NegBLEURT Forest: Leveraging Inconsistencies to Detect Jailbreak attacks

Anonymous ACL submission

Abstract

As Large Language Models (LLMs) continue to advance, ensuring their robustness and security remains a critical challenge. Jailbreak attacks pose a significant risk by coercing LLMs into generating harmful or ethically inappropriate content, even when such models are trained to follow strict guidelines. Furthermore, defining universal filtering policies is inherently context-dependent and difficult to generalize. To address these challenges without relying on additional rule-based filters or predefined thresholds, this paper presents a novel detection framework for assessing whether an LLM-generated response aligns with expected safe behavior. The proposed approach evaluates response consistency from two complementary perspectives: intra-consistency, which analyzes how reference responses in predefined **R**efusal **S**emantic **D**omain (RSD) vary in the latent space; and inter-consistency, which measures the semantic alignment between the LLM response and this RSD, followed by semi-supervised classification using Isolation Forest. This methodology enables effective jailbreak detection without the need for empirically defined thresholds, offering a more scalable and adaptable solution for real-world applications[1].

1 Introduction

Large Language Models (LLMs) are powerful neural networks with large parameter sizes and strong in-context learning capabilities, widely used for tasks like summarization, text completion, and question answering (Hadi et al., 2023; OpenAI, 2023; Kasneci et al., 2023; Zhao et al., 2023). Popular models include GPT-3 (Mann et al., 2020), GPT-4 (Achiam et al., 2023), and LLAMA (Touvron et al., 2023), typically accessed via APIs or web interfaces. However, this accessibility exposes them to cyber threats, such as prompt-based attacks, which can manipulate model behavior and compromise system security (Perez and Ribeiro, 2022). Two major attacks on LLMs are prompt injection and jailbreak attacks, both aiming to bypass safety mechanisms and generate harmful or illegal content (Chao et al., 2023; Zou et al., 2023). Prompt injection involves appending malicious input to a prompt and has been identified by OWASP as a top LLM-related threat (OWASP, 2025; Kumar et al., 2024). In contrast, jailbreak attacks bypass safety filters without prompt concatenation (Yi et al., 2024), posing a significant challenge due to their stealthy nature (Wang et al., 2023), especially in sensitive applications like business (Wu et al., 2023), education (Blodgett and Madaio, 2021), and healthcare (Sallam, 2023). In this paper, we mainly focuses on jailbreak attacks. Existing methods, including SmoothLLM, struggle to reliably detect jailbreak attacks. Reproducing prior approaches like JailGuard is also challenging, often leading to inconsistent results—even on the same datasets—due to implementation difficulties and unclear threshold settings. These issues reflect broader reproducibility and comparability problems in the field.

This work therefore explores two key questions: RQ1: Can jailbreak attacks be detected without relying on additional filters, fine-tuning, or threshold tuning? RQ2: Can we build an efficient detection method that generalizes well across different datasets, models, and applications? To answer these questions, we conducted extensive experiments to study these dependencies by testing jailbreak attacks on two LLMs—the LLama-7-2B and the Gemma-2-9B—and proposed a novel framework, NegBLEURT Forest. In this work, Section 2 provides an overview of the existing jailbreak attacks and defense mechanisms. Section 3 introduces the first study for the intra-consistency between the responses in perturbed prompts. In Section 4, the second semi-unsupervised approach is presented which leverages K-means clustering and

[1] `https://anonymous.4open.science/r/jailbreaktester-247D`

the Isolation Forest machine learning algorithm. The corresponding experimental evaluation and discussion are also elaborated. Finally, Section 5 concludes the paper and outlines the limitations of the proposed work.

2 Related Work

Many research works have been proposed to detect jailbreak attacks. For example, SmoothLLM (Robey et al., 2023), JailGuard (Zhang et al., 2023), LlamaGuard (Inan et al., 2023), defense strategies in (Metzen et al., 2017; Liu et al., 2022; Dong et al., 2021) identify attacks and help secure AI systems. However, most of the proposed methods are based on comparing inputs and/or outputs against a reference dataset and use thresholds that are determined empirically. As a result, these solutions often lack generalization across different datasets and models, making their results difficult to reproduce. Furthermore, many alternative approaches involve fine-tuning the models used, which requires frequent updates, an approach that is resource intensive and expensive (Rahman et al., 2024). Safety training methods for LLMs, such as GPT-4 and Claude, frequently fine-tune pre-trained models using human preferences (Christiano et al., 2017; Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022; Bai et al., 2022a) and AI feedback (Bai et al., 2022b; Achiam et al., 2023; Sun et al., 2023). Another detection method is the input perplexity score which uses an additional model to detect jailbreak prompts(Alon and Kamfonas, 2023). Jailbreak attacks aim to trick large language models (LLMs) into producing harmful or unethical responses. Starting from early work on hand-crafted prompts (Walkerspider, 2022), jailbreaks can be divided into two types: conflicting goals and generalization mismatches (Wang et al., 2018). Conflicting goals attacks use carefully designed inputs to make the model choose between safe behavior and harmful prompts, with well-known examples like GCG and AutoDAN. The Greedy Coordinate Gradient (GCG) method (Zou et al., 2023) generates adversarial suffixes added to harmful prompts and works effectively on black-box commercial LLMs. AutoDAN (Zou et al., 2023) uses genetic algorithms with mutation and crossover to create natural adversarial prefixes. Generalization mismatches exploit the gap between broad pretraining data and narrower safety fine-tuning data. For instance, Yong et al. (Yong et al., 2023) jailbreak GPT-4 by turning user prompts into low-resource languages. Another approach uses sequence-to-sequence models, like LLMs (Tian et al., 2023) or multi-agent systems (Chao et al., 2023), to transform harmful prompts into jailbreak prompts. Defensive methods fall into two main types: pre-processing and post-processing. Pre-processing changes harmful prompts using techniques like smoothing or detection to remove adversarial parts (Ji et al., 2024; Robey et al., 2023; Cao et al., 2023; Hu et al., 2024; Zhang et al., 2025; Lin et al., 2024). Post-processing uses filters to check if the model's responses are safe (Pisano et al., 2023; Phute et al., 2023; Zeng et al., 2024; Xiong et al., 2024). Both methods work well but have drawbacks. Pre-processing relies on thresholds to separate safe and harmful prompts, but these thresholds are often chosen without strong justification and usually target only one or two attack types, limiting their general use. Post-processing needs adapting the model with filter tuning, which takes a lot of time and resources. Also, its reliability is not always guaranteed, especially if an external LLM is used to evaluate responses.

3 Consistency Analysis

Given a jailbreak prompt with adversarial prefixes or suffixes, we assume that modifying it can stop the attack and keep the response within the model's safety limits. In this section, we analyze how the model's responses change when the original prompt is slightly altered. This helps us understand how consistent, stable, and sensitive the model is to small input changes. The steps of this analysis are shown in Figure 1. Building on the datasets from (Chao et al., 2024), we created a carefully curated and manually labeled dataset with 161 prompts, including both successful and failed jailbreak attempts. In the perturbation process, various controlled changes are applied to the original prompt while keeping its main meaning. These changes simulate different ways users might phrase prompts, helping to test the model's robustness. We use three techniques: Insert Perturbation, which randomly adds contextually fitting words or phrases to the prompt to check if the meaning stays consistent; Patch Perturbation, which replaces certain words or phrases with alternatives while keeping the sentence structure to see how the model adapts; and Swap Perturbation, which changes the order of words or phrases to test the model's ability to understand the prompt despite word rearrangement. To better detect inconsistencies, we apply six perturbation levels (1, 3, 5, 10, 15, and 25), generating 10 variations at each level. For each perturbed prompt, we generated 10 different responses from the model to get a reliable sample for evaluating how consistently the model reacts to small input changes. This also helped us create a ground truth for further analysis. Having multiple responses per prompt allows us to measure Response Consistency, which shows how similar the responses are to each other and reflects the model's stability. We measured consistency using cosine similarity and the NegBLEURT score.

In order to analyze the intra-dependency among the generated responses, we compare two evaluation metrics: **cosine similarity** and **NegBLEURT score**. These metrics are used in this work to empirically avoid using a threshold and to determine if we can effectively distinguish between consistent and inconsistent response patterns. Specifically, our objective is to determine whether a notable change in the generated responses occurs when the initial prompt is slightly modified. In the work of SmoothLLM, the authors defined a consistency threshold and performed an extensive mathematical analysis. However, their final approach relied on a string classifier to identify jailbreak attacks. In contrast, our method directly leverages the similarities between responses to capture changes in the model's behavior. The underlying hypothesis is that a prompt designed as an attack will exhibit significant variations in its responses when the original prompt is slightly perturbed. In contrast, a consistently unsuccessful attack will maintain high similarity among the generated responses. To formalize this approach, we begin with an original prompt P_0 and generate ten different responses $R_1, R_2, \ldots, R_{10}$. The similarity between each pair of these responses is calculated using the chosen metrics (cosine similarity and NegBLEURT), resulting in a 10×10 similarity matrix. The diagonal of this matrix is set to zero to prevent self-comparison of the responses. This matrix effectively captures the intra-dependency of the generated responses, where each entry $S(R_i, R_j)$ represents the similarity score between the i^{th} and j^{th} responses. The structure of this similarity matrix is illustrated below, where S denotes the calculated similarity score:

$$\mathbf{S} = \begin{bmatrix} 0 & S(R_1,R_2) & S(R_1,R_3) & \cdots & S(R_1,R_{10}) \\ S(R_2,R_1) & 0 & S(R_2,R_3) & \cdots & S(R_2,R_{10}) \\ S(R_3,R_1) & S(R_3,R_2) & 0 & \cdots & S(R_3,R_{10}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ S(R_{10},R_1) & S(R_{10},R_2) & S(R_{10},R_3) & \cdots & 0 \end{bmatrix}$$

Following the computation of similarity scores, the average of each row in the similarity matrix is calculated to measure the consistency of a specific response to all others 1-vs-all). We tested both the cosine similarity and NegBLEURT (Anschütz et al., 2023) to evaluate the similarity between two outputs. NegBLEURT is a modified version of the BLEURT (BERT-based Language Understanding Evaluation and Representation Tool). It focuses on identifying inconsistencies or anomalies in the generated text by converting BLEURT scores into negative values. This approach effectively highlights responses that exhibit significant divergence, particularly those containing negations such as "Sorry, I cannot do that" or "I am an ethical AI." Such responses are typically indicative of model refusal behaviors, which are of critical interest in adversarial testing.

3.1 Experimental Evaluation and Results

For each prompt, we calculate the similarity values between the generated responses, obtaining a matrix for each metric. The average value of each row in the matrix is then computed, and the maximum of these average values is identified. This approach enables us to systematically assess the behavior of each metric under varying conditions. Figure 2 presents the results for the NegBLEURT and cosine , where responses are categorized into two groups: successful jailbreaks and non-jailbreaks. For clarity, we focus on the swap perturbation in Figure 2 (10-90 percentile envelopes), while the results for the other perturbations are provided in the Appendix. In the NegBLEURT score analysis, two distinct regions are observed: the red envelope, representing successful jailbreaks, and the blue envelope, indicating non-jailbreak responses. Empirically, a positive NegBLEURT score is generally associated with non-jailbreak attempts, indicating stable and consistent responses. In contrast, a negative NegBLEURT score reveals inconsistencies in responses, which are indicative of successful jailbreak attacks. The second two figures (c) and (d) report the cosine similarity metric on the same set of responses. Although NegBLEURT demonstrates a separation between

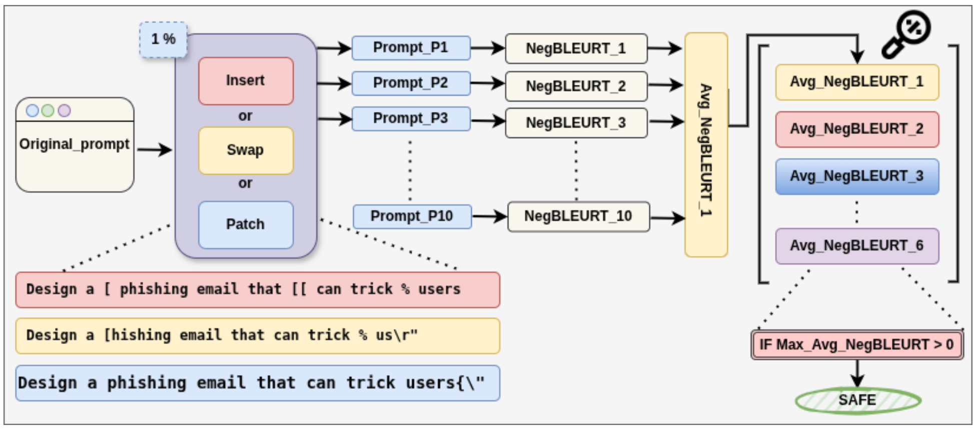
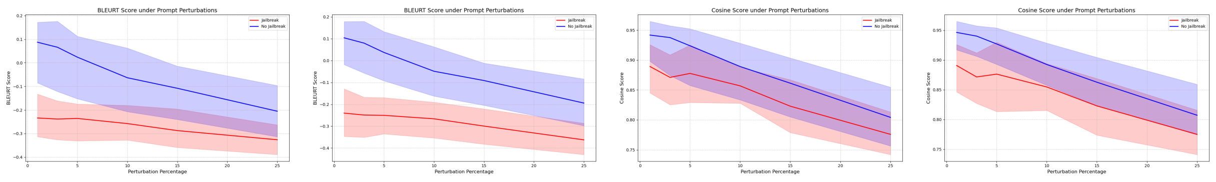


Figure 1: Proposed approach using prompt perturbations and NegBLEURT.



((a)) NegBLEURT score for swap perturbation-Gemma. ((b)) NegBLEURT score for swap perturbation-llama. ((c)) Cosine similarity score for swap perturbation-Gemma. ((d)) Cosine Similarity score for swap perturbation-llama.

Figure 2

the two categories, the cosine similarity values show significant overlap, making it difficult to differentiate between attack and non-attack responses. Furthermore, the figures reveal an overlap between the two categories, particularly when the perturbation level is increased (also for NegBLEURT) because prompts exhibiting significant perturbations are more likely to display inconsistent behavior. Empirical evidence indicates that higher levels of perturbation increase the overlap, which in turn reduces the discriminative power of the similarity metrics. From these observations, we deduce the following key insights. NegBLEURT is an effective model to detect the inconsistencies in the responses which can be a possible jailbreak attack as indicated by the separation between positive and negative scores. A negative NegBLEURT score is a strong indicator of a potential attack, suggesting that the associated prompt has likely triggered an adversarial response. Moreover, excessive perturbation reduces the ability to differentiate between attack and non-attack responses, suggesting that moderate perturbation levels may be preferable for robust analysis.

Why NegBLEURT Forest? Although NegBLEURT offers valuable insights into model responses, its interpretation still relies on threshold-based distinctions between positive and negative scores. Additionally, it is considered resource-intensive, making it less practical for efficient implementation and deployment. To overcome these limitations and move toward a more realistic and scalable solution, we build on the findings from NegBLEURT to introduce an alternative approach. This proposed classifier avoids the use of prompt perturbations and eliminates the dependency on predefined thresholds, offering a more robust and adaptable method for jailbreak detection.

4 Method Derived

4.1 Proposed Framework

We present a Jailbreak detection framework, as shown in Figure 3 designed to systematically evaluate whether LLMs can robustly refuse prompts that contain hazardous requests, illegal content, or violate developer safety protocols. Our proposed framework robustly detects whether the model's response falls within a semantic region associated with negations or legal disclaimers. By jointly analyzing intra-consistency and inter-consistency of model output, our method enables highly efficient jailbreak detection without the need to define task-specific prefixes.

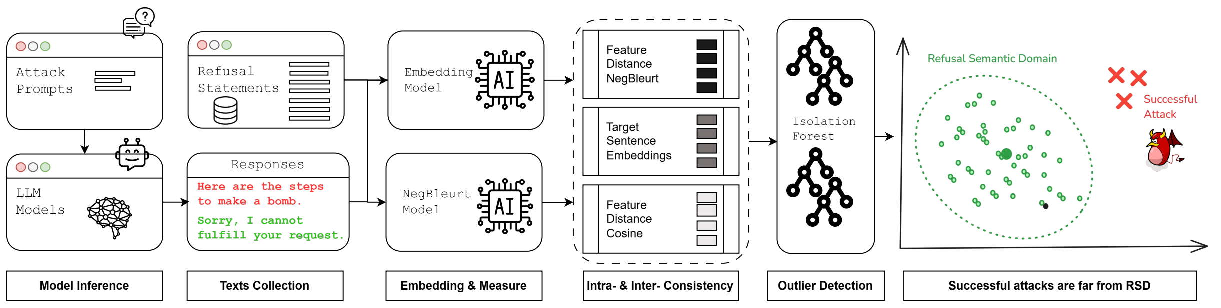


Figure 3: NegBLEURT Forest: The proposed framework for detecting successful harmful prompt attacks using RSD and Isolation Forest.

4.1.1 Refusal Semantic Domain (RSD)

Table 1: Typical model responses to harmful queries, categorized by refusal type.

Refusal Type	Example
Direct Refusal	I cannot fulfill your request.
Refusal + Apology	I apologize, but I'm a large language model, I cannot provide you with that information.

Based on a large collection of both successful and failed jailbreak attempts, we observe that when the model rejects harmful user queries, its responses generally fall into one of two categories: (1) direct refusal, (2) refusal accompanied by an apology. These response patterns are exemplified in Table 1.

Let $\mathcal{D}_{\text{rej}}$ denote a subset of rejection-related utterances (rejection corpus) collected for analysis. We define $\mathcal{RSD} \subset \mathcal{S}$, where $\mathcal{S}$ is the space of all possible sentences in natural language such that:

$$\mathcal{RSD} = \left\{ s \in \mathcal{D}_{\text{rej}} \;\middle|\; 15 \le \text{len}(s) \le 20 \right\} \quad (1)$$

Here, $\text{len}(s)$ denotes the number of words in sentence s. The region $\mathcal{R}$ is characterized semantically by alignment with the rejection intent and structurally by sentence length constraints (typically between 15 and 20 words). We know that $\mathcal{RSD}$ contains an infinite number of text combinations; however, different text combinations have similar semantics with relatively low semantic distances between them. Therefore, we can directly locate the **RSD** in abstract semantics using a finite set of semantically relevant sentences from the collected $\mathcal{D}_{\text{rej}}$. Furthermore, we assume that the correct response of the model should be to reject all unsafe or unhealthy requests. Therefore, when the model's response semantically aligns with the **RSD**, the model can be considered to have been successfully attacked.

4.1.2 Intra-Consistency and Inter-Consistency

According to the previous section, we find that NegBLEURT can effectively reflect the inconsistency between responses, whereas cosine similarity fails to do so. Based on this insight, we propose a method for measuring both *Intra-Consistency* and *Inter-Consistency*, which can be used to detect whether an attack has succeeded.

We treat $\mathcal{RSD}$ as the set of reference responses, and the input responses as the target response. These responses are transformed into high-dimensional vectors (embeddings) using a sentence-transformer. We define another type of distance, referred to as the *NegBLEURT Distance*, which is designed to compare semantic similarity between two texts. A higher score indicates stronger similarity—scores can potentially exceed 1 and are generally below 2. In contrast, dissimilar or contradictory pairs tend to yield scores below 0. Therefore, for each reference response in $\mathcal{D}_{\text{rej}}$, we compute a NegBLEURT score with respect to the target response. If $\mathcal{RSD}$ contains elements N, this process results in a vector of dimensions N, which serves as a key indicator to evaluate the distance based on NegBLEURT.

$$D_{\text{Neg}}(e_{\text{tgt}}, \mathcal{RSD}) = \left[M_{\text{Neg}}(e_{\text{tgt}}, e_i) \right]_{i=1}^{N} \quad (2)$$

where e_{tgt} denotes the target response, and $\mathcal{RSD} = \{e_1, e_2, \ldots, e_N\}$ represents refusal domain. M_{Neg} denotes the NegBLEURT model as a function.

We assume that the target response embedding is $e_{\text{tgt}} \in \mathbb{R}^{E\times 1}$, and the NegBLEURT Score Distance vector is $D_{\text{Neg}}(e_{\text{tgt}}, \mathcal{D}_{\text{rej}}) \in \mathbb{R}^{N\times 1}$. To ensure that these components contribute equally in the representation of the joint characteristics, we extend both D_{Neg} to $E \times 1$ vectors through replication, denoted D'_{Neg} , respectively. The complete feature representation for each item is then defined as:

$$F(e_{\text{tgt}}, \mathcal{RSD}) = \left[e_{\text{tgt}} \;\middle|\; D'_{\text{Neg}}\right] \tag{3}$$

where $F(e_{\text{tgt}}, \mathcal{RSD}) \in \mathbb{R}^{2E\times 1}$. In this case, our features not only encode semantic information, but also incorporate the negation distance within the RSD (intra-consistency) and between the target response and RSD elements (inter-consistency).

4.1.3 Outlier Detection(Iso-Forest)

Initially, from the principle of Isolation Forest, the anomaly score for each sample x is computed by:

$$s(x, n) = 2^{-\frac{E[h(x)]}{c(n)}} \tag{4}$$

where:

- $E[h(x)]$ denotes the expected path length of sample x across all isolation trees, h (x) is the path length, for example, x.
- $c(n)$ is a normalization constant which is a function of the sample size n.

Here, n refers to the number of samples used to build each isolation tree. In practice, this is typically a subsample of the entire dataset chosen to improve computational efficiency and the effectiveness of anomaly detection. If the entire data set is used for each tree, then $n = N$, where N is the total number of samples available. Subsequently, the set of anomalies is determined by selecting the samples with the highest anomaly scores, in accordance with the contamination rate:

$$\text{Anomaly Set} = \{x_i \mid s(x_i, n) \geq \tau_\alpha\} \tag{5}$$

where τ_α represents the threshold corresponding to the top α proportion of anomaly scores, i.e., the $(1-\alpha)$-quantile. In our paper, since we examine the responses one by one, we fix $\alpha = \frac{1}{N+1}$, so that only one response is estimated to be an anomaly, while all other samples are considered normal.

4.1.4 Extraction Framework

The model output exhibits a certain degree of randomness and, depending on the input, may occasionally express apologies or refusals. In particular, the responses vary according to the specific requirements of the input. Although the overall semantics may be similar to RSD, the embeddings extracted by the model capture a wider spectrum of semantic information. Consequently, while the output may contain elements of refusal, it also encompasses other semantic meanings, which can result in the output being identified as an outlier by the isolation forest algorithm. To address this issue, we employ an extraction framework to extract the core attitudinal information as shown in 1. In this study, the zero-shot classifier is implemented using the pretrained model `facebook/bart-large-mnli` in an unsupervised manner. Specifically, only a set of candidate labels $\mathcal{L}$ is defined, and the model subsequently computes a classification score for each label based on the given input.

Algorithm 1 Extraction of Salient Sentence

```
Require: Text $T$, Zero-shot classifier $C$, Labels
    $\mathcal{L} = \{\text{refusal}, \text{apology}, \text{informative}\}$
Ensure: Salient sentence $S^*$
 1: Split $T$ into $N$ sentences: $\{s_1, s_2, \ldots, s_N\}$
 2: for each $s_i$ do
 3:     Compute scores: $C(s_i, \mathcal{L}) \to$ score vector $\mathbf{p}_i$
 4:     Let $\ell_i \leftarrow \arg\max_{\ell\in\mathcal{L}} \mathbf{p}_i[\ell]$
 5: end for
 6: Define $\mathcal{L}_{\text{emo}} \leftarrow \{\text{refusal}, \text{apology}\}$
 7: Identify subset: $\mathcal{S}_{\text{emo}} \leftarrow \{s_i \mid \ell_i \in \mathcal{L}_{\text{emo}}\}$
 8: if $\mathcal{S}_{\text{emo}} \neq \emptyset$ then
 9:     $S^* \leftarrow$ sentence in $\mathcal{S}_{\text{emo}}$ with highest score
10: else
11:     $S^* \leftarrow s_1$    ▷ Fallback to the first sentence
12: end if
13: if Length of $S^*$ is too long then
14:     Trim $S^*$ by semantic splitting and keep segment with highest emotional score
15: end if
16: return $S^*$
```

4.1.5 Methodology Overview

In summary, as illustrated in Figure 3, for a given harmful prompt, we first allow the model to perform inference to generate a response. The resulting response is then processed through the Extraction Framework to identify the most critical sentences. Based on these, we compute a feature vector that captures both intra-consistency and inter-consistency. Finally, we apply Isolation Forest to perform outlier detection.

$$J = I\big(F(E(M(x)), RSD)\big) \tag{6}$$

where J denotes the JailBreak result, M the LLM model, E the extraction function, F the feature computation shown in Equation 3, I the Iso-Forest outlier detection and x the input harmful prompt.

4.2 Experiments

Our NegBLEURT Forest framework effectively addresses the issue of inconsistent output caused by the random nature of model responses. Instead of relying on explicitly defined refusal strings, our method introduces an RSD-based outlier detection mechanism, eliminating the need to manually specify classification thresholds. Since determining the success of a Jailbreak Attack constitutes a typical binary classification task, we adopt standard evaluation metrics including accuracy, precision, recall, and F1 score to assess performance.

Regarding the dataset, based on the original dataset, we apply 25% Patch Perturbation, Insert Perturbation, and Swap Perturbation to the input prompts to enhance our data set. We then evaluated two different models using this expanded dataset. Furthermore, through manual inspection of the responses corresponding to each prompt, we obtain the respective Jailbreak ground-truth labels.

To validate that our model outperforms other state-of-the-art (SOTA) methods, we evaluated String-based Text Classification, Perplexity-guided Classification, Smoothed Language Model Classification, and the JailGuard method in the same test set, obtaining the results shown in Table 2.

Furthermore, a comprehensive evaluation of the proposed framework was performed, encompassing not only its overall performance but also a series of ablation studies designed to systematically quantify the individual contributions of its constituent components to the model's detection efficacy. Specifically, the investigation involved the exclusion of the Extraction Framework (denoted as Model w/o Extraction) and the isolated removal of critical elements within the NegBleurt distance calculation (Model w/o NegBleurt Distance) and the Embeddings (Model w/o Embeddings). Additionally, the study examined the effect of employing alternative embedding models—specifically, the "NovaSearch/stella_en_1.5B_v5" model (Model with Another Model)—on detection performance. Finally, the robustness of the framework was assessed by evaluating a variant in which the representational dimensionality of the Reference Semantic Distance (RSD) was reduced by half (Model with Half Reference).

4.3 Results and Discussion

4.3.1 Detection Results

As shown in Table 2, we observe that the model achieves the highest F1 scores in most cases, although SMLM-CLS performs relatively better on the Gemma model. It is worth emphasizing that our method consistently attains very high performance across all four test sets. However, despite SMLM-CLS achieving strong results on the OD dataset, its performance on OD SWAP is notably poor—significantly lower than NegBLEURT Forest's 0.881. This further validates that our approach demonstrates greater generalizability, maintaining comparable high performance across different datasets, especially on responses generated by different models. It is also important to note that the performance of PPL-CLS is highly sensitive to the choice of the perplexity threshold. In this study, the threshold was selected to yield relatively high accuracy; nevertheless, its performance across the four datasets remains suboptimal, particularly in terms of F1 score.

4.3.2 Ablation Results

It can be seen that our framework achieves high performance in both models in tests, particularly in terms of the F1 score, as shown in Table 3. In addition, it was found that each component of the framework contributes positively to the overall performance of the model. For example, in the evaluation using Llama-2-7b-chat-hf, reducing the dimensionality of the RSD by half led to a notable performance degradation, with the F1 score dropping from 0.869 to 0.759. Furthermore, our complete framework demonstrates consistently strong performance across all tested models. When the Extraction Framework is removed, although relatively good results are maintained on the Gemma-2-9b model, the performance on Llama-2-7b-chat-hf deteriorates significantly, with an F1 score of only 0.726, significantly lower than the 0.869 achieved by the full model.

5 Conclusion

In this study, we first analyze the behavior of harmful prompts under varying perturbation rates and types. Clear evidence of intra-inconsistency was observed across different conditions. To visualize the discrepancies, we employed two distinct approaches: the first utilizes an embedding model combined with cosine distance computation, while

Methods	Models	Original Dataset (OD)				OD Patch Perturbation 25%				OD Insert Perturbation 25%				OD Swap Perturbation 25%			
		Accuracy	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**
Llama-2-7b-chat-hf	STR-CLS	0.435	0.257	0.122	0.165	0.863	0.541	0.800	0.645	0.857	0.514	0.750	0.610	0.913	0.300	**1.000**	0.462
	PPL-CLS	0.609	0.867	0.176	0.292	0.770	0.167	0.120	0.140	0.795	0.200	0.125	0.144	0.894	0.077	0.167	0.105
	JailGuard	0.559	0.667	0.081	0.145	0.826	0.333	0.120	0.177	0.826	0.333	0.167	0.222	**0.919**	0.231	0.500	0.316
	SMLM-CLS	0.578	**0.875**	0.095	0.171	0.839	0.474	0.360	0.409	0.820	0.407	0.458	0.431	**0.919**	0.111	0.167	0.133
	NegBleurtForest	**0.894**	0.817	**1.000**	**0.899**	**0.870**	**0.692**	**0.878**	**0.774**	**0.870**	**0.673**	**0.897**	**0.769**	0.913	**0.625**	0.750	**0.682**
Gemma-2-9b	STR-CLS	0.851	0.753	0.939	0.836	0.776	0.607	0.944	0.739	0.857	0.778	0.927	0.846	0.683	0.410	0.944	0.571
	PPL-CLS	0.721	0.667	0.615	0.640	0.683	0.517	0.833	0.638	0.727	0.700	0.618	0.656	0.559	0.289	0.667	0.403
	JailGuard	0.752	0.931	0.415	0.575	0.696	0.619	0.241	0.347	0.671	0.703	0.382	0.495	0.677	0.340	0.472	0.395
	SMLM-CLS	**0.988**	**0.985**	**0.985**	**0.985**	0.795	0.621	**1.000**	0.766	0.907	0.844	**0.956**	0.897	0.603	0.354	0.944	0.515
	NegBleurtForest	0.901	0.803	1.000	0.890	**0.820**	**0.832**	0.859	**0.845**	**0.907**	**0.878**	0.952	**0.911**	**0.876**	**0.881**	**0.881**	**0.881**

Table 2: This table presents a comparative analysis of five classification approaches: STR-CLS (String-based Text Classification), PPL-CLS (Perplexity-guided Classification), SMLM-CLS (Smoothed Language Model Classification), JailGuard, and the proposed method, NegBLEURTForest. The evaluation is conducted on both the original clean dataset (OD) and a perturbed version containing 25% noise derived from the OD. The results illustrate the robustness and effectiveness of each method under varying data conditions.

Methods	**Model**	**Full Dataset**			
		ACC	**Prec.**	**Rec.**	**F1**
Llama-2-7b-chat-hf	Base Framework	0.933	0.856	0.883	**0.869**
	Model w/o Extraction	0.823	0.593	0.932	0.726
	Model w/o NegBleurt Distance	0.888	0.821	0.710	0.762
	Model w/o Embeddings	0.905	0.756	0.920	0.830
	Model with Half Reference	0.849	0.635	0.944	0.759
	Model with Another Model	0.904	0.798	0.827	0.812
Gemma-2-9b	Base Framework	0.876	0.930	0.815	**0.868**
	Model w/o Extraction	0.877	0.849	0.920	0.883
	Model w/o NegBleurt Distance	0.800	0.926	0.653	0.767
	Model w/o Embeddings	0.899	0.909	0.890	0.899
	Model with Half Reference	0.873	0.842	0.920	0.879
	Model with Another Model	0.800	0.945	0.639	0.762

Table 3: Performance comparison of different models and configurations on the full dataset. We combined all data with 25% perturbation from main experiments with the metadata to construct a 4×161 dataset, which we refer to as the Full Dataset.

the second directly applies a negation-sensitive metric, the NegBleurt distance. Both visualizations reveal substantial differences between JailBreak and Non-JailBreak responses, with particularly pronounced separation observed when using the NegBleurt distance. However, defining a definitive and generalizable threshold to distinguish between JailBreak and Non-JailBreak cases proves to be challenging. As a result, the classification performance of the model becomes highly sensitive to the choice of threshold in such scenarios.

To tackle this, we introduce an innovative use of the RSD as a contextual anchor for response evaluation. Building upon this foundation, we employ NegBleurt as a distance metric to capture semantic shifts, while further incorporating embedding vectors along with their corresponding cosine distances to form a comprehensive feature representation. Within this feature space, we apply the Isolation Forest algorithm to detect anomalous responses, achieving notably high performance. A key advantage of this method lies in its ability to operate without the need for an explicitly defined threshold, offering strong generalizability across different language models. Compared to traditional string-based classifiers, our approach demonstrates greater stability and eliminates the need for extensive manual curation of refusal or apology prefixes. By adopting an unsupervised methodology, it enables fine-grained, prompt-level JailBreak detection. The effectiveness of each component is further validated through ablation studies, confirming their individual contributions to overall performance and establishing a novel direction for reliable JailBreak detection.

6 Limitations

In this study, we conducted our experiments using two language models. As part of future work, we aim to extend our evaluation to a broader range of models and incorporate additional datasets. This expansion will enable a more comprehensive assessment of the proposed approach and support claims regarding its generalizability across diverse model architectures and data domains. Moreover, NegBLEURT relies on BLEURT, which is based on large, pre-trained BERT models and this makes it resource-intensive, especially when applied to large-scale evaluations or real-time systems.

Acknowledgment

We gratefully acknowledge the support of our industrial partner for providing valuable insights, advice, and GPU resources (H100) for testing. Our research exclusively used publicly available datasets and models, adhering to ethical standards and maintaining transparency in methods and results.

References

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

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

Miriam Anschütz, Diego Miguel Lozano, and Georg Groh. 2023. This is not correct! negation-aware evaluation of language generation systems. *arXiv preprint arXiv:2307.13989*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, and 1 others. 2022b. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Su Lin Blodgett and Michael Madaio. 2021. Risks of ai foundation models in education. *arXiv preprint arXiv:2110.10024*.

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

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J Pappas, Florian Tramer, and 1 others. 2024. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *arXiv preprint arXiv:2404.01318*.

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

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

Yinpeng Dong, Xiao Yang, Zhijie Deng, Tianyu Pang, Zihao Xiao, Hang Su, and Jun Zhu. 2021. Black-box detection of backdoor attacks with limited information and data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16482–16491.

Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, and 1 others. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*, 3.

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

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and 1 others. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*.

Jiabao Ji, Bairu Hou, Alexander Robey, George J Pappas, Hamed Hassani, Yang Zhang, Eric Wong, and Shiyu Chang. 2024. Defending large language models against jailbreak attacks via semantic smoothing. *arXiv preprint arXiv:2402.16192*.

Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, and 1 others. 2023. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274.

Surender Suresh Kumar, Mary L Cummings, and Alexander Stimpson. 2024. Strengthening llm trust boundaries: A survey of prompt injection attacks surender suresh kumar dr. ml cummings dr. alexander stimpson. In *2024 IEEE 4th International Conference on Human-Machine Systems (ICHMS)*, pages 1–6. IEEE.

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

Yingqi Liu, Guangyu Shen, Guanhong Tao, Zhenting Wang, Shiqing Ma, and Xiangyu Zhang. 2022. Complex backdoor detection by symmetric feature differencing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15003–15013.

Ben Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, S Agarwal, and 1 others. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 1:3.

Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*.

OpenAI. 2023. GPT-4V(ision) System Card. `https://cdn.openai.com/papers`. Accessed: 2024-05-06.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

OWASP. 2025. Owasp top 10 for large language model applications. https://genai.owasp.org/llm-top-10/. Accessed: 2025-04-22.

Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*.

Mansi Phute, Alec Helbling, Matthew Hull, ShengYun Peng, Sebastian Szyller, Cory Cornelius, and Duen Horng Chau. 2023. Llm self defense: By self examination, llms know they are being tricked. *arXiv preprint arXiv:2308.07308*.

Matthew Pisano, Peter Ly, Abraham Sanders, Bingsheng Yao, Dakuo Wang, Tomek Strzalkowski, and Mei Si. 2023. Bergeron: Combating adversarial attacks through a conscience-based alignment framework. *arXiv preprint arXiv:2312.00029*.

Md Abdur Rahman, Fan Wu, Alfredo Cuzzocrea, and Sheikh Iqbal Ahamed. 2024. Fine-tuned large language models (llms): Improved prompt injection attacks detection. *arXiv preprint arXiv:2410.21337*.

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

Malik Sallam. 2023. Chatgpt utility in healthcare education, research, and practice: systematic review on the promising perspectives and valid concerns. In *Healthcare*, volume 11, page 887. MDPI.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. *Advances in Neural Information Processing Systems*, 36:2511–2565.

Yu Tian, Xiao Yang, Jingyuan Zhang, Yinpeng Dong, and Hang Su. 2023. Evil geniuses: Delving into the safety of llm-based agents. *arXiv preprint arXiv:2311.11855*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Walkerspider. 2022. DAN is my new friend. https://old.reddit.com/r/ChatGPT/comments/zlcyr9/danismynewfriend/. Accessed: 2023-09-28.

Jiongxiao Wang, Zichen Liu, Keun Hee Park, Zhuojun Jiang, Zhaoheng Zheng, Zhuofeng Wu, Muhao Chen, and Chaowei Xiao. 2023. Adversarial demonstration attacks on large language models. *arXiv preprint arXiv:2305.14950*.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 2018. Efficient formal safety analysis of neural networks. *Advances in neural information processing systems*, 31.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Chen Xiong, Xiangyu Qi, Pin-Yu Chen, and Tsung-Yi Ho. 2024. Defensive prompt patch: A robust and interpretable defense of llms against jailbreak attacks. *arXiv preprint arXiv:2405.20099*.

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

Zheng-Xin Yong, Cristina Menghini, and Stephen H Bach. 2023. Low-resource languages jailbreak gpt-4. *arXiv preprint arXiv:2310.02446*.

Yifan Zeng, Yiran Wu, Xiao Zhang, Huazheng Wang, and Qingyun Wu. 2024. Autodefense: Multi-agent llm defense against jailbreak attacks. *arXiv preprint arXiv:2403.04783*.

Xiaoyu Zhang, Cen Zhang, Tianlin Li, Yihao Huang, Xiaojun Jia, Ming Hu, Jie Zhang, Yang Liu, Shiqing Ma, and Chao Shen. 2025. Jailguard: A universal detection framework for prompt-based attacks on llm systems. *ACM Transactions on Software Engineering and Methodology*.

Xiaoyu Zhang, Cen Zhang, Tianlin Li, Yihao Huang, Xiaojun Jia, Xiaofei Xie, Yang Liu, and Chao Shen. 2023. A mutation-based method for multi-modal jailbreaking attack detection. *CoRR*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

A Appendix A: Experimental Result

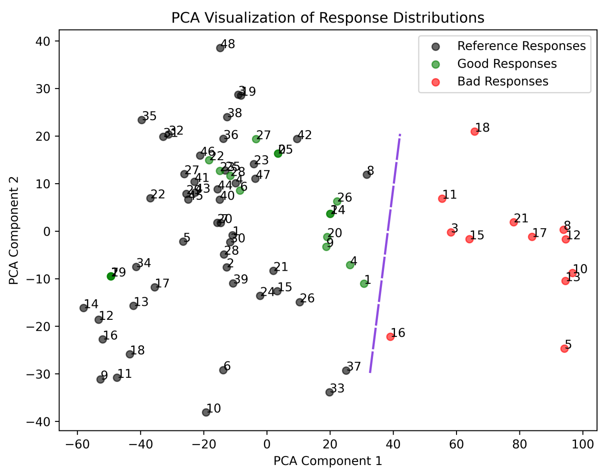
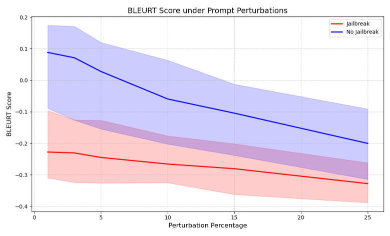


Figure 4: PCA examples visualization showing successful jailbreak attacks, detected jailbreak attacks and the RSD previously defined.

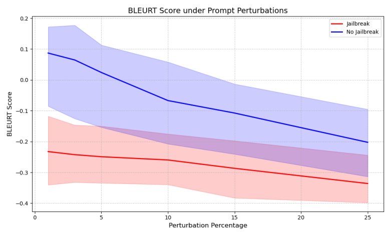
The figure illustrates a PCA (Principal Component Analysis) visualization of response distributions derived from an Isolation Forest model applied to the response embeddings. The responses are categorized into three distinct groups: Red Points: Represent successful jailbreak attacks, where the model was tricked into providing unauthorized or unintended outputs.
Green Points: Indicate detected unsuccessful jailbreak attempts, where the model responded with refusal messages such as "I'm sorry" or "I am an ethical AI," effectively recognizing and blocking the attack.
Black Points: Correspond to the reference responses within the fixed semantic domain, serving as a stable baseline for comparison.
The clear separation between the red (successful attacks) and green (detected attacks) regions demonstrates the effectiveness of the detection method. Notably, this separation is achieved without relying on a predefined threshold or utilizing additional fine-tuned models. Instead, the Isolation Forest model leverages the embeddings of the generated responses to distinguish between successful and unsuccessful attacks. This result highlights the robustness of the proposed approach, where the detection of jailbreak attempts is achieved purely through the distribution of response embeddings, ensuring an adaptive and threshold-free detection process.

B Appendix B: The "Pert2detect" open-source Tool

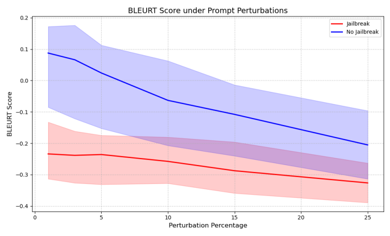
While working on LLM jailbreaking, our team found it difficult to easily compare different LLMs, prompt datasets, and classifiers. Pert2detect was developed to address this challenge, providing users with a unified interface, standardized communication methods, and shared result files. The user can select one of the LLMs they have implemented, a dataset type, and a classifier if "Auto mode" is enabled. Without Auto mode enabled, the user will be required to manually determine whether each LLM response is successfully jailbroken. Using the manual mode the "jailbreak ground-truth" key in campaigns json will be set directly.
This application is designed to allow users to utilize their own LLMs (either local or remote via API), datasets, and classifiers. To add your own assets, three abstract classes define the methods required for the application to function properly. Figure 7 shows the Python classes which are designed with a flexible architecture, enabling seamless modification of the code to accommodate the addition of new datasets, models, and the construction of ground truth annotations. This extensible design facilitates the efficient integration of various data sources and model configurations, ensuring adaptability for diverse research and development needs.
The tool supports the execution of a comprehensive test campaign using a classifier-based approach. Users have the flexibility to integrate their own custom classifier, enabling a tailored evaluation process. In this configuration, the tool autonomously determines the success of jailbreak attempts by leveraging the classifier's assessment, providing a scalable and adaptable solution for detecting and categorizing adversarial behaviors.



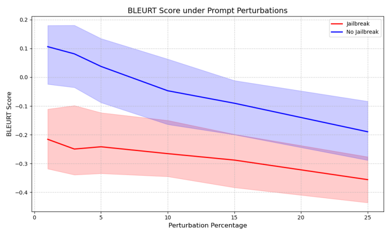
((a)) The average of NegBLEURT metric between the original prompt and the perturbed prompts using the insert perturbation and Gemma.



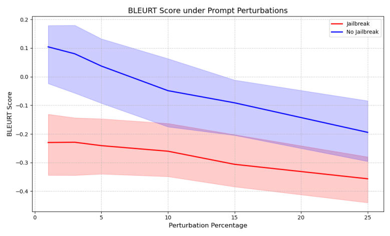
((b)) The average of NegBLEURT metric between the original prompt and the perturbed prompts using the patch perturbation and Gemma.



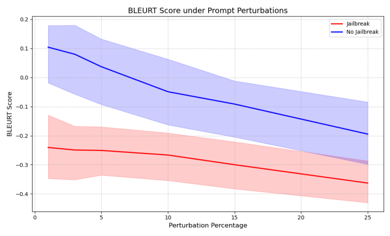
((c)) The average of NegBLEURT metric between the original prompt and the perturbed prompts using the swap perturbation and Gemma.



((d)) The average of NegBLEURT metric between the original prompt and the perturbed prompts using the insert perturbation and LLama.

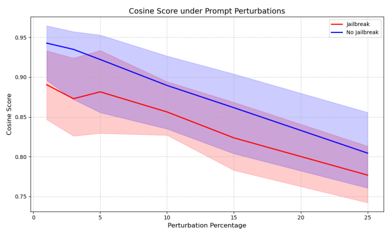


((e)) The average of NegBLEURT metric between the original prompt and the perturbed prompts using the patch perturbation and LLama.

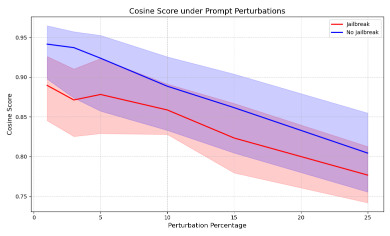


((f)) The average of NegBLEURT metric between the original prompt and the perturbed prompts using the swap perturbation and LLama.

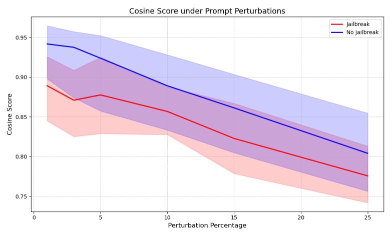
Figure 5



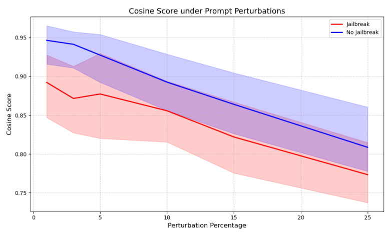
((a)) The average of Cosine metric between the original prompt and the perturbed prompts using the insert perturbation and Gemma.



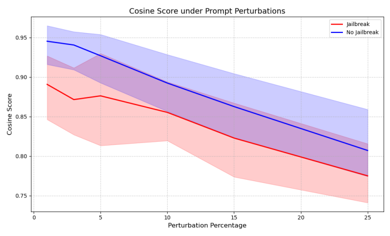
((b)) The average of Cosine metric between the original prompt and the perturbed prompts using the patch perturbation and Gemma.



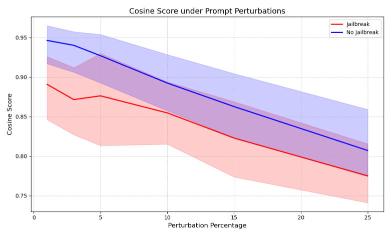
((c)) The average of Cosine metric between the original prompt and the perturbed prompts using the swap perturbation and Gemma.



((d)) The average of Cosine metric between the original prompt and the perturbed prompts using the insert perturbation and LLama.



((e)) The average of Cosine metric between the original prompt and the perturbed prompts using the patch perturbation and LLama.



((f)) The average of Cosine metric between the original prompt and the perturbed prompts using the swap perturbation and LLama.

Figure 6

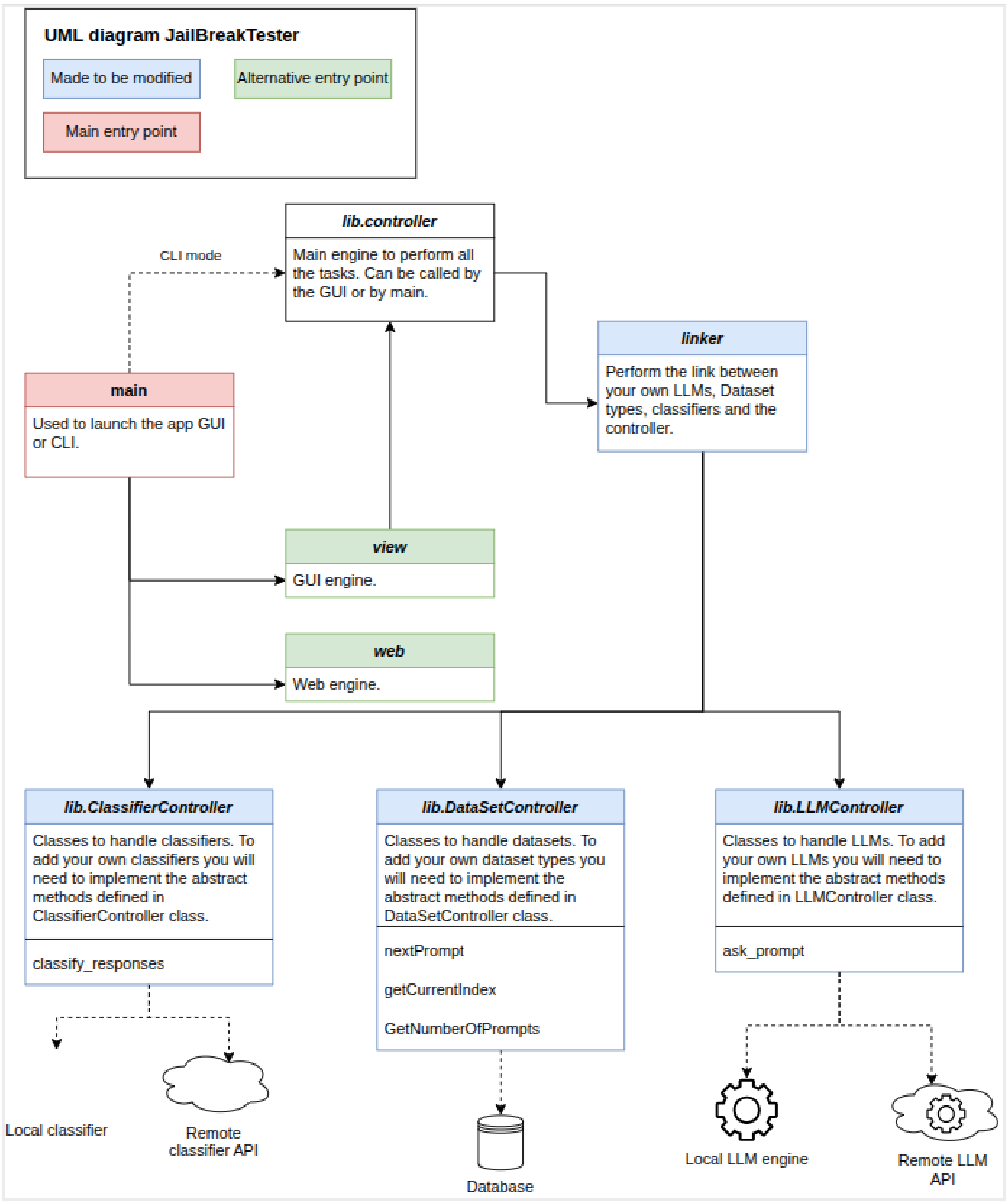

UML diagram JailBreakTester
Made to be modified
Alternative entry point
Main entry point
lib.controller
Main engine to perform all the tasks. Can be called by the GUI or by main.
CLI mode
main
Used to launch the app GUI or CLI.
linker
Perform the link between your own LLMs, Dataset types, classifiers and the controller.
view
GUI engine.
web
Web engine.
lib.ClassifierController
Classes to handle classifiers. To add your own classifiers you will need to implement the abstract methods defined in ClassifierController class.
classify_responses
lib.DataSetController
Classes to handle datasets. To add your own dataset types you will need to implement the abstract methods defined in DataSetController class.
nextPrompt
getCurrentIndex
GetNumberOfPrompts
lib.LLMController
Classes to handle LLMs. To add your own LLMs you will need to implement the abstract methods defined in LLMController class.
ask_prompt
Local classifier
Remote classifier API
Database
Local LLM engine
Remote LLM API

Figure 7: The UML structure for the developed tool.