.40	0.72
	DRO	0.50	0.88	0.96	1.00	0.40	0.46	1.00	0.48	0.42	0.68
	Ours	0.24	0.36	0.12	0.00	0.08	0.04	0.00	0.02	0.00	0.10
Vicuna-7B	No-def	0.38	0.86	0.96	0.96	0.88	0.94	0.95	0.12	0.18	0.69
	Self-Re	0.34	1.00	0.88	1.00	0.70	0.62	0.95	0.18	0.00	0.63
	PR	0.22	1.00	0.82	1.00	0.75	0.34	0.80	0.40	0.22	0.62
	ICD	0.26	0.80	0.68	1.00	0.65	0.70	0.85	0.00	0.02	0.55
	SD	0.08	0.00	0.04	0.08	0.22	0.12	0.35	0.00	0.00	0.10
	DRO	0.36	1.00	0.64	1.00	0.60	0.52	0.95	0.54	0.06	0.63
	Ours	0.04	0.18	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.03
Vicuna-13B	No-def	0.36	0.78	0.92	1.00	0.68	0.98	0.95	0.0	0.10	0.64
	Self-Re	0.28	1.00	0.76	1.00	0.50	0.30	0.95	0.02	0.02	0.54
	PR	0.32	1.00	0.48	1.00	0.55	0.32	0.95	0.26	0.12	0.56
	ICD	0.28	0.75	0.52	1.00	0.70	0.78	0.45	0.00	0.02	0.50
	SD	0.04	0.02	0.02	0.02	0.08	0.00	0.00	0.00	0.00	0.02
	DRO	0.28	1.00	0.60	1.00	0.40	0.60	0.95	0.14	0.04	0.56
	Ours	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00
Llama2-7B	No-def	0.26	0.50	0.60	0.60	0.30	0.32	0.95	0.14	0.30	0.44
	Self-Re	0.10	0.30	0.48	0.55	0.20	0.22	0.00	0.00	0.00	0.21
	PR	0.20	0.30	0.32	0.40	0.20	0.06	0.15	0.82	0.02	0.27
	ICD	0.02	0.25	0.36	0.70	0.05	0.12	0.00	0.00	0.00	0.17
	SD	0.32	0.00	0.00	0.00	0.24	0.10	0.40	0.00	0.42	0.16
	DRO	0.20	0.10	0.28	0.90	0.30	0.48	0.55	0.02	0.04	0.32
	Ours	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Llama3-8B	No-def	0.24	0.64	0.74	0.62	0.30	0.38	0.45	0.52	0.48	0.49
	Self-Re	0.02	0.15	0.44	0.30	0.05	0.36	0.00	0.02	0.00	0.15
	PR	0.26	0.10	0.14	0.10	0.20	0.04	0.05	0.46	0.06	0.16
	ICD	0.00	0.10	0.18	0.30	0.05	0.00	0.00	0.00	0.00	0.07
	SD	0.42	0.34	0.28	0.26	0.44	0.40	0.95	0.50	0.50	0.45
	DRO	0.24	0.20	0.42	0.50	0.10	0.12	0.00	0.60	0.14	0.26
	Ours	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00

tributed to the differences in decoding strategies between the Vicuna series and the Llama and Mistral series, as well as the Llama family LLMs having superior in-context learning capabilities. Additionally, our method is effective against all types of jailbreaks, while some baselines may exacerbate certain attacks. For example, PR increases the ASR of Zulu on Mistral-7B, Vicuna-13B, and Llama2-7B because it translates low-resource language text into English with lower toxicity, inadvertently raising the ASR. These results demonstrate the efficiency and generalizability of JBSHIELD-M in mitigating various jailbreak attacks across different LLMs.

Utility. To evaluate the performance of models deployed with JBSHIELD-M on regular tasks, we used the 5-shot MMLU benchmark [20] to assess the impact of our methods on LLM usability. The results for JBSHIELD-M, along with all baselines, are shown in Figure 4. Our jailbreak mitigation method impacts the understanding and reasoning capabilities of LLMs by less than 2%, significantly outperforming the baselines. JBSHIELD-M is activated only when a jailbreak prompt is detected, which limits its effect on normal inputs. Among the baselines, PR achieved the lowest MMLU score because it rewrites the stems of test prompts, making it difficult for LLMs to produce the required outputs in multiple-choice questions.

Ablation Study. The two core steps of JBSHIELD-M are the manipulation of the toxic and jailbreak concepts. To verify that both steps are necessary, we conducted ablation studies. We tested the impact of removing the toxic concept enhancement (wo/TCE) and the jailbreak concept weakening (wo/JCW) on JBSHIELD-M across the five selected target

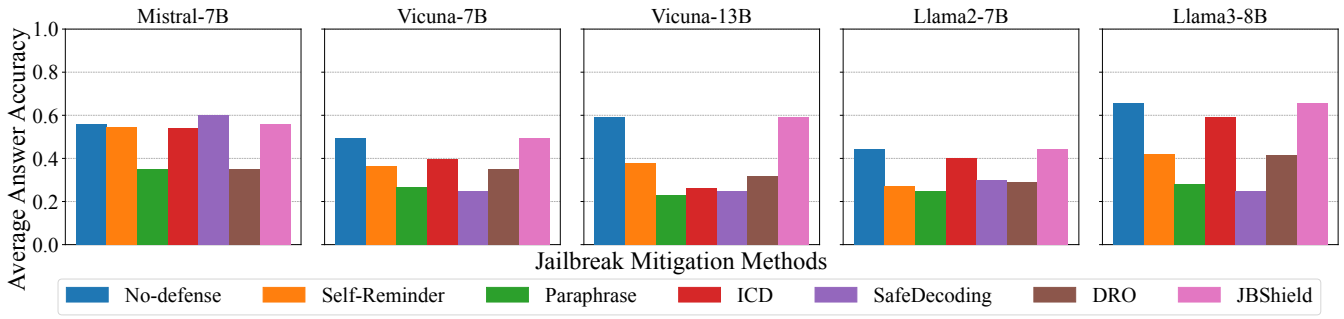

Figure 4: Performance on the MMLU benchmark.

Table 8: Ablation study.

Models	Methods	Attack Success Rate↓								
		IJP	GCG	SAA	AutoDAN	PAIR	DrAttack	Puzzler	Zulu	Base64
Mistral-7B	wo/TCE	0.38	0.20	0.52	0.68	0.22	0.40	1.00	0.10	0.00
	wo/JCW	0.32	0.20	0.06	0.56	0.14	0.36	1.00	0.06	0.00
Vicuna-7B	wo/TCE	0.16	0.04	0.00	0.14	0.42	0.02	0.00	0.06	0.00
	wo/JCW	0.16	0.00	0.18	0.34	0.24	0.00	0.20	0.02	0.00
Vicuna-13B	wo/TCE	0.02	0.00	0.00	0.00	0.20	0.00	0.00	0.00	0.00
	wo/JCW	0.12	0.02	0.58	0.12	0.14	0.06	0.45	0.00	0.00
Llama2-7B	wo/TCE	0.12	0.00	0.00	0.00	0.22	0.08	0.00	0.00	0.00
	wo/JCW	0.04	0.02	0.00	0.02	0.08	0.12	0.00	0.08	0.00
Llama3-8B	wo/TCE	0.10	0.00	0.02	0.02	0.20	0.02	0.00	0.12	0.04
	wo/JCW	0.02	0.00	0.06	0.04	0.08	0.02	0.00	0.02	0.00

Table 9: Performance of JBSHIELD-D against adaptive attacks.

Models	Attack Success Rate↓		
	AutoDAN-based	GCG-based	LLMFuzzer-based
Mistral-7B	0.00	0.14	0.02
Vicuna-7B	0.18	0.00	0.00
Vicuna-13B	0.00	0.02	0.00
Llama2-7B	0.00	0.04	0.00
Llama3-8B	0.00	0.00	0.00

models. The results are shown in Table 8. As demonstrated, removing either of the two key steps leads to a decline in performance. After removing the manipulation of the toxic and jailbreak concepts, the overall average ASR increased to 12% and 13%, respectively. Interestingly, we found that different models appear to have varying sensitivities to different concepts. For example, on Vicuna-13B, omitting the weakening of the jailbreak concept significantly increases the attack success rate, while on Mistral-7B, the opposite effect is observed.

Performance against Adaptive Attacks. To evaluate the robustness of JBSHIELD, we tested it against three types of adaptive attacks: AutoDAN-based, GCG-based, and LLMFuzzer-based. Each attack was designed to bypass our mitigation strategy and incorporate weakening the toxic concept and enhancing the jailbreak concept into the attack's objective function. For each LLM, 50 jailbreak prompts were generated for evaluation. The results, as shown in Table 9, demonstrate that JBSHIELD maintains exceptional robustness across all attack types and models. Specifically, the average attack success rates for AutoDAN-based, GCG-based, and LLMFuzzer-based attacks are 0.4%, 4.0%, and 0.4%, respectively. These results confirm that JBSHIELD effectively mitigates adaptive jailbreak attempts, showcasing its resilience in real-world scenarios.

6 Discussions

6.1 Practicality and Scalability

As illustrated in Table 10, unlike existing solutions that typically focus on either detection or mitigation, our JBSHIELD integrates both functionalities, effectively addressing these two aspects of jailbreak defense. In terms of resource utilization and operational overhead, JBSHIELD stands out by eliminating extra tokens, model fine-tuning, and reducing reliance on extensive additional training data. These properties make our approach easily deployable on existing LLMs. Notably, JBSHIELD requires only about 30 jailbreak prompts for

Table 10: Summary of existing jailbreak defenses. ● indicates that the method utilizes the corresponding resource or requires the specified operation. Conversely, ○ denotes that the method does not require the listed resource or the operation. In the additional tokens consumed during the inference stage, m represents the number of tokens in the original user input.

Categories	Defenses	Extra Tokens in Inference	Extra Model for Defense	Target LLM Fine-tuning	Extra Data (prompts)	User Input Modified
Detection	PPL [3]	○	GPT-2	○	∼500	○
	Gradient cuff [21]	∼20m	○	○	∼100	●
	Self-Ex [19]	∼40	○	○	○	○
	SmoothLLM [39]	∼5m	○	○	○	●
	GradSafe [49]	○	○	○	∼4	○
	LlamaG [22]	○	Llama Guard	○	13,997	○
Mitigation	Self-Re [50]	∼40	○	○	○	●
	PR [23]	∼20+m	GPT-3.5	○	○	●
	ICD [47]	∼50	○	○	∼1	●
	SD [51]	∼m	LoRA Model	●	∼70	○
	LED [59]	○	○	●	∼700	○
	DRO [61]	∼120	○	○	∼200	●
Comprehensive Defense	JBSHIELD	○	○	○	∼90	○

calibration to effectively defend against each type of jailbreak attack. This minimal cost enables JBSHIELD to achieve better scalability compared to previous methods, making it easier to adapt to future emerging attacks.

6.2 Limitations

Model Dependency. Our detection and mitigation strategies rely on access to the internal architecture and parameters of LLMs, as well as the ability to probe and modify hidden representations during the forward pass. Although we have validated the effectiveness of JBSHIELD across multiple existing LLMs, its effectiveness on future, potentially novel LLM architectures remains uncertain. However, since neural network models inherently process and understand data through hidden representations, we believe that even with the emergence of new LLM architectures, our method will still be capable of addressing jailbreak attacks by analyzing these representations to extract the relevant concepts.

Data Sensitivity. The performance of our approach relies on the quality and diversity of the calibration dataset, which serves as the foundation for detecting and mitigating jailbreak prompts. A less diverse calibration dataset may limit the method's generalizability to novel or significantly different jailbreak attempts. However, our experiments (Section 5.3) demonstrate that JBShield exhibits strong transferability across unseen jailbreaks, leveraging shared similarities in jailbreak concepts. Furthermore, JBShield requires minimal calibration samples (only 30) to achieve high performance. By augmenting the calibration dataset with additional diverse samples, JBShield can effectively adapt to emerging jailbreak attacks, ensuring its robustness in evolving scenarios.

7 Conclusion and Future Works

In this work, we conducted an in-depth exploration of how jailbreaks influence the output of LLMs. We revealed that LLMs can indeed recognize the toxic concept within jailbreak prompts, and the primary reason these prompts alter model behavior is the introduction of the jailbreak concept. Building on these findings, we proposed a comprehensive jailbreak defense framework, JBSHIELD, comprising both detection and mitigation components. The detection method, JBSHIELD-D, identifies jailbreak prompts by analyzing and detecting the activation of the toxic and jailbreak concepts. The mitigation method, JBSHIELD-M, safeguards LLMs from the influence of jailbreak inputs by enhancing the toxic concept while weakening the jailbreak concept. Extensive experiments demonstrated that JBSHIELD effectively defends against various state-of-the-art (SOTA) jailbreaks across multiple LLMs.

Building on our findings, we identify two promising directions for future work. First, it is essential to further investigate the mechanisms underlying jailbreak attacks on LLMs. Future work should aim to uncover more nuanced aspects of how these attacks manipulate model behavior, particularly under new LLM architectures. Such investigations could lead to the development of more advanced detection algorithms that are better equipped to adapt to changes in adversarial strategies and model updates. Additionally, our current method utilizes calibration data to determine a fixed value for the scaling factor, which remains constant throughout the process but lacks flexibility. As new tokens are generated, the overall semantics of the input prompt keep changing, leading to variations in concept activation. Designing an adaptive control method for the scaling factor would further improve the performance of concept manipulation-based defenses.

Acknowledgments

We thank the anonymous reviewers and our shepherd for their helpful and valuable feedback. This work was partially supported by the NSFC under Grants U2441240 ("Ye Qisun" Science Foundation), 62441238, U21B2018, U24B20185, T2442014, 62161160337, and 62132011, the National Key R&D Program of China under Grant 2023YFB3107400, the Research Grants Council of Hong Kong under Grants R6021-20F, R1012-21, RFS2122-1S04, C2004-21G, C1029-22G, C6015-23G, and N_CityU139/21, the Shaanxi Province Key Industry Innovation Program under Grants 2023-ZDLGY-38 and 2021ZDLGY01-02.

Ethics Considerations

Our jailbreak defense framework JBSHIELD serves as a safeguard to prevent the exploitation of LLMs for generating inappropriate or unsafe content. By improving the detection and mitigation of jailbreak attacks, we contribute to a safer deployment of LLMs, ensuring that their outputs align with ethical standards and societal norms. Our study does not require Institutional Review Board (IRB) approval as it involves the use of publicly available data and methods without direct human or animal subjects. All experimental protocols are designed to adhere to ethical standards concerning artificial intelligence research, focusing on improving technology safety without infringing on personal privacy or well-being. Our research activities strictly comply with legal and ethical guidelines applicable to computational modeling and do not engage with sensitive or personally identifiable information. Addressing the exposure to harmful content during the development and calibration of JBSHIELD, we ensure that all team members have access to support and resources to manage potential distress. Ethical guidelines are strictly followed to minimize direct exposure and provide psychological safety measures. While our framework has demonstrated robustness against current jailbreak strategies, the dynamic nature of threats necessitates ongoing development. We propose the design of dynamic strategies for key parameters like detection thresholds and scaling factors to effectively counteract new and evolving jailbreak strategies.

Open Science

In compliance with the Open Science policy, we will share all necessary artifacts with the research community and ensure that they are accessible for review by the artifact evaluation committee to enhance the reproducibility of our work. Specifically, we will provide our test datasets, the code for extracting concept-related interpretable tokens, and the implementation of JBShield-D and JBShield-M for testing across five target LLMs.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Henk Alkemade, Steven Claeyssens, Giovanni Colavizza, Nuno Freire, Jörg Lehmann, Clemens Neudeker, Giulia Osti, Daniel van Strien, et al. Datasheets for digital cultural heritage datasets. *Journal of open humanities data*, 9(17):1–11, 2023.

[3] Gabriel Alon and Michael Kamfonas. Detecting language model attacks with perplexity. *arXiv preprint arXiv:2308.14132*, 2023.

[4] Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned llms with simple adaptive attacks. *arXiv preprint arXiv:2404.02151*, 2024.

[5] Anthropic. Introducing claude. https://www.anthropic.com/news/introducing-claude, 2023.

[6] Usman Anwar, Abulhair Saparov, Javier Rando, Daniel Paleka, Miles Turpin, Peter Hase, Ekdeep Singh Lubana, Erik Jenner, Stephen Casper, Oliver Sourbut, et al. Foundational challenges in assuring alignment and safety of large language models. *arXiv preprint arXiv:2404.09932*, 2024.

[7] Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Rimsky, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. *arXiv preprint arXiv:2406.11717*, 2024.

[8] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[9] Bochuan Cao, Yuanpu Cao, Lu Lin, and Jinghui Chen. Defending against alignment-breaking attacks via robustly aligned llm. *arXiv preprint arXiv:2309.14348*, 2023.

[10] Nicholas Carlini, Milad Nasr, Christopher A. Choquette-Choo, Matthew Jagielski, Irena Gao, Pang Wei Koh, Daphne Ippolito, Florian Tramèr, and Ludwig Schmidt. Are aligned neural networks adversarially aligned? In *Proc. of NeurIPS*, volume 36, pages 61478–61500, 2023.

[11] Zhiyuan Chang, Mingyang Li, Yi Liu, Junjie Wang, Qing Wang, and Yang Liu. Play guessing game with llm: Indirect jailbreak attack with implicit clues. *arXiv preprint arXiv:2402.09091*, 2024.

[12] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*, 2023.

[13] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. https://lmsys.org/blog/2023-03-30-vicuna/, 2023.

[14] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Proc. of NeurIPS*, 30, 2017.

[15] Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. Or-bench: An over-refusal benchmark for large language models. *arXiv preprint arXiv:2405.20947*, 2024.

[16] Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. Jailbreaker: Automated jailbreak across multiple large language model chatbots. *arXiv preprint arXiv:2307.08715*, 2023.

[17] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[18] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[19] Alec Helbling, Mansi Phute, Matthew Hull, and Duen Horng Chau. Llm self defense: By self examination, llms know they are being tricked. *arXiv preprint arXiv:2308.07308*, 2023.

[20] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *Proc. of ICLR*, 2021.

[21] Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. Gradient cuff: Detecting jailbreak attacks on large language models by exploring refusal loss landscapes. *arXiv preprint arXiv:2403.00867*, 2024.

[22] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

[23] Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*, 2023.

[24] Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Proc. of NeurIPS*, 36, 2024.

[25] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[26] Shuyu Jiang, Xingshu Chen, and Rui Tang. Prompt packer: Deceiving llms through compositional instruction with hidden attacks. *arXiv preprint arXiv:2310.10077*, 2023.

[27] Jigsaw. Perspective api. https://www.anthropic.com/news/introducing-claude, 2021.

[28] Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. Openassistant conversations-democratizing large language model alignment. *Proc. of NeurIPS*, 36, 2024.

[29] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

[30] Chak Tou Leong, Yi Cheng, Kaishuai Xu, Jian Wang, Hanlin Wang, and Wenjie Li. No two devils alike: Unveiling distinct mechanisms of fine-tuning attacks. *arXiv preprint arXiv:2405.16229*, 2024.

[31] Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. Drattack: Prompt decomposition and reconstruction makes powerful llm jailbreakers. *arXiv preprint arXiv:2402.16914*, 2024.

[32] Yuping Lin, Pengfei He, Han Xu, Yue Xing, Makoto Yamada, Hui Liu, and Jiliang Tang. Towards understanding jailbreak attacks in llms: A representation space analysis. *arXiv preprint arXiv:2406.10794*, 2024.

[33] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box llms automatically. *arXiv preprint arXiv:2312.02119*, 2023.

[34] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proc. of NAACL-HLT*, 2013.

[35] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. In *Proc. of BlackboxNLP*, 2023.

[36] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Proc. of NeurIPS*, 35:27730–27744, 2022.

[37] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Proc. of ICML*, 2024.

[38] Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *Proc. of ICLR*, 2024.

[39] Alexander Robey, Eric Wong, Hamed Hassani, and George J Pappas. Smoothllm: Defending large language models against jailbreaking attacks. *arXiv preprint arXiv:2310.03684*, 2023.

[40] Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. *arXiv preprint arXiv:2308.03825*, 2023.

[41] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

[42] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[43] Mengru Wang, Ningyu Zhang, Ziwen Xu, Zekun Xi, Shumin Deng, Yunzhi Yao, Qishen Zhang, Linyi Yang, Jindong Wang, and Huajun Chen. Detoxifying large language models via knowledge editing. *arXiv preprint arXiv:2403.14472*, 2024.

[44] Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966*, 2023.

[45] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? In *Proc. of NeurIPS*, volume 36, pages 80079–80110, 2023.

[46] Boyi Wei, Kaixuan Huang, Yangsibo Huang, Tinghao Xie, Xiangyu Qi, Mengzhou Xia, Prateek Mittal, Mengdi Wang, and Peter Henderson. Assessing the brittleness of safety alignment via pruning and low-rank modifications. *arXiv preprint arXiv:2402.05162*, 2024.

[47] Zeming Wei, Yifei Wang, and Yisen Wang. Jailbreak and guard aligned language models with only few in-context demonstrations. *arXiv preprint arXiv:2310.06387*, 2023.

[48] Tinghao Xie, Xiangyu Qi, Yi Zeng, Yangsibo Huang, Udari Madhushani Sehwag, Kaixuan Huang, Luxi He, Boyi Wei, Dacheng Li, Ying Sheng, et al. Sorry-bench: Systematically evaluating large language model safety refusal behaviors. *arXiv preprint arXiv:2406.14598*, 2024.

[49] Yueqi Xie, Minghong Fang, Renjie Pi, and Neil Gong. Gradsafe: Detecting unsafe prompts for llms via safety-critical gradient analysis. *arXiv preprint arXiv:2402.13494*, 2024.

[50] Yueqi Xie, Jingwei Yi, Jiawei Shao, Justin Curl, Lingjuan Lyu, Qifeng Chen, Xing Xie, and Fangzhao Wu. Defending chatgpt against jailbreak attack via self-reminders. *Nature Machine Intelligence*, 5(12):1486–1496, 2023.

[51] Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. Safedecoding: Defending against jailbreak attacks via safety-aware decoding. *arXiv preprint arXiv:2402.08983*, 2024.

[52] Zhihao Xu, Ruixuan Huang, Changyu Chen, Shuai Wang, and Xiting Wang. Uncovering safety risks of large language models through concept activation vector. *arXiv preprint arXiv:2404.12038*, 2024.

[53] Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. Jailbreak attacks and defenses against large language models: A survey. *arXiv preprint arXiv:2407.04295*, 2024.

[54] Zheng Xin Yong, Cristina Menghini, and Stephen Bach. Low-resource languages jailbreak GPT-4. In *Proc. of NeurIPS SoLaR Workshop*, 2023.

[55] William J Youden. Index for rating diagnostic tests. *Cancer*, 3(1):32–35, 1950.

[56] Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. {LLM-Fuzzer}: Scaling assessment of large language model jailbreaks. In *Proc. of USENIX Security*, pages 4657–4674, 2024.

[57] Zhiyuan Yu, Xiaogeng Liu, Shunning Liang, Zach Cameron, Chaowei Xiao, and Ning Zhang. Don't listen to me: Understanding and exploring jailbreak prompts of large language models. In *Proc. of USENIX Security*, pages 4675–4692, 2024.

[58] Lifan Yuan, Yichi Zhang, Yangyi Chen, and Wei Wei. Bridge the gap between cv and nlp! a gradient-based textual adversarial attack framework. In *Proc. of ACL*, pages 7132–7146, 2023.

[59] Wei Zhao, Zhe Li, Yige Li, Ye Zhang, and Jun Sun. Defending large language models against jailbreak attacks via layer-specific editing. *arXiv preprint arXiv:2405.18166*, 2024.

[60] Wei Zhao, Zhe Li, and Jun Sun. Causality analysis for evaluating the security of large language models. *arXiv preprint arXiv:2312.07876*, 2023.

[61] Chujie Zheng, Fan Yin, Hao Zhou, Fandong Meng, Jie Zhou, Kai-Wei Chang, Minlie Huang, and Nanyun Peng. On prompt-driven safeguarding for large language models. In *Proc. of ICML*, 2024.

[62] Zhenhong Zhou, Haiyang Yu, Xinghua Zhang, Rongwu Xu, Fei Huang, and Yongbin Li. How alignment and jailbreak work: Explain llm safety through intermediate hidden states. *arXiv preprint arXiv:2406.05644*, 2024.

[63] Sicheng Zhu, Ruiyi Zhang, Bang An, Gang Wu, Joe Barrow, Zichao Wang, Furong Huang, Ani Nenkova, and Tong Sun. Autodan: Automatic and interpretable adversarial attacks on large language models. *arXiv preprint arXiv:2310.15140*, 2023.

[64] Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

A Additional Explanation and Results of Concept Extraction

The overall process of using our Concept Extraction algorithm to get the toxic concept in harmful prompts is shown in Algorithm 1. The extraction process for the other two concepts is similar. It only requires replacing the prompt types forming the counterfactual pairs with the corresponding ones (toxic concept: (harmful, benign) and (jailbreak, benign), jailbreak concept: (jailbreak, harmful)). The results of concept extraction on two Llama family models and two Vicuna family LLMs for all three concepts are presented in Table 11 and 12. As observed, different LLMs have slight variations in their understanding of toxic and jailbreak concepts. For instance, Llama3-8B, similar to Mistral-7B, associates the toxic concept with words like "illegal," while Llama2-7B associates it with words like "Sorry" and "cannot." However, the overall findings align with the statements in Section 3.2: LLMs can recognize similar toxic concepts in both jailbreak and harmful prompts, and the activation of jailbreak concepts in jailbreak prompts is the reason they can change the model output from rejection to compliance.

Algorithm 1 Concept Extraction of the Toxic Concept

Input: N harmful prompts $\{(x_i^h)\}_{i=1}^N$ and N benign prompts $\{(x_i^b)\}_{i=1}^N$, target LLM f, layer index l for extraction, vocabulary $\mathcal{V}$ for f.
Output: Toxic subspace $\mathbf{v}$ at layer l, tokens $\{t_i\}_{i=1}^k$ that interpret the toxic concept.

```
Form counterfactual pairs of prompts {(x_i^h, x_i^b)}_{i=1}^N
Initialize difference matrix D^l
for i ← 1 to N do
    Get embeddings e_h^l and e_b^l at layer l for x_i^h and x_i^b
    Form representation pair (e_b^l, e_h^l)
    Append the pair to matrix D^l
end for
Perform SVD on D^l and get singular vector matrix V
Extract the first column of V as v
Project v onto vocabulary V to get scores
Get top-k tokens {t_i}_{i=1}^k with highest k scores
return v, {t_i}_{i=1}^k
```

B Additional Experiment Results

B.1 Concept-Based Detection vs. Direct Embedding Comparison

To evaluate whether comparing conceptual subspaces is necessary for jailbreak detection, we conducted additional experiments comparing JBShield's concept-based detection approach with a direct embedding similarity comparison. In the latter approach, the detection relied solely on calculating the similarity between the sentence embedding of a new input prompt and the average embeddings of anchor prompts (benign and harmful). The results, summarized in Table 13, demonstrate the superiority of JBShield's concept-based approach. Direct embedding comparisons achieved an average F1-score of only 0.62 across five LLMs and nine jailbreak attacks, significantly lower than JBShield's F1-score of 0.94. This substantial difference highlights that directly comparing

Table 11: Results of concept extraction on layer23 of Vicuna-7B and layer26 Vicuna-13B.

Concepts	Source Prompts	Associated Interpretable Tokens
Vicuna-7B		
Toxic Concepts	Harmful	**Sorry**, **sorry**, azionale, **Note**
	IJP	understood, Hi, Hello, hi
	GCG	**sorry**, **Sorry**, **orry**, Portail
	SAA	explo, Rule, Step, RewriteRule
	AutoDAN	character, lista, character, multicol
	PAIR	**sorry**, **Sorry**, Please, yes
	DrAttack	question, example, Example, Example
	Puzzler	step, setup, steps, re
	Zulu	Ubuntu, ubuntu, mlung, **sorry**
	Base64	step, base, Step, step
Jailbreak Concepts	IJP	**understood**, **understand**, in, hi
	GCG	**sure**, **Sure**, zyma, **start**
	SAA	**sure**, **Sure**, rules, **started**
	AutoDAN	**character**, list, **Character**, **character**
	PAIR	**sure**, **Sure**, of, ure
	DrAttack	example, question, Example, **answer**
	Puzzler	re, step, **establish**, Re
	Zulu	Ubuntu, Johannes, **translated**, African
	Base64	**base**, **Base**, **Base**, **decode**
Vicuna-13B		
Toxic Concepts	Harmful	**NOT**, **neither**, **warning**, please
	IJP	understood, ok, okay, OK
	GCG	**sorry**, **Sorry**, **unfortunately**, sad
	SAA	purely, surely, ‘<, enta
	AutoDAN	list, List, List, lists
	PAIR	**NOT**, **sorry**, **NOT**, **unfortunately**
	DrAttack	answering, answer, **sorry**, question
	Puzzler	step, Step, manipulate, step
	Zulu	South, Johannes, Ubuntu, **sorry**
	Base64	decode, base, Base, BASE
Jailbreak Concepts	IJP	**understood**, **okay**, **welcome**, **Ready**
	GCG	advis, **please**, disc, **doing**
	SAA	**Sure**, **sure**, **readily**, Sitz
	AutoDAN	list, points, List, **Character**
	PAIR	Unterscheidung, **sure**, **Sure**, **initially**
	DrAttack	**answers**, **answer**, question, **answered**
	Puzzler	step, Step, prep, **establish**
	Zulu	Johannes, Ubuntu, South, Cape
	Base64	**Received**, **decode**, **base**, deser

Table 12: Results of concept extraction on layer22 of Llama2-7B and layer32 Llama3-8B.

Concepts	Source Prompts	Associated Interpretable Tokens
Llama2-7B		
Toxic Concepts	Harmful	**Sorry**, **cannot**, **I**, **sorry**
	IJP	Hello, **I**, Language, Gre
	GCG	**Sorry**, **I**, **sorry**, **cannot**
	SAA	onymes, Kontrola, edeut, **limits**
	AutoDAN	**Sorry**, **cannot**, **I**, **sorry**
	PAIR	**Sorry**, **cannot**, **I**, **Cannot**
	DrAttack	I, dex, **cannot**, ich
	Puzzler	ungs, elle, unable, **Cannot**
	Zulu	**sorry**, mlung, forg, **Sorry**
	Base64	Base, Hi, BASE, hi
Jailbreak Concepts	IJP	gre, **wel**, **welcome**, **hi**
	GCG	**press**, in, gon, Krie
	SAA	**press**, sak, in, nal
	AutoDAN	ains, ola, GridView, vre
	PAIR	**yes**, **Yes**, **Yes**, **thanks**
	DrAttack	**thanks**, Jorge, **yes**, dust
	Puzzler	ode, **yes**, rell, prove
	Zulu	**transl**, **translate**, **meaning**, **translated**
	Base64	**hi**, **sure**, **base**, **yes**
Llama3-8B		
Toxic Concepts	Harmful	I, I, **Illegal**, **legality**
	IJP	**Illegal**, **legality**, **unethical**, **illegal**
	GCG	Oh, **no**, **neither**, **cannot**
	SAA	[text, Sure, text, =text
	AutoDAN	I, **Illegal**, **legality**, **unethical**
	PAIR	I, **Illegal**, **legality**, **unethical**
	DrAttack	USER, USER, I, (USER
	Puzzler	Step, Dr, Step, step
	Zulu	Ng, Ing, Uk, Iz
	Base64	base, Dec, Base, decoding
Jailbreak Concepts	IJP	., :, S, C
	GCG	.**Accessible**, S, C, (
	SAA	**Sure**, **Sure**, <, {text
	AutoDAN	**here**, **as**, **Here**, **Here**
	PAIR	**as**, ylvania, when, what
	DrAttack	**Sure**, **Sure**, **sure**, **sure**
	Puzzler	based, **here**, **Here**, after
	Zulu	to, Looks, looks, another
	Base64	siz, podob, **base**, .accounts

embeddings fails to capture nuanced distinctions between benign, harmful, and jailbreak prompts. By leveraging conceptual subspaces, JBShield identifies and interprets critical semantic differences that are overlooked by direct embedding comparison.

B.2 Performance on harmful benchmarks

To demonstrate the scalability of our approach, we retained the detection and enhancement of toxic semantics in JBSHIELD-M and tested the proportion of unsafe responses on two harmful benchmarks, AdvBench [64] and HEx-PHI [38]. The results are shown in Table 14. By controlling toxic concepts, we can effectively prevent LLMs from outputting unsafe content. These results indicate that detecting and strengthening toxic concepts enables all target models to generate safe outputs for harmful inputs, whereas existing defenses do not guarantee effectiveness across all five models. This highlights the potential of our approach for toxicity detection applications.

Table 13: Comparison with a direct embedding similarity comparison.

Models	F1-Score↑								
	IJP	GCG	SAA	AutoDAN	PAIR	DrAttack	Puzzler	Zulu	Base64
Mistral-7B	0.02	0.46	0.57	0.91	0.31	0.84	1.00	0.99	1.00
Vicuna-7B	0.17	0.00	0.57	0.48	0.29	0.99	0.95	0.92	1.00
Vicuna-13B	0.02	0.00	0.57	0.61	0.00	0.72	0.95	0.95	1.00
Llama2-7B	0.68	0.04	0.88	0.81	0.68	0.44	0.94	0.92	1.00
Llama3-8B	0.06	0.00	0.75	0.68	0.21	0.35	0.98	0.97	1.00

Table 14: Performance of jailbreak mitigation methods against harmful inputs.

Models	Methods	Harmful Benchmark↓	
		AdvBench	HEx-PHI
Mistral-7B	No-defense	0.30	0.10
	Self-Re	0.00	0.03
	PR	0.57	0.23
	ICD	0.03	0.00
	SD	0.73	0.37
	DRO	0.00	0.03
	JBShield-M	0.00	0.00
Vicuna-7B	No-defense	0.07	0.00
	Self-Re	0.00	0.00
	PR	0.10	0.03
	ICD	0.00	0.00
	SD	0.00	0.00
	DRO	0.00	0.00
	JBShield-M	0.00	0.00
Vicuna-13B	No-defense	0.00	0.00
	Self-Re	0.00	0.00
	PR	0.03	0.07
	ICD	0.00	0.00
	SD	0.03	0.00
	DRO	0.00	0.00
	JBShield-M	0.00	0.00
Llama2-7B	No-defense	0.00	0.00
	Self-Re	0.00	0.00
	PR	0.00	0.00
	ICD	0.00	0.00
	SD	0.00	0.00
	DRO	0.00	0.00
	JBShield-M	0.00	0.00
Llama3-8B	No-defense	0.03	0.00
	Self-Re	0.00	0.00
	PR	0.07	0.07
	ICD	0.00	0.00
	SD	0.10	0.07
	DRO	0.00	0.00
	JBShield-M	0.00	0.00

Table 15: Performance on normal inputs with seemingly toxic words.

Models	False Positive Rate↓
Mistral-7B	0.06
Vicuna-7B	0.04
Vicuna-13B	0.00
Llama2-7B	0.00
Llama3-8B	0.00

B.3 Evaluation on Normal Inputs with Seemingly Toxic Words

To investigate the impact of JBShield on normal inputs containing seemingly toxic words, we conducted an additional evaluation using the OR-Bench-Hard-1K dataset [15], which comprises prompts designed to appear toxic without harmful intent. The evaluation focused on measuring JBShield's false positive rate across five LLMs. The results, presented in Table 15, demonstrate JBShield's robustness in handling such inputs. The average false positive rate was 2%, indicating that JBShield rarely misclassifies normal inputs containing toxic language as jailbreak prompts. These findings validate JBShield's ability to distinguish between genuinely harmful or jailbreak inputs and benign inputs with superficially toxic semantics. This evaluation further highlights the reliability and precision of JBShield in real-world applications.