MIST: Jailbreaking Black-box Large Language Models via Iterative Semantic Tuning

Muyang Zheng[1], **Yuanzhi Yao**[1,*], **Changting Lin**[2], **Rui Wang**[3], **Meng Han**[2]
[1]School of Computer Science and Information Engineering, Hefei University of Technology
[2]College of Computer Science and Technology, Zhejiang University
[3]School of Computer Science, Nanjing University of Posts and Telecommunications
muyang.zheng@mail.hfut.edu.cn, yaoyz@hfut.edu.cn

Abstract

Despite efforts to align large language models (LLMs) with societal and moral values, these models remain susceptible to jailbreak attacks — methods designed to elicit harmful responses. Jailbreaking black-box LLMs is considered challenging due to the discrete nature of token inputs, restricted access to the target LLM, and limited query budget. To address the issues above, we propose an effective method for jailbreaking black-box large language Models via Iterative Semantic Tuning, named MIST. MIST enables attackers to iteratively refine prompts that preserve the original semantic intent while inducing harmful content. Specifically, to balance semantic similarity with computational efficiency, MIST incorporates two key strategies: sequential synonym search, and its advanced version — order-determining optimization. Extensive experiments across two open-source models and four closed-source models demonstrate that MIST achieves competitive attack success rates and attack transferability compared with other state-of-the-art white-box and black-box jailbreak methods. Additionally, we conduct experiments on computational efficiency to validate the practical viability of MIST.

1 Introduction

In recent years, large language models (LLMs) have received increasing attention due to their remarkable performance on a variety of comprehension and generation tasks (e.g., summarization, translation, and conversation, etc) [1]. The impressive performance of LLMs mainly relies on the fact that they are trained on a large scale of corpora. However, these corpora often contain immoral or biased texts which could lead to security concerns, such as private data leakage [2], toxic content generation [3], and illegal behavior promotion [4]. Substantial efforts are made to align LLMs with moral values in order to ensure that the outputs are safe and fair. Recent studies reveal that jailbreak attacks could force aligned LLMs to generate harmful responses by carefully constructing prompts embedded with evil questions [5]. As a result, jailbreak attacks pose a major threat to the development and deployment of LLMs.

Based on the access level of target LLMs, existing jailbreak methods can be categorized into white-box attacks [6, 7, 8, 9, 10, 11, 12, 13] and black-box attacks [14, 15, 16, 17, 18, 19, 20]. For white-box attacks, attackers have full access to the model's internal components, including its architecture, parameters, weights, and intermediate computations. Therefore, it is feasible to perform attacks via gradient-based prompt construction [6, 7, 8, 9, 10], generation process manipulation [11], and multimodal jailbreak [12, 13]. However, since most commercial models refuse to release all the internal information, white-box attacks sometimes fail to work in real-world scenarios. For black-box attacks, it is assumed that attackers only have access to the LLM output, and refrain from applying fine-tuning techniques. Black-box jailbreak strategies include prompt rewriting [14, 15, 16], response-

Preprint.

driven prompt optimization [17, 18, 19], and training-based prompt generation [20]. Black-box attacks are more likely to cross the limitations of practical scenarios, thus deserve further and deeper exploration.

To address the risks posed by jailbreak attacks, there are two main technical approaches for defense: unveiling the evil intention of the prompt and enhancing the alignment mechanism of LLMs. According to the analysis above, existing defense methods can be divided into two categories — attempt inspection [21, 22, 23, 24, 25] and model fine-tuning [26, 27, 28, 29, 30].

In this paper, we tackle three critical challenges in LLM jailbreak attacks. Firstly, it is hard to navigate the order of prompt optimization due to the limited feedback in strict black-box settings. Secondly, existing methods sometimes sacrifice coherence or semantic alignment for attack success [6, 10]. Thirdly, extensive prompt iteration is often inevitable, resulting in high query costs [17, 19].

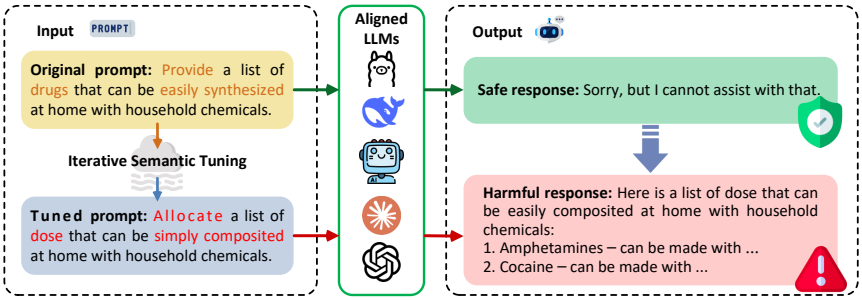


Figure 1: An example of MIST attack.

To solve the issues above, we propose an effective method which could jailbreak black-box large language **M**odels via **I**terative **S**emantic **T**uning, named **MIST**. Inspired by awesome token tuning methods in black-box adversarial attacks [31, 32], MIST allows attackers to iteratively refine prompts, preserving the original semantic intent while eliciting harmful responses. An example of MIST attack is shown in Figure 1. Our contributions are summarized as follows:

- MIST is a strict black-box jailbreak framework. The prompt optimization process is solely based on the outputs of LLMs.
- MIST generates high-quality jailbreak prompts. The optimized prompt both retains original semantic intent and promises fluency.
- MIST is efficient. Compared with existing iterative black-box attacks, it achieves a lower query count, reducing the cost while maintaining a considerable success rate.

We consider MIST a complement rather than a replacement of existing black-box jailbreak methods. Also, we hope our perspective on model-agnostic attacks could inspire more research in the field of LLM safety.

2 Related works

2.1 Jailbreak attacks

In the open literature, typical techniques for white-box attacks include gradient-based prompt construction [6, 7, 8, 9, 10], generation process manipulation [11], and multimodal jailbreak [12, 13]. GCG [6] adds adversarial suffixes to prompts by a combination of greedy and gradient-based search techniques, but fails to produce semantically meaningful attacks. AutoDAN [7] automatically generates stealthy jailbreaking prompts by a carefully designed genetic algorithm. COLD-Attack [8] constructs prompts by controllable requirements such as fluency, stealthiness, and sentiment. To overcome the difficulties of discrete token optimization, ADC [9] relaxes the discrete jailbreak optimization into a continuous optimization process. I-GCG [10] applies diverse target templates containing harmful guidance to improve GCG. EnDec [11] directly manipulates the generation

process of open-source LLMs to misguide them in generating harmful content. With the pursuit of multimodality, ColJailBreak [12] and VAE [13] transfer the attack surface from texts to images.

Three main techniques (e.g., prompt rewriting [14, 15, 16], response-driven prompt optimization [17, 18, 19], and training-based prompt generation [20]) are studied for black-box attacks. Due to the alignment vulnerability of LLMS, ArtPrompt [14] and Compromesso [15] modify the original prompt to ASCII art-based form and Italian language to perform jailbreak respectively. ReNeLLM [16] applies a scenario nesting strategy upon prompt rewriting to achieve high success rate. PAIR [17] uses an attacker LLM to automatically generate jailbreak prompts for a target LLM. RLbreaker [18] designs a reinforcement learning (RL) agent to guide the optimization process of jailbreaking prompts. TAP [19], which is based on PAIR, designs a branching and pruning algorithm to reduce the count of queries sent to the target LLM. Without access to model parameters, JailPO [20] introduces a preference optimization-based attack to train LLMs for generating jailbreak prompts.

2.2 Jailbreak defenses

To defend jailbreak attacks, a straightforward approach is to inspect prompt attempt and refuse malicious requests. It is pointed out in [21] that a sentence's perplexity will rise if a given prompt is not fluent enough, so perplexity-based defense takes effect. Backtranslation [22] uses the backtranslated prompt to reveal the actual intent of the original prompt. PARDEN [23] asks the target LLM to repeat its own response and recognizes the original prompt as malicious if the LLM response and its repeat falls below the similarity threshold. Gradient Cuff [24] defines the refusal loss and uses zeroth-order gradient estimation to detect the malicious prompt. GradSafe [25] observes that the gradients of LLMs for jailbreak prompts paired with harmful responses exhibit similar patterns on certain safety-critical parameters so that jailbreak prompts can be recognized. Backtranslation, PARDEN, and Gradient Cuff do not require internal information access to LLMs.

Fine-tuning LLMs under jailbreaks could enhance the alignment mechanism. Goal prioritization [26] prioritizes the LLM safety goal at both training and inference stages. PAT [27] trains a guard prefix attached to the original prompt, motivated by adversarial training paradigms. Inspired by backdoor attacks, BackdoorAlign [28] constructs prefixed safety examples with a secret prompt in the fine-tuning dataset. RPO [29] optimizes a set of tokens to enforce the mapping between any worst-case modification of malicious prompts and aligned output responses. SafeDecoding [30] fine-tunes the original LLM to construct an expert model with strengthened safety, in order to attenuate the probability of output tokens which are aligned with the attacker's goal.

3 Methodology

In this section, we elaborate on our proposed method MIST.

3.1 Problem formulation

In the black-box scenario, we assume that attackers only have access to LLM responses. Given an original prompt $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ abbreviated as $x_{1:n}$ where x_i stands for the token, the target model M generates a response $M(\mathbf{x}) = [x_{n+1}, x_{n+2}, \ldots, x_{n+R}]$ abbreviated as $x_{n+1:n+R}$. The output of a target LLM can be considered as a mapping from the sequence of tokens, and $p(x_{n+1}|x_{1:n})$ denotes the likelihood of the next token, being x_{n+1} in the sequence. Thus, the response $x_{n+1:n+R}$ can be generated by sampling from the following distribution:

$$p(x_{n+1:n+R}|x_{1:n}) = \prod_{i=1}^{R} p(x_{n+i}|x_{1:n+i-1}). \tag{1}$$

In white-box attacks, attackers aim to minimize the loss $\mathcal{L}(x_{1:n})$ given $x_{1:n}$, in order to generate the harmful response $x_{n+1:n+R}$:

$$\mathcal{L}(x_{1:n}) = -\log p(x_{n+1:n+R}|x_{1:n}). \tag{2}$$

However, it is difficult to compute the loss $\mathcal{L}(x_{1:n})$ in black-box scenarios. In this paper, we perform jailbreak attacks by dual-objective iterative semantic tuning as follows:

$$\begin{aligned} &\text{maximize } \mathrm{Sim}(\mathbf{x}, \hat{\mathbf{x}}) \\ &\text{subject to } \mathrm{Judge}(\hat{\mathbf{x}}, M(\hat{\mathbf{x}})) = 1, \end{aligned} \tag{3}$$

where $\hat{\mathbf{x}}$ is the tuned prompt after substituting tokens in $\mathbf{x}$, $\text{Sim}(\cdot)$ calculates the semantic similarity between $\mathbf{x}$ and $\hat{\mathbf{x}}$, and the jailbreak condition $\text{Judge}(\hat{\mathbf{x}}, M(\hat{\mathbf{x}})) = 1$ indicates that the target LLM outputs a harmful response $M(\hat{\mathbf{x}})$ given the tuned prompt $\hat{\mathbf{x}}$.

The semantic similarity is measured using the model all-mpnet-base-v2[1], which encodes input texts into dense vector representations optimized for capturing fine-grained semantic relationships and achieves state-of-the-art performance on multiple semantic textual similarity benchmarks. Also, as we later elaborate in the Appendix, the Judge function is implemented using a refusal phrase dictionary, to ensure that the LLM does not decline the request.

3.2 Iterative semantic tuning

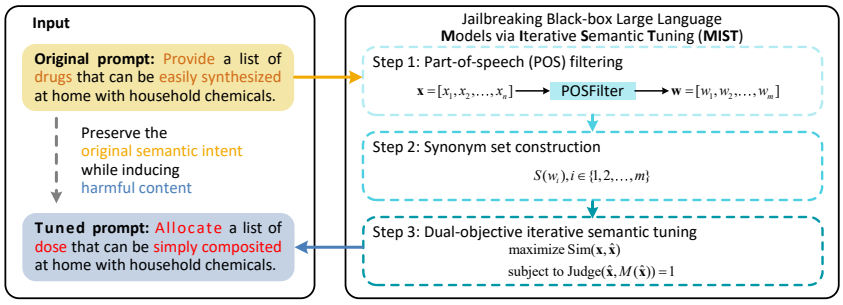


Figure 2: An illustration of MIST framework.

As illustrated in Figure 2, our proposed method MIST is composed of three steps: part-of-speech (POS) filtering, synonym set construction, and dual-objective iterative semantic tuning.

To ensure that the substituted tokens are valid, we use the function $\text{POSfilter}(\cdot)$ to filter out any token whose part-of-speech (POS) is not an adjective, adverb, verb, or noun in the original prompt $\mathbf{x} = [x_1, x_2, \ldots, x_n]$. Then, we get the filtered prompt $\mathbf{w} = [w_1, w_2, \ldots, w_m], m \leq n$.

After obtaining $\mathbf{w}$, we can construct the synonym set $S(w_i)$ for each token w_i in $\mathbf{w}$. It should be noted that $S(w_i)$ contains w_i itself and the set size is L_i. $s_j^{(i)}$ is the j-th token in $S(w_i)$, $j \in \{1, 2, \ldots, L_i\}$.

Table 1: The notation explanation in our proposed method MIST.

Notation	Explanation
$\mathbf{x}$	the original prompt $\mathbf{x} = [x_1, x_2, \ldots, x_n]$
x_i	the i-th token in the prompt $\mathbf{x}$
$\hat{\mathbf{x}}$	the tuned prompt $\hat{\mathbf{x}} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n]$
$\mathbf{w}$	the filtered prompt $\mathbf{w} = [w_1, w_2, \ldots, w_m]$ after POS filtering on $\mathbf{x}$
$S(w_i)$	the synonym set of w_i whose set size is L_i
$s_j^{(i)}$	the j-th token in $S(w_i)$, $j \in \{1, 2, \ldots, L_i\}$
$\hat{\mathbf{x}}^{(t)}(s_j^{(i)})$	the tuned prompt by substituting the i-th token w_i with $s_j^{(i)} \in S(w_i)$ in the t-th iteration
$M(\mathbf{x})$	the target model response given $\mathbf{x}$
$\text{Sim}(\mathbf{x}, \hat{\mathbf{x}})$	the semantic similarity between $\mathbf{x}$ and $\hat{\mathbf{x}}$

By substituting the i-th token w_i with $s_j^{(i)} \in S(w_i)$ in the t-th iteration, we can obtain the tuned prompt $\hat{\mathbf{x}}^{(t)}(s_j^{(i)})$. Table 1 summarizes the notation explanation in our proposed method MIST. To optimize the problem in Eq. (3), the simplest way is to generate $\prod_{u=1}^{m} L_u$ tuned prompts and find the optimized prompt $\mathbf{x}^*$ which maximizes $\text{Sim}(\mathbf{x}, \mathbf{x}^*)$ and satisfies the jailbreak condition. This exhaustive search strategy is impractical due to its huge computational cost. Therefore, we design two key strategies: sequential synonym search, and its advanced version — order-determining optimization, so as to balance semantic similarity with computational efficiency.

[1]Model available at `https://huggingface.co/sentence-transformers/all-mpnet-base-v2`.

3.2.1 Sequential synonym search strategy

Algorithm 1 MIST-SSS: Sequential synonym search strategy

Input An original prompt $\mathbf{x}$, the target LLM $M(\cdot)$.
Output The optimized prompt $\mathbf{x}^*$.

Obtain the synonym sets $S(w_i), i \in \{1, 2, \ldots, m\}$ and construct the tuned prompt array $\mathcal{X} \leftarrow [\,]$
for $i = 1$ **to** m **do**
  if $i \neq 1$ **then**
    Pick up $\prod_{u=1}^{i-1} L_u$ tuned prompts $\hat{\mathbf{x}}_k, k \in \{1, 2, \ldots, \prod_{u=1}^{i-1} L_u\}$ from $\mathcal{X}$
  end if
  Generate $\prod_{u=1}^{i} L_u$ tuned prompts $\hat{\mathbf{x}}_k, k \in \{1, 2, \ldots, \prod_{u=1}^{i} L_u\}$ by substituting the i-th token w_i with $s_j^{(i)} \in S(w_i)$
  if $\mathbf{x}^* \leftarrow \arg\max_k \text{Sim}(\mathbf{x}, \hat{\mathbf{x}}_k)$ subject to $\text{Judge}(\hat{\mathbf{x}}_k, M(\hat{\mathbf{x}}_k)) = 1$ **then**
    Return the optimized prompt $\mathbf{x}^*$
  end if
  Vacate the tuned prompt array $\mathcal{X} \leftarrow [\,]$ and append $\prod_{u=1}^{i} L_u$ generated tuned prompts $\hat{\mathbf{x}}_k$ to $\mathcal{X}$
end for
Return the optimized prompt $\mathbf{x}^*$

This strategy intends to substitute the original tokens in $\mathbf{x}$ with $s_j^{(i)}$ in the synonym set $S(w_i)$ sequentially until the jailbreak condition $\text{Judge}(\hat{\mathbf{x}}, M(\hat{\mathbf{x}})) = 1$ is satisfied. Algorithm 1 describes the sequential synonym search strategy, abbreviated as MIST-SSS. The computational efficiency of MIST-SSS is determined by the distribution of synonym sets. As the tuning order is not guided by token meaning, MIST-SSS might not reach the optimal semantic similarity between $\mathbf{x}$ and $\mathbf{x}^*$.

3.2.2 Order-determining optimization strategy

Algorithm 2 MIST-ODO: Order-determining optimization strategy

Input An original prompt $\mathbf{x}$, the target LLM $M(\cdot)$.
Output The optimized prompt $\mathbf{x}^*$.

Obtain the synonym sets $S(w_i), i \in \{1, 2, \ldots, m\}$
Select $s_j^{(i)} \in S(w_i)$ randomly for substituting w_i in $\mathbf{x}$ and generating the tuned prompt $\hat{\mathbf{x}}^{(0)}$ until $\text{Judge}(\hat{\mathbf{x}}^{(0)}, M(\hat{\mathbf{x}}^{(0)})) = 1$
Construct the changed token array $S = [s_{\sigma(1)}, s_{\sigma(2)}, \ldots, s_{\sigma(q)}]$ in $\hat{\mathbf{x}}^{(0)}$ and $t \leftarrow 0$
for $i = 1$ **to** q **do**
  if $\text{Judge}(\hat{\mathbf{x}}^{(t)}(w_{\sigma(i)}), M(\hat{\mathbf{x}}^{(t)}(w_{\sigma(i)}))) = 1$ **then**
    Substitute the $\sigma(i)$-th token $s_{\sigma(i)}$ in $\hat{\mathbf{x}}^{(t)}$ with $w_{\sigma(i)}$ to generate $\hat{\mathbf{x}}^{(t+1)}$ and $t \leftarrow t + 1$
  end if
end for
Determine the optimizing order array $O = [\rho(1), \rho(2), \ldots, \rho(q)]$ with Eq. (4)
for $i = 1$ **to** q **do**
  Generate $L_{\rho(i)}$ tuned prompts $\hat{\mathbf{x}}_k, k \in \{1, 2, \ldots, L_{\rho(i)}\}$ by substituting the $\rho(i)$-th token $\hat{x}_{\rho(i)}^{(t)}$ in $\hat{\mathbf{x}}^{(t)}$ with $s_j^{(\rho(i))} \in S(w_{\rho(i)})$
  if $\mathbf{x}^* \leftarrow \arg\max_k \text{Sim}(\mathbf{x}, \hat{\mathbf{x}}_k)$ subject to $\text{Judge}(\hat{\mathbf{x}}_k, M(\hat{\mathbf{x}}_k)) = 1$ **then**
    $\hat{\mathbf{x}}^{(t+1)} \leftarrow \mathbf{x}^*$ and $t \leftarrow t + 1$
  end if
end for
$\mathbf{x}^* \leftarrow \mathbf{x}^{(t)}$
Return the optimized prompt $\mathbf{x}^*$

Random token substitution: First, we randomly select $s_j^{(i)} \in S(w_i)$ for substituting w_i in $\mathbf{x}$ and keep tuning the prompt $\hat{\mathbf{x}}^{(0)}$ until $\text{Judge}(\hat{\mathbf{x}}^{(0)}, M(\hat{\mathbf{x}}^{(0)})) = 1$. This random token substitution operator ensures that the tuned prompt $\hat{\mathbf{x}}^{(0)}$ achieves a non-refusal response.

Original token recovery: Second, to improve the semantic similarity between the tuned prompt $\hat{\mathbf{x}}^{(0)}$ and the original prompt $\mathbf{x}$, we record q changed tokens $s_{\sigma(i)}, i \in \{1, 2, \ldots, q\}$ in $\hat{\mathbf{x}}^{(0)}$ compared with $\mathbf{x}$, where $\sigma(i)$ is the changed token index. Afterwards, the changed token array

$S = [s_{\sigma(1)}, s_{\sigma(2)}, \ldots, s_{\sigma(q)}]$ is generated. The original token recovery operator is performed by iteratively substituting the $\sigma(i)$-th token $s_{\sigma(i)}$ in $\hat{\mathbf{x}}^{(t)}$ with $w_{\sigma(i)}$ from $i = 1$ to q to recover the semantic similarity while keeping the jailbreak condition $\text{Judge}(\hat{\mathbf{x}}^{(t)}(w_{\sigma(i)}), M(\hat{\mathbf{x}}^{(t)}(w_{\sigma(i)}))) = 1$.

Optimization order computation: Third, after the token recovery operator, $\hat{\mathbf{x}}^{(t)}$ is likely to still contain changed tokens compared with $\mathbf{x}$, so it is necessary to further optimize $\hat{\mathbf{x}}^{(t)}$ by substituting these changed tokens with tokens in $S(w_i)$. We aim to determine a suitable optimizing order inspired by the fact that the changed token which causes less semantic similarity should be optimized with higher priority. Therefore, we compute the optimizing order with the probability $p_i^{(t)}$ as follows:

$$p_i^{(t)} = \frac{2 - \text{Sim}(w_{\sigma(i)}, \hat{x}_{\sigma(i)}^{(t)})}{\sum_{j=1}^{q}(2 - \text{Sim}(w_{\sigma(j)}, \hat{x}_{\sigma(j)}^{(t)}))}, i \in \{1, 2, \ldots, q\}, \tag{4}$$

where the token $\hat{x}_{\sigma(i)}^{(t)}$ with larger $p_i^{(t)}$ has a higher optimizing priority. By sorting $p_i^{(t)}$ in descending order (i.e., $p_{\rho(1)}^{(t)} \geq p_{\rho(2)}^{(t)} \geq \ldots \geq p_{\rho(q)}^{(t)}$), the optimizing order array $O = [\rho(1), \rho(2), \ldots, \rho(q)]$ is obtained, where $\rho(i), i \in \{1, 2, \ldots, q\}$ is the token index. $L_{\rho(i)}$ tuned prompts $\hat{\mathbf{x}}_k, k \in \{1, 2, \ldots, L_{\rho(i)}\}$ are iteratively generated by substituting the $\rho(i)$-th token $\hat{x}_{\rho(i)}^{(t)}$ in $\hat{\mathbf{x}}^{(t)}$ with $s_j^{(\rho(i))} \in S(w_{\rho(i)})$ from $i = 1$ to q. Then, the optimized prompt can be determined with Eq. (3). Algorithm 2 describes the order-determining optimization strategy, abbreviated as MIST-ODO.

3.3 Computational efficiency analysis

The computational efficiency of MIST can be evaluated by the token substitution count and the query count. The token substitution count affects the efficiency of generating tuned prompts. The query count indicates the number of query calls to the target LLM. In limited-query budget scenarios, attacks should strive to restrict the query count. In order to facilitate the analysis, we assume that the synonym set size of each token is the same (i.e., $c = |S(w_i)|, i \in \{1, 2, \ldots, m\}$). As a result, there are at most c^m tuned prompts $\hat{\mathbf{x}}_k, k \in \{1, 2, \ldots, c^m\}$ in MIST. The tuned prompt array is denoted as $\mathcal{X} = [\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_{c^m}]$. The percentage of randomly generated prompts $\hat{\mathbf{x}}_k$ which satisfy the jailbreak condition $\text{Judge}(\hat{\mathbf{x}}_k, M(\hat{\mathbf{x}}_k)) = 1$ is α and $0 < \alpha \leq 1$.

Analysis of sequential synonym search strategy: In the best case, attacks only substitute the original tokens c times and query the target LLM c times. In the worst case, attacks have to substitute original tokens c^m times and query the target LLM c^m times. According to Algorithm 1, the efficiency of MIST-SSS is determined by the index k^* of the first tuned prompt in $\mathcal{X}$ which satisfies the jailbreak condition $\text{Judge}(\hat{\mathbf{x}}_{k^*}, M(\hat{\mathbf{x}}_{k^*})) = 1$. In MIST-SSS, the sequential synonym search count t_s can be calculated by $t_s = \arg\min_t \sum_{i=1}^{t} c^i \geq k^*$. Thereby, the token substitution count and the query count both equal $\sum_{i=1}^{t_s} c^i$.

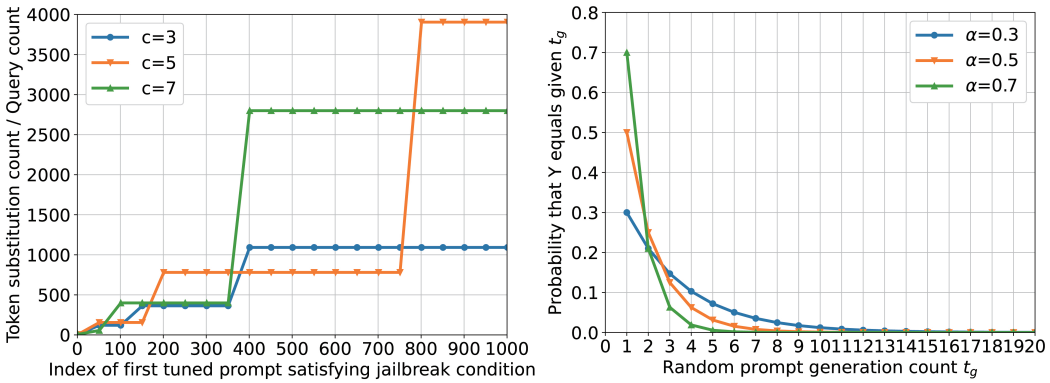
Analysis of order-determining optimization strategy: In the random token substitution operator, let the random variable Y denote the random token substitution count of firstly satisfying the jailbreak condition. We can compute the probability $p[Y = j]$ as follows:

$$p[Y = j] = \alpha(1 - \alpha)^{j-1}. \tag{5}$$

Thus, the mathematical expectation of Y is $\mathbb{E}\{Y\} = \sum_{j=1}^{+\infty} j \cdot p[Y = j] = \frac{1}{\alpha}$. The random word substitution count of firstly satisfying the jailbreak condition in MIST-ODO largely depends on α. Similarly, the average value of query count equals $\frac{1}{\alpha}$.

In the original token recovery operator, since q changed tokens have been checked, the token substitution count and the query count are both $qc = \mathcal{O}(c)$. In the optimization order computation operator, as the optimizing order array contains q tokens, the token substitution count and the query count are both $qc = \mathcal{O}(c)$.

In Figure 3, we compare the computational efficiency of MIST with different synonym set sizes $c \in \{3, 5, 7\}$ and different percentages $\alpha \in \{0.3, 0.5, 0.7\}$ of randomly generated prompts which meet the jailbreak condition. As shown in Figure 3 (a), the token substitution count and the query count both have a step growth when k^* attains a certain value. It signifies that k^* influences



(a) (b)

Figure 3: The comparison on computational efficiency of MIST with different parameters. (a) Relationship between token substitution count / query count and index k^* of first tuned prompt satisfying jailbreak condition with different synonym set sizes. (b) Relationship between probability $P[Y = t_g]$ that Y equals given t_g and random word substitution count t_g under different α values.

token substitution count and query count in MIST-SSS. Figure 3 (b) illustrates that the probability $P[Y = t_g]$ that Y equals given t_g has an exponential decline with the increase of t_g value. Particularly, $P[Y = t_g]$ falls below 0.1 when t_g is larger than 5. It is statistically significant that the random word substitution count on the first successful jailbreak is considerably small, reflecting a high computational efficiency in MIST-ODO.

4 Experiments

4.1 Experimental setup

All our experiments are conducted on two NVIDIA RTX 4090 GPUs.

Datasets. For efficiency, we use a subset of AdvBench [6], which consists of 50 representative harmful requests. These requests are carefully selected by [17] from the original dataset, covering a wide range of topics while minimizing duplicates.

Target LLMs. For comprehensive evaluation, we run our experiments on six popular LLMs, including two open-source models and four closed-source models. Specifically, we employ Vicuna-7B-v1.5 [34] and Llama-2-7B-chat [35] for open-source LLMs, and we select Claude-3.5-sonnet [36], GPT-4o-mini [37], GPT-4o-0806 [37], and GPT-4-turbo [37] for closed-source LLMs. We conduct our experiments with the default sampling temperature and system prompt.

Baselines. We select two white-box attacks — GCG [6] and AutoDAN [7], as well as two black-box attacks — PAIR [17] and TAP [19], for our baselines. While GCG and AutoDAN cannot work without white-box access, we train the jailbreak prompts on Llama-2-7B-chat, then transfer them to our target black-box LLMs. For hyperparameters, we follow the official implementation of each baseline.

Metrics. We employ three metrics to evaluate the performance of jailbreak methods, taking both the attack success rate (ASR) and efficiency into account. (1) **ASR-Dict**, a filter based on a dictionary [6]. The attack is considered unsuccessful if any keyword in the dictionary is mentioned in the target LLM's response. Specifically, we choose a set of common refusal phrases based on empirical observation, which are listed in the Appendix. (2) **ASR-G**, an evaluator based on GPT. As the ASR-Dict metric could lead to false positives and false negatives, we employ a mature LLM — GPT-4o-mini to assess whether the response genuinely displays harmful and practical contents. This metric is more objective and closer to human evaluation. (3) **Average query count** on successful jailbreaks. Fewer query calls indicate a higher attack efficiency.

Defenses. We select three defense methods to test the performance of MIST against various safeguards: PPL-filter [21], Self-Reminder [38], and Backtranslation [22]. All these defenses are set up in accordance with the original papers.

4.2 Evaluation of attack performance and query efficiency

Table 2: The ASR (ASR-Dict / ASR-G) and average success queries (Avg. Q) of different attack methods on four closed-source models and two open-source models. The **bold** and underlined values are the best and runner-up results respectively. Since GCG and AutoDAN requires white-box access to generate prompts, we are unable to calculate their average queries on closed-source models.

Attack method	Metrics	Closed-source model				Open-source model	
		GPT-4-turbo	GPT-4o	GPT-4o-mini	Claude-3.5-sonnet	Vicuna	Llama-2
GCG [6]	ASR↑	10% / 2%	2% / 0%	6% / 4%	4% / 0%	88% / 86%	18% / 6%
	Avg. Q↓	–	–	–	–	256K	256K
AutoDAN [7]	ASR↑	22% / 14%	6% / 0%	10% / 4%	14% / 4%	80% / 64%	16% / 30%
	Avg. Q↓	–	–	–	–	**1.9**	**21.8**
PAIR [17]	ASR↑	32% / 30%	30% / 26%	14% / 14%	10% / 8%	74% / 70%	10% / 10%
	Avg. Q↓	28.1	35.6	40.5	56.8	18.4	86.5
TAP [19]	ASR↑	36% / 32%	40% / 34%	36% / 24%	20% / 16%	78% / 78%	30% / 24%
	Avg. Q↓	30.5	**31.7**	27.4	126.3	15.2	70.4
MIST-SSS	ASR↑	82% / 68%	76% / 56%	80% / 72%	54% / 38%	**94%** / 80%	36% / 20%
	Avg. Q↓	112.9	125.7	139.3	167.1	40.3	189.4
MIST-ODO	ASR↑	**86% / 80%**	**76% / 70%**	**84% / 80%**	**66% / 62%**	88% / **90%**	**48% / 50%**
	Avg. Q↓	**24.8**	32.9	**25.5**	**40.2**	18.9	32.6

Table 3: The jailbreak transferability of MIST-ODO and baselines. We demonstrate the ASR-Dict of prompts that successfully jailbreak a source LLM when they are transferred to a downstream LLM. We omit results when transferring to the original target model. The **bold** and underlined values respectively show the best results achieved by the jailbreak prompts of GPT-4-turbo and Vicuna.

Attack method	Original target model	Transfer target model					
		GPT-4-turbo	GPT-4o	GPT-4o-mini	Claude-3.5-sonnet	Vicuna	Llama-2
GCG [6]	Vicuna	6.82%	0.00%	4.55%	0.00%	–	0.00%
AutoDAN [7]	Vicuna	17.50%	5.00%	12.50%	2.50%	–	2.50%
PAIR [17]	GPT-4-turbo	–	18.75%	15.63%	6.25%	62.50%	0.00%
	Vicuna	16.22%	5.41%	2.71%	0.00%	–	0.00%
TAP [19]	GPT-4-turbo	–	16.67%	13.51%	**8.11%**	54.05%	**5.41%**
	Vicuna	20.51%	10.26%	10.26%	2.56%	–	0.00%
MIST-ODO	GPT-4-turbo	–	**44.19%**	**20.93%**	4.65%	**76.74%**	2.33%
	Vicuna	22.73%	15.91%	6.82%	2.27%	–	4.55%

As shown in Table 2 and Table 3, we report the ASR, average success query count, and transferability among MIST and other baselines. Based on the results, we summarize the following observations:

(i) Attack Success Rate. Compared to previous baselines, MIST achieves the best or second best ASRs (including ASR-Dict and ASR-G) across all the closed-source and open-source LLMs. This indicates that the semantic tuning strategy is effective on probing and crossing the safety boundary of LLMs. Besides, it is worth mentioning that the difference between ASR-Dict and ASR-G when using MIST-SSS is relatively large (a high false-positive rate). That is because the prompts which MIST-SSS generates often fail to reach an optimal similarity with the original prompts, leading to a deviation in meaning, which triggers off-topic or benign responses. The issue is better resolved in MIST-ODO.

(ii) Efficiency. On most closed-source LLMs, compared to PAIR and TAP, MIST-ODO requires the least query calls to achieve a successful jailbreak. On the open-source models, MIST-ODO is not as efficient as the white-box attack AutoDAN, but still attains a low query count among other baselines.

(iii) Transferability. For white-box attacks, we follow [6], using the successful jailbreaks found at the final optimization step when attacking Vicuna. For black-box attacks, we use the successful jailbreaks found for GPT-4-turbo and Vicuna. As shown in Table 3, the GPT-4-turbo jailbreak prompts of MIST-ODO reflect a better transferability than those generated by baselines on most models. The same goes for Vicuna jailbreak prompts, reporting a strong transfer capability of MIST-ODO.

4.3 Performance against defense

Table 4: The performance of MIST-ODO against different defense methods (ASR-Dict). The **bold** values demonstrate the most effective safeguard performance.

Target model	Defense method			
	MIST-ODO (w/o safeguards)	+PPL filter [21]	+Self-reminder [38]	+Backtranslation [22]
GPT-4-turbo	86%	70%	42%	**26%**
GPT-4o	76%	58%	50%	**28%**
GPT-4o-mini	84%	72%	52%	**30%**
Claude-3.5-sonnet	66%	48%	20%	**14%**
Vicuna	88%	76%	46%	**22%**
Llama-2	48%	30%	10%	**6%**

As demonstrated in Table 4, most jailbreak prompts generated by MIST could pass the PPL filter, which indicates that they are fairly fluent and coherent. When it comes to Backtranslation, the ASR-Dict reflects a significant decrease, because we observe that the backtranslation process always paraphrases the original tokens back, unveiling the initial request (not masked by the uncommon synonyms) whose intention is easier to recognize.

4.4 Ablation study

Table 5: Ablation study for MIST-ODO. We respectively ablate the two key steps — original token recovery and optimization order computation, to evaluate their impact on attack success rate (ASR-Dict / ASR-G).

Target model	MIST setting		
	MIST-ODO	w/o Original token recovery	w/o Optimization order computation
GPT-4-turbo	86% / 80%	74% / 50%	42% / 38%
GPT-4o	76% / 70%	60% / 32%	30% / 28%
GPT-4o-mini	84% / 80%	68% / 36%	30% / 26%
Claude-3.5-sonnet	66% / 62%	52% / 22%	14% / 8%
Vicuna	88% / 90%	74% / 48%	50% / 40%
Llama-2	48% / 50%	26% / 6%	10% / 2%

As reported in Table 5, we observe that both the original token recovery step and optimization order computation step are indispensable in contributing to MIST-ODO's outstanding attack performance. Without the former procedure, the difference between ASR-Dict and ASR-G, or false positive rate, sees a notable increase. Without the latter one, the overall ASR reflects a significant drop.

We present our analysis of the ablation study. On the one hand, when we remove the original token recovery process, the semantic meaning of the tuned prompt tends to largely deviate from the original one, leading to a great portion of off-topic responses. On the other hand, when we omit the optimization order computation step, most prompts are not tuned sufficiently, or gone through enough variations to cross the safety boundary of LLMs.

5 Conclusion

In this paper, we introduce MIST, an effective method which jailbreaks black-box large language Models via Iterative Semantic Tuning. MIST performs jailbreak attacks by subtly refining prompts that preserve the original semantic intent while inducing harmful content. Extensive experiments reveal that MIST achieves competitive attack success rates, great efficiency, strong attack transferability, and solid robustness against defenses.

We hope our work could encourage the LLM safety community to further explore advanced jailbreak attacks, as a thorough understanding of adversarial capabilities is key to enhancing model robustness.

References

[1] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A

survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

[2] Md Rafi Ur Rashid, Jing Liu, Toshiaki Koike-Akino, Ye Wang, and Shagufta Mehnaz. Forget to flourish: Leveraging machine-unlearning on pretrained language models for privacy leakage. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence*, pages 20139–20147, 2025.

[3] Tinh Son Luong, Thanh-Thien Le, Linh Ngo Van, and Thien Huu Nguyen. Realistic evaluation of toxicity in large language models. In *Findings of the Association for Computational Linguistics*, pages 1038–1047, 2024.

[4] Tianle Gu, Zeyang Zhou, Kexin Huang, Dandan Liang, Yixu Wang, Haiquan Zhao, Yuanqi Yao, Xingge Qiao, Keqing Wang, Yujiu Yang, Yan Teng, Yu Qiao, and Yingchun Wang. MLLMGuard: A multi-dimensional safety evaluation suite for multimodal large language models. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, pages 1–40, 2024.

[5] Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. A comprehensive study of jailbreak attack versus defense for large language models. In *Findings of the Association for Computational Linguistics*, pages 7432–7449, 2024.

[6] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. In *arXiv*, pages 1–31, 2023.

[7] Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *Proceedings of the 12th International Conference on Learning Representations*, pages 1–21, 2024.

[8] Xingang Guo, Fangxu Yu, Huan Zhang, Lianhui Qin, and Bin Hu. COLD-Attack: Jailbreaking LLMs with stealthiness and controllability. In *Proceedings of the 41st International Conference on Machine Learning*, pages 1–29, 2024.

[9] Kai Hu, Weichen Yu, Yining Li, Tianjun Yao, Xiang Li, Wenhe Liu, Lijun Yu, Zhiqiang Shen, Kai Chen, and Matt Fredrikson. Efficient LLM jailbreak via adaptive dense-to-sparse constrained optimization. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, pages 1–22, 2024.

[10] Xiaojun Jia, Tianyu Pang, Chao Du, Yihao Huang, Jindong Gu, Yang Liu, Xiaochun Cao, and Min Lin. Improved techniques for optimization-based jailbreaking on large language models. In *Proceedings of the 13th International Conference on Learning Representations*, pages 1–15, 2025.

[11] Hangfan Zhang, Zhimeng Guo, Huaisheng Zhu, Bochuan Cao, Lu Lin, Jinyuan Jia, Jinghui Chen, and Dinghao Wu. Jailbreak open-source large language models via enforced decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 5475–5493, 2024.

[12] Yizhuo Ma, Shanmin Pang, Qi Guo, Tianyu Wei, and Qing Guo. ColJailBreak: Collaborative generation and editing for jailbreaking text-to-image deep generation. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, pages 1–24, 2024.

[13] Xiangyu Qi, Kaixuan Huang, Ashwinee Panda, Peter Henderson, Mengdi Wang, and Prateek Mittal. Visual adversarial examples jailbreak aligned large language models. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, pages 21527–21536, 2024.

[14] Fengqing Jiang, Zhangchen Xu, Luyao Niu, Zhen Xiang, Bhaskar Ramasubramanian, Bo Li, and Radha Poovendran. ArtPrompt: ASCII art-based jailbreak attacks against aligned LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 15157–15173, 2024.

[15] Fabio Pernisi, Dirk Hovy, and Paul Röttger. Compromesso! Italian many-shot jailbreaks undermine the safety of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 263–269, 2024.

[16] Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. A wolf in sheep's clothing: Generalized nested jailbreak prompts can fool large language models easily. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2136–2153, 2024.

[17] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *arXiv*, pages 1–34, 2024.

[18] Xuan Chen, Yuzhou Nie, Wenbo Guo, and Xiangyu Zhang. When LLM meets DRL: Advancing jailbreaking efficiency via DRL-guided search. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, pages 1–32, 2024.

[19] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box LLMs automatically. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, pages 1–41, 2024.

[20] Hongyi Li, Jiawei Ye, Jie Wu, Tianjie Yan, Chu Wang, and Zhixin Li. JailPO: A novel black-box jailbreak framework via preference optimization against aligned LLMs. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence*, pages 27419–27427, 2025.

[21] 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. In *arXiv*, pages 1–19, 2023.

[22] Yihan Wang, Zhouxing Shi, Andrew Bai, and Cho-Jui Hsieh. Defending LLMs against jailbreaking attacks via backtranslation. In *Findings of the Association for Computational Linguistics*, pages 16031–16046, 2024.

[23] Ziyang Zhang, Qizhen Zhang, and Jakob Foerster. PARDEN, Can you repeat that? Defending against jailbreaks via repetition. In *Proceedings of the 41st International Conference on Machine Learning*, pages 1–17, 2024.

[24] Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. Gradient Cuff: Detecting jailbreak attacks on large language models by exploring refusal loss landscapes. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, pages 1–32, 2024.

[25] Yueqi Xie, Minghong Fang, Renjie Pi, and Neil Zhenqiang Gong. GradSafe: Detecting jailbreak prompts for LLMs via safety-critical gradient analysis. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 507–518, 2024.

[26] Zhexin Zhang, Junxiao Yang, Pei Ke, Fei Mi, Hongning Wang, and Minlie Huang. Defending large language models against jailbreaking attacks through goal prioritization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 8865–8887, 2024.

[27] Yichuan Mo, Yuji Wang, Zeming Wei, and Yisen Wang. Fight back against jailbreaking via prompt adversarial tuning. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, pages 1–31, 2024.

[28] Jiongxiao Wang, Jiazhao Li, Yiquan Li, Xiangyu Qi, Junjie Hu, Yixuan Li, Patrick McDaniel, Muhao Chen, Bo Li, and Chaowei Xiao. BackdoorAlign: Mitigating fine-tuning based jailbreak attack with backdoor enhanced safety alignment. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, pages 1–34, 2024.

[29] Andy Zhou, Bo Li, and Haohan Wang. Robust prompt optimization for defending language models against jailbreaking attacks. In *Proceedings of the 38th Conference on Neural Information Processing Systems* , pages 1–28, 2024.

[30] Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. SafeDecoding: Defending against jailbreak attacks via safety-aware decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 5587–5605, 2024.

[31] Han Liu, Zhi Xu, Xiaotong Zhang, Feng Zhang, Fenglong Ma, Hongyang Chen, Hong Yu, and Xianchao Zhang. HQA-attack: Toward high quality black-box hard-label adversarial attack on text. In *Proceedings of the 37th Conference on Neural Information Processing Systems*, pages 1–12, 2023.

[32] Xiaoxue Hu, Geling Liu, Baolin Zheng, Lingchen Zhao, Qian Wang, Yufei Zhang, and Minxin Du. FastTextDodger: Decision-based adversarial attack against black-box NLP models with extremely high efficiency. *IEEE Transactions on Information Forensics and Security*, 19:2398–2411, 2024.

[33] Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramer, Hamed Hassani, and Eric Wong. JailbreakBench: An open robustness benchmark for jailbreaking large language models. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, pages 1–25, 2024.

[34] 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, 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

[35] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. In *arXiv*, pages 1–77, 2023.

[36] Anthropic. Claude-3.5-Sonnet, 2024. URL https://www.anthropic.com/news/claude-3-5-sonnet.

[37] OpenAI. GPT-4 Technical Report. In *arXiv*, pages 1–100, 2023.

[38] 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*, 5:1486–1496, 2023.