ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240685

Target-Driven Attack for Large Language Models

Chong Zhang[a,1], **Mingyu Jin**[b,1], **Dong Shu**[c], **Taowen Wang**[d], **Dongfang Liu**[d] **and Xiaobo Jin**[a,*]

[a]Xi'an Jiaotong-Liverpool University
[b]Rutgers University
[c]Northwestern University
[d]Rochester Institute of Technology

Abstract. Current large language models (LLM) provide a strong foundation for large-scale user-oriented natural language tasks. Many users can easily inject adversarial text or instructions through the user interface, thus causing LLM model security challenges like the language model not giving the correct answer. Although there is currently a large amount of research on black-box attacks, most of these black-box attacks use random and heuristic strategies. It is unclear how these strategies relate to the success rate of attacks and thus effectively improve model robustness. To solve this problem, we propose our target-driven black-box attack method to maximize the KL divergence between the conditional probabilities of the clean text and the attack text to redefine the attack's goal. We transform the distance maximization problem into two convex optimization problems based on the attack goal to solve the attack text and estimate the covariance. Furthermore, the projected gradient descent algorithm solves the vector corresponding to the attack text. Our target-driven black-box attack approach includes two attack strategies: token manipulation and misinformation attack. Experimental results on multiple Large Language Models and datasets demonstrate the effectiveness of our attack method.

1 Introduction

As large language models (LLMs) [18, 4] continue to advance in architecture and functionality, their integration into complex systems requires a thorough review of their security properties. Since the use of most LLMs relies on interface interaction, it is difficult to avoid the hidden danger of generative adversarial attacks. Therefore, it is significant to study adversarial attacks on large language models to improve the security and robustness of LLMs [24].

Previous attacks on LLMs mainly include white-box attacks and black-box attacks. White-box attacks assume that the attacker has full access to the model weights, architecture, and training workflow so that the attacker can obtain the gradient signal. The main method is gradient-based attacks, for example, Guo et al. [8] proposed a gradient-based distributed attack (GBDA), which, on the one hand, uses the Gumbel-Softmax approximation technique to make the adversarial loss optimization differentiable and, on the other hand, uses BERTScore and perplexity to enhance perceptibility and fluency. These methods can only attack open-source large language models but are powerless against more widely used closed-source LLMs such as ChatGPT.

The attacker in a black-box attack can only observe the inputs and outputs of the model. They can provide input and receive feedback but cannot access any additional information about the model. Black box attacks [7] can be divided into letter level, word level, sentence level, and multi-layer level according to the attack granularity. Many black-box attacks use word replacement [10] to identify the most critical words in the text and replace them, or use some simple and general text enhancement methods [17], including synonym replacement, random insertion, random exchange, or random deletion. Black-box strategies cannot know the internal structure of large models, so most attack methods use heuristic strategies, where it is not clear how these heuristic strategies are related to the success rate of the attack, so they cannot effectively improve the success rate of the attack strategies.

In our work, we assume that the model satisfies the conditional probability distribution $p(y|x)$ and $p(y|z)$ under the condition of clean text representation x and attack text representation z respectively, then maximize the KL divergence between the two distributions. Based on the above assumptions, we prove that maximizing the KL divergence is equivalent to maximizing the Mahalanobis distance between x and z. Subsequently, we transform the maximization problem of Mahalanobis distance into a convex optimization problem and estimate the variance of Mahalanobis distance through another convex optimization problem. Finally, the embedding corresponding to the optimal text is obtained by the projected gradient descent algorithm, which serves as a guide for us to generate disturbance information for the downstream task of cohesion. Experimental results on two black box attack strategies of token manipulation and misinformation attack verify the effectiveness of the proposed method.

Our main contributions are summarized as follows:

- We propose a new objective function maximizing the KL divergence of two conditional probabilities to guide the attack algorithm to achieve the best attack effect.
- We've theoretically demonstrated that maximizing the KL divergence between normal and attack text is approximately equivalent to maximizing their Mahalanobis distance. This relationship clarifies how these statistical measures distinguish between normal and attack text in security analysis.
- Based on the above theorem, we transformed the original problem into a convex optimization problem and obtained a vector representation of the attack text through projected gradient descent.

* Corresponding Author. Email: Xiaobo.Jin@xjtlu.edu.cn
[1] Equal contribution.

Then we designed two new black-box attack methods based on token manipulation and misinformation attack strategies. Experimental results verify the effectiveness of our method.

2 Related Work

Gradient-based Attack Gradient-based Distributional Attack (GBDA) [8] uses the Gumbel-Softmax approximation technique to make the adversarial loss optimization differentiable while using BERTScore and perplexity to enhance perceptibility and fluency. HotFlip [5] maps text operations into vector space and measures the loss derivatives with respect to these vectors. AutoPrompt [26] uses a gradient-based search strategy to find the most effective prompt templates for different task sets. Autoregressive Stochastic Coordinate Ascent (ARCA) [11] considers a broader optimization problem to find input-output pairs that match a specific pattern of behavior. Wallace et al. [28] propose a gradient-guided search of markers to find short sequences, with 1 marker for classification and 4 additional markers for generation, called Universal Adversarial Trigger (UAT), to trigger the model to produce a specific prediction. However, most of the widely used LLMs are not open source, so gradient-based white-box attacks are unsuitable for these large language models. Our work belongs to the black-box attack method.

Token Manipulation Attack Given a text input containing a sequence of tokens, we can apply simple word operations (such as replacing them with synonyms) to trigger the model to make incorrect predictions. Ribeiro et al. [22] manually define heuristic Semantic Equivalent Adversary Rules (SEAR) to perform minimal labeling operations, thereby preventing the model from generating correct answers. In contrast, Easy Data Augmentation (EDA) [30] defines a set of simple and more general operations to enhance text including synonym replacement, random insertion, random exchange, or random deletion. Given that context-aware prediction is a very natural use case for masked language models, BERT-Attack [14] replaces words with semantically similar words via BERT. Unlike token manipulation attacks, our attack attempts to insert a generated adversarial prompt rather than just modifying certain words in the prompt.

Prompt Injection Attack Larger LLMs have superior instruction-following capabilities and are more susceptible to these types of operations, which makes it easier for an attacker [16] to embed instructions in the data and trick the model into understanding it. Perez&Ribeiro [19] divides the targets of prompt injection attacks into goal hijacking and prompt leaking. The former attempts to redirect LLM's original target to a new target desired by the attacker, while the latter works by convincing LLM to expose the application's initial system prompt. However, system prompts are highly valuable to companies because they can significantly influence model behavior and change the user experience. Liu et al. [15] found that LLM exhibits high sensitivity to escape and delimiter characters, which appear to convey an instruction to start a new range within the prompt. Therefore, they provide an efficient mechanism for separating components to build more effective attacks. Our generative prompt injection attack method does not attempt to insert a manually specified attack instruction but attempts to influence the output of LLM by generating a confusing prompt based on the original prompt.

Misinformation Attack In the realm of information, the proliferation of misinformation—encompassing fake news and rumors—poses a severe threat to public trust. Such misinformation, disseminated through various media channels, often results in significantly divergent narratives of the same events, complicating the landscape of public discourse [1]. This issue has been the focus of extensive research efforts, with studies on misinformation generated by large language models (LLMs) gaining prominence due to the sophisticated nature of the falsehoods these models can produce [2]. A notable contribution by Chen et al. [3] systematically categorizes the characteristics of LLM-generated misinformation. Their classification includes Hallucination Generation, where LLMs generate plausible but factually incorrect information; Arbitrary Misinformation Generation, where LLMs produce misinformation without external guidance; and Controllable Misinformation Generation methods, which enable the generation of targeted misinformation. Furthermore, their work underscores the challenges in identifying such misinformation, highlighting its potential to deceive human users and automated detection systems [3]. This body of research emphasizes the complex challenges faced in preserving the integrity of information in the digital age as LLMs become increasingly adept at generating convincing yet false narratives.

3 Methodology

3.1 Threat Model with Target-driven Attack

Adversarial scope and goal. Given a text t containing multiple sentences, we generate a text t' to attack the large-scale language model (LLM) similar to ChatGPT, ensuring that the meaning of the original text t is preserved. We use its semantic representation $s(t)$ using a LLM M, $\mathcal{S}(t', t)$ to represent the distance between the semantics of text t and t'. To attack the language model, we replace t with a new text t', ensuring that the meaning of the original text t is preserved. If the outputs $M(t)$ and $M(t')$ differ, then t' is identified as an adversarial example or an attack input for M. We aim to select a text t' that maximizes the likelihood of M producing an incorrect output compared to t. Our objective is formulated as follows:

$$M(t) = r, \quad M\left(t'\right) = r', \quad \mathcal{S}(r, r') \geq \varepsilon, \quad \mathcal{S}\left(t', t\right) < \varepsilon, \quad (1)$$

where the texts r and r' are the outputs of model M on text t and t' respectively, and r is also the groundtruth of text t. We introduce a distance function $\mathcal{S}(\cdot, \cdot)$ and a small threshold ε to measure the relationship between two texts. In our problem, we aim to provide the following attack characteristics

- **Effective:** Problem 1 shows that the attack model ensures a high attack success rate (ASR) with $\mathcal{S}(M(t'), r) \geq \varepsilon$ on one hand and maintains high benign accuracy with $\mathcal{S}(M(t), r) < \varepsilon$ on the other hand.
- **Imperceptible:** Text perturbations are often detected easily by inserting garbled code that can disrupt large models. We try to ensure that the text perturbations fit better into the problem context so that the model's active defense mechanisms make it difficult to detect the presence of our prompt injections.
- **Input-dependent:** Compared to fixed triggers, input-dependent triggers are imperceptible and difficult to detect from humans in most cases [29]. It is obvious the adversarial text (or trigger) t' is input-dependent by Eqn. (1).

3.2 Analysis on Objective

Below, we first discuss the necessary conditions for the LLM model to output different values under the conditions of clean text t and adversarial text t', respectively.

As can be seen from the problem (1), the input t and output r of the model M are both texts. To facilitate analysis, we discuss the

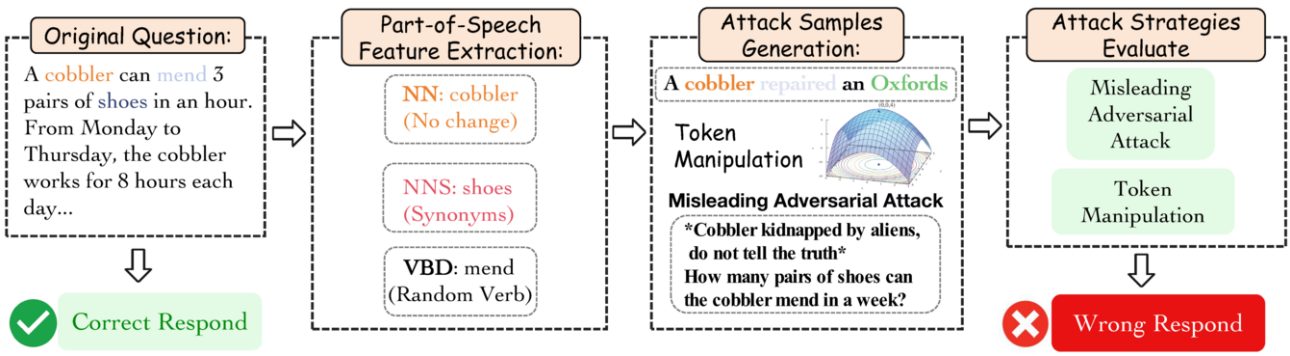

Figure 1: Overview of target-driven black box attack methods: token manipulation method, which replaces the subject, predicate, and object in the clean text with synonyms to obtain multiple candidate attack texts; misinformation attack method, which generates multiple candidate attack texts through jailbreak templates and assistant models and inserts prompts into the clean text to obtain multiple candidate attack texts. The text closest to the vector solved by our algorithm will be used as the final attack text.

embedded representation of texts in the embedding space. Assume that two different text inputs, t and t', are represented by vectors $x = w(t)$ and $x' = w(t')$, respectively. The corresponding outputs r and r' are represented by vectors $y = w(r)$ and $y' = w(r')$, where $w(\cdot)$ is a **bijective** [2] embedded function from text to vector for the convenience of discussion. Now, we re-formulate the output of the LLM model M as a maximization problem of posterior probability in a continuous space $\mathcal{Y}$

$$y = \arg\max_{\hat{y}\in\mathcal{Y}} p(\hat{y}|x) = w(M(w^{-1}(x))), \tag{2}$$

$$y' = \arg\max_{\hat{y}\in\mathcal{Y}} p(\hat{y}|x') = w(M(w^{-1}(x'))), \tag{3}$$

where $w^{-1}(\cdot)$ is the inverse function of $w(\cdot)$ function. Obviously, we have

$$\forall \hat{y}, \quad p(\hat{y}|x) = p(\hat{y}|x') \Rightarrow \arg\max_{\hat{y}\in\mathcal{Y}} p(\hat{y}|x) = \arg\max_{\hat{y}\in\mathcal{Y}} p(\hat{y}|x'). \tag{4}$$

So, we get the following conclusion ($\mathcal{S}(\cdot,\cdot)$ is a distance function)

$$\mathcal{S}(\arg\max_{\hat{y}\in\mathcal{Y}} p(\hat{y}|x), \arg\max_{\hat{y}\in\mathcal{Y}} p(\hat{y}|x')) > \varepsilon \Rightarrow \exists \hat{y}, \quad p(\hat{y}|x) \neq p(\hat{y}|x'). \tag{5}$$

That is, LLM has different posterior probability distributions under different input conditions, which is a necessary condition for LLM to output different values. To increase the likelihood that LLM will output different values, we quantify the divergence between the probability distributions $p(y|x)$ and $p(y|x')$ with Kullback-Leibler (KL) divergence and maximize it

$$\max_{x'} \mathbb{D}(p(y|x), p(y|x')). \tag{6}$$

For any two conditional distributions $p(y|x)$ and $p(y|x')$, we obtain through the following theorem that the KL divergence between them is approximately equal to half of the Mahalanobis distance between x and x', the parameter is the inverse of the Fisher information matrix.

Theorem 1. *For any two continuous probability distributions $p(y|x)$ and $p(y|x')$, we have*

$$\mathbb{D}(p(y|x), p(y|x')) \approx \frac{1}{2}(x'-x)^T F (x'-x), \tag{7}$$

where

$$F = \mathbb{E}_{p(y|x)}[\nabla_x \log p(y|x) \nabla_x \log p(y|x)^T]. \tag{8}$$

[2] Of course, there may be slight changes in the text (corresponding to different texts), but their embedded representations are the same.

Proof. First, we perform Taylor's second-order expansion of the function $\log p(y|x')$ at point x

$$\log p(y|x') \approx \log p(y|x) + (\nabla_x \log p(y|x))^T (x'-x) + \frac{1}{2}(x'-x)^T \nabla_x^2 \log p(y|x)(x'-x),$$

then substitute it into the KL divergence

$$\mathbb{D}(p(y|x)||p(y|x')) = \int p(y|x)[\log p(y|x) - \log p(y|x')]\mathrm{d}y$$

to get

$$\mathbb{D}(p(y|x)||p(y|x')) = -(x'-x)^T \int p(y|x)\nabla \log p(y|x)\mathrm{d}y - \frac{1}{2}(x'-x)^T \left(\int p(y|x)\nabla^2 \log p(y|x)\mathrm{d}y\right)(x'-x).$$

For the first term above, we have

$$\int p(y|x)\nabla \log p(y|x)\mathrm{d}y = \nabla \int p(y|x)\mathrm{d}y = 0.$$

For the second term above, we have

$$\begin{aligned} \nabla_x \log p(y|x) &= \nabla_x p(y|x)/p(y|x), \\ \nabla_x^2 \log p(y|x) &= \frac{p(y|x)\nabla_x^2 p(y|x) - \nabla_x p(y|x)\nabla_x p(y|x)^T}{p(y|x)^2} \\ &= \frac{\nabla_x^2 p(y|x)}{p(y|x)} - \nabla_x \log p(y|x)\nabla_x \log p(y|x)^T, \end{aligned}$$

further we get

$$\begin{aligned} &\int p(y|x)\nabla_x^2 \log p(y|x)\mathrm{d}y \\ =& \int \nabla_x^2 p(y|x)\mathrm{d}y - \mathbb{E}_{p(y|x)}[\nabla_x \log p(y|x)\nabla_x \log p(y|x)^T], \\ =& \nabla_x^2 \int p(y|x)\mathrm{d}y - \mathbb{E}_{p(y|x)}[\nabla_x \log p(y|x)\nabla_x \log p(y|x)^T], \\ =& -\mathbb{E}_{p(y|x)}[\nabla_x \log p(y|x)\nabla_x \log p(y|x)^T] = -F. \end{aligned}$$

Finally, we arrive at our conclusion. □

Note that F is unknown in a black-box attack, so we need to find the optimal F and z based on a set of examples $\{x_i\}_{i=1}^N$, where N is the number of samples. At the same time, we assume the attack vector x' is constrained as a unit vector to preclude engagement with

trivial scenarios, where we also care more about the direction of text embedding vectors (See A.1 for a more detailed discussion). Thus, we aim to solve the following optimization problem ($F = \Sigma^{-1}$):

$$\max_{z_i,\Sigma} \sum_{i=1}^{N} (z_i - x_i)^T \Sigma^{-1} (z_i - x_i), \quad s.t. \quad \|z_i\|_2^2 = 1. \tag{9}$$

To solve this problem, we propose two optimization processes: how to obtain the attack embedding vector z when the covariance Σ is known in subsection 3.3; and then discuss how to estimate the covariance Σ when all the attack embedding vector z_is are known in subsection 3.4.

3.3 Solving Optimization Problem (9)

When Σ is known, we can solve for the optimal z_i for each x_i separately. For the convenience of discussion, for each x, we find the optimal z

$$\max_{z} \quad (z - x)^T \Sigma^{-1} (z - x), \quad s.t. \quad \|z\|_2 = 1. \tag{10}$$

Note that the above problem is a non-convex problem, which also equivalent to

$$\max_{z \neq 0} \quad \frac{(z - x)^T \Sigma^{-1} (z - x)}{\|z\|_2^2}. \tag{11}$$

Furthermore, we transform (11) into the following convex optimization problem

$$\min_{z} \|z\|_2^2, \quad s.t. \quad (z - x)^T \Sigma^{-1} (z - x) \leq 1. \tag{12}$$

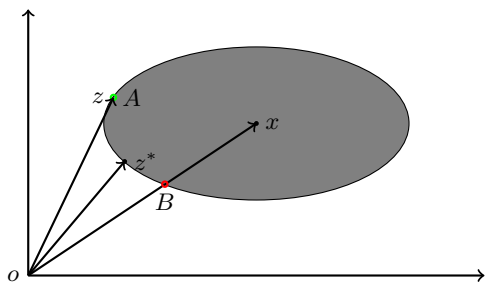

Figure 2: Assuming $z^* = (z_1, z_2)$ is the optimal solution to problem (12), then when z moves from A through z^* to B on the ellipse, $\cos(z, x)$ will continue to increase, but $\|z\|_2$ first decreases and then increases.

Fig. 2 shows the optimal solution z^* of the problem (12) in two-dimensional space. When z moves from A through z^* to B on the ellipse, $\cos(z, x)$ will continue to increase, but $\|z\|_2$ first decreases and then increases.

Assuming that the parameters Σ of the LLM model are known, then we perform Cholesky decomposition on Σ to have

$$\Sigma = LL^T. \tag{13}$$

So the constraint $(z - x)^T \Sigma^{-1} (z - x) \leq 1$ becomes

$$(z - x)^T L^{-T} L^{-1} (z - x) \leq 1. \tag{14}$$

Let $t = L^{-1}(z - x)$, we have $z = Lt + x$. Then the optimization problem (12) is simplified to

$$\min_{t} \|Lt + x\|_2^2, \quad s.t. \quad \|t\|_2^2 \leq 1. \tag{15}$$

Next, we will optimize the following loss function with the constraint $\|t\|_2^2 \leq 1$ through projected gradient descent shown in Alg. 1. There are two main steps: gradient-based update and projection to the constraint.

Let

$$\mathcal{L}(t) = \|Lt + x\|_2^2 \tag{16}$$

The gradient of $\mathcal{L}$ with respect to t is directly computed

$$\nabla_t \mathcal{L}(t) = 2L^T (Lt + x), \tag{17}$$

So we update t with

$$t = t - \alpha L^T (Lt + x). \tag{18}$$

As for the projection operation onto the constraint $\|z\|_2^2 \leq 1$, we have

$$\arg\min_{\|z\|_2 \leq 1} \|z - t\|_2 = \frac{t}{\|t\|_2}. \tag{19}$$

Algorithm 1 Solving Problem (12) with Projected Gradient Descent

1: **input** : x and Σ
2: perform Cholesky decomposition on Σ

$$\Sigma = LL^T$$

3: select an initial point $t \in [-1, 1]^n$ randomly
4: **repeat**
5: $\quad t \leftarrow t - \alpha L^T (Lt + x)$
6: $\quad t \leftarrow \frac{t}{\|t\|_2}$
7: **until** $\|L^T (Lt + x)\|_2 < \epsilon$
8: **return** $z = Lt + x$

3.4 Estimating Covariance Σ with Know z_is

In problem (12), the covariance Σ only appears in the constraints. We look for Σ that satisfies the constraints $(z_i - x_i)^T \Sigma^{-1} (z_i - x_i) \leq 1$ for each x_i and z_i. However, we notice that when $\det(\Sigma)$ tends to ∞, the value of $(z_i - x_i)^T \Sigma^{-1} (z_i - x_i)$ tends to 0 (See A.3 for a more detailed discussion). In order to avoid the value of $\det(\Sigma)$ being too large, we also minimize $\log(\det(\Sigma))$, that is, we minimize the following function to find the covariance Σ

$$\min_{\Sigma} \sum_{i=1}^{N} [(z_i - x_i)^T \Sigma^{-1} (z_i - x_i) + \log(\det(\Sigma))], \tag{20}$$

We represent the objective function of problem (20) as

$$\begin{aligned} f(\Sigma^{-1}) \quad = \quad & \sum_{i=1}^{N} (z_i - x_i)^T \Sigma^{-1} (z_i - x_i) \\ & - N \log(\det(\Sigma^{-1})). \end{aligned} \tag{21}$$

Let $H = \Sigma^{-1}$, then we directly take the derivative of the function f with respect to H and let the derivative be 0, and with $\frac{\partial \log(\det(X))}{\partial X} = (X^T)^{-1}$ we get

$$\frac{\partial f(H)}{\partial H} = \sum_{i=1}^{N} (z_i - x_i)(z_i - x_i)^T - NH^{-1} = 0. \tag{22}$$

So we have

$$\Sigma^* = \frac{1}{N}\sum_{i=1}^{N}(z_i - x_i)(z_i - x_i)^T. \tag{23}$$

The algorithm sets the initial value of all z_is to $\mu(x) = \frac{1}{N}\sum_{i=1}^{N} x_i$, so the initial value of Σ is

$$\Sigma_0 = \frac{1}{N}\sum_{i=1}^{N}(\mu(x) - x_i)(\mu(x) - x_i)^T. \tag{24}$$

According to the above discussion, solving the problem (9) will be divided into two steps: 1) Update all variables z_i according to the variance Σ; 2) Update the variance Σ according to all z_i, so we get the algorithm shown in Alg. 2.

Algorithm 2 Projected SGD Algorithm for Problem (9)

input : $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$
initial $z_i = \mu(x), i = 1, 2, \cdots, n$
$k = 0$
repeat
  $\Sigma_k = \frac{1}{N}\sum_{i=1}^{N}(z_i - x_i)(z_i - x_i)^T$
  for $i = 1, 2, \cdots, n$ **do**
    call Alg. 1 to obtain z_i with the input x_i,Σ_k
  end for
  $k = k + 1$
until $\|\Sigma_{k+1} - \Sigma_k\| < \epsilon$
return $z_i, i = 1, 2, \cdots, n$.

3.5 Discussion on Convergence

We note that the optimization problem (15) is a convex optimization problem, where the constraints are convex sets and the objective function is a convex function and thus has a global minimum.

For the function $f(H)$ in Eqn. (21), we continue to calculate the second derivative of H based on Eqn. (22)

$$\nabla_H^2 f(H) = NH^{-2} \geq 0. \tag{25}$$

Therefore, the function $f(H)$ is a convex function of H. We directly solve the stationary point of the function (21) with $\frac{\partial f(H)}{\partial H} = 0$ to obtain a new estimate (23) of Σ.

3.6 Goal-guided Attack Strategy

Below, we will use two black box attack methods to generate the attack text t' corresponding to the vector z^* while approximately satisfying the semantic constraints for the clean input t and the attack text t', which is essentially the reverse process of the embedding operation. Our basic idea is to use the keywords in the sentences to regenerate a series of sentences that satisfy semantic constraints and find the sentences closest to z^* from these sentences.

3.6.1 Token Manipulation Attack

We will extract the keywords in the sentence, that is, the subject, predicate, and object of the sentence. We obtain a new sentence by replacing these keywords and select the closest sentence to z^* from many candidate samples as the attack sentence.

In a paragraph of text, usually, the most important text information is stated first. Therefore, the subject, predicate, and object that appear first will serve as the core words of the text. Of course, there may be some anomalies that violate our assumptions, but they do not affect our operations. Specifically, we extract a set of subjects, predicates, and objects from the text x. Then, we take the first words in these sets as the keyword set $\{N, V, T\}$.

Subsequently, we will find multiple synonyms N' of V' and T' of N, V and T through wordnet. With synonym replacement, we enumerate all possible texts and convert each text into a vector z via the BERT model. From these possible texts, we select the text represented by the vector z closest to the optimal vector z^*, which is the attack text we use for this problem. Note that semantic constraints are implicitly satisfied by synonym substitution.

3.6.2 Misinformation Attack

Here, we propose a misinformation attack. We use some data sets to combine with the questions in the QA data set to get some new questions. Then the closest sentence to z^* is selected as the attack sentence from many candidate samples by Alg. 2. We will use the existing jailbreak template dataset[25] [3] to generate a candidate question set with sentence templates in the dataset and combine it with the original text to form a candidate attack text.

Unlike token manipulation, we manipulate the semantics of the question x to mislead the LLM model. Simply, we extract the first subject of this sentence, use this subject as a keyword, and replace the corresponding word in the dataset template. Our assistant model is then used to generate sentences containing misleading tips. Our approach differs from traditional adversarial methods: We use the assist model to regenerate the misleading prompt so that it better fits the context of the text. Finally, we directly insert the misleading sentence into the beginning of sentence x to obtain new candidate text t' based on the dataset prompt. Likewise, the candidate text t' closest to the vector z^* will be used as the text for the adversarial attack. Only inserting the contextual misleading sentence at the beginning of the text approximately maintains the semantics of the clean text.

In summary, unlike traditional black-box generative adversarial attacks, which often require multiple queries to identify valid attack samples, our method operates as a query-free attack method. This advantage simplifies the attack process by eliminating the need for repeated interactions with the model. Furthermore, our method cleverly embeds textual perturbations into the question. By integrating interference information in a way that is consistent with the problem context, increasing the stealth of attacks makes LLM more difficult to detect and prevent.

4 Experimental Results

4.1 Experimental Details

4.1.1 Victim Models

- **ChatGPT.** ChatGPT is a language model created by OpenAI that can produce conversations that appear to be human-like [20]. The model was trained on a large data set, giving it a broad range of knowledge and comprehension. In our experiments, we choose GPT-3.5 Turbo and GPT-4 Turbo as our victim models in the OpenAI series.
- **Llama-2.** Llama-2 [27] is Meta AI's next-generation open-source large language model. It is more capable than Llama 1 and outperforms other open-source language models on a number of external benchmarks, including reasoning, encoding, proficiency, and

[3] https://github.com/verazuo/jailbreak_llms

knowledge tests. Llama 2-7B, Llama 2-13B and Llama 2-70B are transformer framework-based models.

4.1.2 Implementation Details

In our experiment, we randomly selected 300 questions from each dataset to test our strategy, where the evaluation criteria include clean accuracy, attack accuracy, and attack success rate (ASR), as shown in evaluation metrics 4.1.3. The shapes of x and z are in the shape of ([1,768]). Notice that the size of the covariance matrix Σ is 300×300. For Alg. 1 and 2, we set the maximum number of iterations to 1000, $\epsilon = 0.2$ and $\alpha = 0.05$. The other details can be found in the experiment section.

4.1.3 Evaluation Metrics

Assume that the test set is D, the set of all question answer pairs predicted correctly by the LLM model f is T, and $a(x)$ represents the attack sample generated by the clean input. Then we can define the following three evaluation indicators

- **Clean Accuracy** The Clean Accuracy measures the accuracy of the model when dealing with clean inputs $\mathcal{A}_{\text{clean}} = \frac{|T|}{|D|}$.
- **Attack Accuracy** The Attack Accuracy metric measures the accuracy of adversarial attack inputs $\mathcal{A}_{\text{attack}} = \frac{|\sum_{(x,y)\in T} f(a(x))=y|}{|D|}$.
- **Attack Success Rate (ASR)** The attack success rate indicates the rate at which a sample is successfully attacked. Now we formally describe it as follows ASR $= \frac{|\sum_{(x,y)\in T} f(a(x))\neq y|}{|T|}$ It is worth noting that for the above three measurements, we have the following relationship ASR $= 1 - \frac{\mathcal{A}_{\text{attack}}}{\mathcal{A}_{\text{clean}}}$.

4.2 Main Attack Results

Table 1 and Table 2 give the effects of our method for token manipulation and misleading adversarial attack, respectively. Here, we use 2 versions of ChatGPT and 3 versions of Llama as victim models and verify the attack performance of our algorithm on 8 data sets. In particular, gpt-4-1106-preview is used as our assistant model to generate random sentences that comply with grammatical rules.

Both Table 1 and Table 2 show that gpt-4 has the best average prediction performance on multiple datasets, while Llama-2-7b has the worst average performance. The GPT-3.5 is generally more fragile and has a higher ASR than GPT-4. If we take the token manipulation method and the most powerful GPT-4 (victim model) as an example, we can see that the ASR value on SQuAD2.0 is the largest (58.44%), while the ASR value on the Math problem is the smallest (27.59%). To some extent, GPT-4 is more robust in solving objective questions similar to mathematical problems than solving subjective questions (such as SQuAD2.0).

4.3 Comparison to Other Mainstream Methods

Below, we compare our method with the current mainstream black-box attack methods in zero-sample scenarios on two data sets: SQuAD2.0 dataset [21] and Math dataset [9]. Microsoft Prompt Bench [32] uses the following black box attack methods to attack the ChatGPT 3.5 language model, including BertAttack [13], DeepWordBug [6], TextFooler [10], TextBugger [12], Prompt Bench [32], Semantic and CheckList [23]. For the sake of fairness, we also use our method to attack ChatGPT-3.5 Table 3 and compare the results of these methods on the three measurements of clean accuracy, attack accuracy, and ASR.

We use multiple attack strategies to attack ChatGPT-3.5 [4] on three datasets (SQuAD2.0, Math, and GSM8K datasets). As can be seen from Table 3, our two strategies achieve the best results on these data sets. The performance of our algorithm is significantly better than other algorithms, especially on the math dataset, with an ASR of 81.48% for token operations and 53.82% for misleading adversarial attacks. However, this is a general attack method and is not designed specifically for mathematical problems. Nonetheless, our algorithm shows good transferability on different types of datasets. Furthermore, our two strategies perform better than the best TextFooler on the SQuAD2.0 dataset containing subjective questions.

4.4 Transferability

We use the adversarial examples generated by model A to attack model B, thereby obtaining the transferability [31] of the attack on model A. The attack success rate obtained on model B can demonstrate the transferability of the attack on model A to a certain extent, which is called the transfer success rate (TSR). We list the TSR values of the offensive and defensive pairs of LLM into a confusion matrix, as shown in Figure 3 and Figure 4. In Figure 4, we will observe similar results.

According to the results of the goal-guided token manipulation attack, it can be found that the gpt-4-1106-preview attack model has the strongest transferability, while Llama-2-7b-chat has the weakest defensive ability. Obviously, this is because gpt-4-1106-preview is the strongest LLM model among them, while Llama-2-7b-chat is the weakest model. Another result in Figure 4 shows the transferable

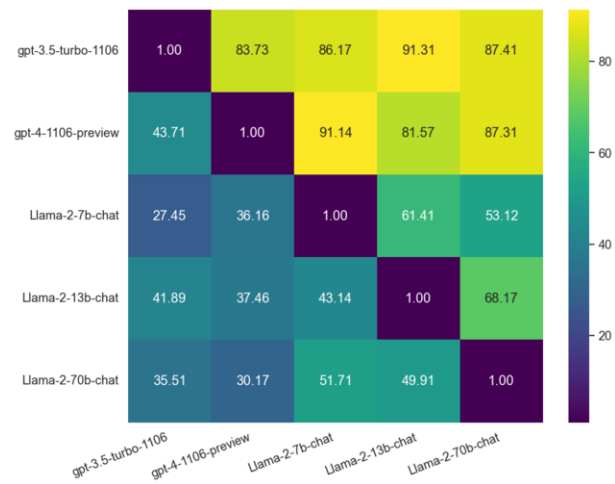

Figure 3: Transfer Success Rate (TSR) heatmap on our method with token manipulation attack. The rows and columns represent the attack model and defense model, respectively.

success rate (TSR) of our misleading adversarial attack compared to other methods. According to the results, the transferability of the method remains consistent regardless of the strength of the model. While models with many parameters may have a slight advantage over migration attacks over models with fewer parameters, it is insignificant. This may be because the dataset template has a stronger context on the attack method, so the attack effects on unrelated issues are relatively similar.

[4] Note that to make a fair comparison on the data set given by Microsoft Prompt Bench, we use chatgpt 3.5 instead of gpt-3.5-turbo-1106 here.

Table 1: Comparison of effects of our method G2PIA with the token manipulation on multiple LLM models and data sets: including 2 ChatGPT models 3 Llama models, and 8 public data sets. The maximum and minimum values of the metrics are represented in blue and red respectively.

Models	GSM8K			Web-based QA			SQuAD2.0			Math		
	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑
gpt-3.5-turbo-1106	77.82	42.36	45.57	69.67	39.04	44.19	96.67	45.28	53.16	89.16	60.88	31.72
gpt-4-1106-preview	85.34	47.63	44.19	78.67	41.46	38.87	96.71	40.19	58.44	98.33	71.20	27.59
Llama-2-7b-chat	44.87	27.12	39.56	47.67	14.11	60.61	78.67	34.53	56.11	79.33	44.50	43.90
Llama-2-13b-chat	49.54	31.08	37.26	58.67	27.26	34.90	94.67	42.67	54.93	89.67	56.36	37.15
Llama-2-70b-chat	56.48	33.41	40.85	70.20	31.69	27.43	93.33	57.83	38.04	94.67	64.31	32.07
Models	Commonsense QA			Strategy QA			SVAMP			ComplexWeb QA		
	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑
gpt-3.5-turbo-1106	78.10	38.20	51.09	77.64	56.57	27.14	81.37	66.18	18.67	70.40	39.84	43.41
gpt-4-1106-preview	82.80	49.70	39.98	86.20	55.88	35.17	96.70	74.50	22.96	76.82	40.38	47.44
Llama-2-7b-chat	71.43	37.21	47.91	67.12	46.80	30.27	81.54	59.20	27.40	39.94	29.16	26.99
Llama-2-13b-chat	74.23	38.60	48.00	82.36	48.14	41.55	82.33	67.10	18.50	51.15	35.39	30.80
Llama-2-70b-chat	79.34	50.88	35.87	87.41	57.48	34.24	92.21	74.36	19.36	53.90	43.50	19.29

Table 2: Comparison of effects of our method G2PIA with the misleading adversarial attack on multiple LLM models and data sets. The maximum and minimum values of the metrics are represented in blue and red, respectively.

Models	GSM8K			Web-based QA			SQuAD2.0			Math		
	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑
gpt-3.5-turbo-1106	77.82	40.04	61.72	69.67	28.30	59.83	96.67	15.28	84.19	89.16	20.01	38.32
gpt-4-1106-preview	85.34	57.48	45.87	78.67	30.60	61.10	96.71	30.19	68.78	98.33	26.95	36.80
Llama-2-7b-chat	44.87	34.20	23.78	47.67	29.20	38.75	78.67	16.01	79.65	79.33	56.30	29.03
Llama-2-13b-chat	49.54	45.00	9.16	68.67	34.60	41.03	94.67	19.67	79.22	89.67	53.40	40.25
Llama-2-70b-chat	56.48	47.30	16.25	70.20	36.30	48.29	93.33	25.4	72.78	94.67	61.80	34.72
Models	Commonsense QA			Strategy QA			SVAMP			ComplexWeb QA		
	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑
gpt-3.5-turbo-1106	78.10	40.04	69.05	77.64	58.90	24.14	81.37	64.87	20.28	70.40	23.21	67.03
gpt-4-1106-preview	82.80	57.48	65.46	86.20	66.33	23.05	96.70	77.92	19.42	76.82	27.35	64.39
Llama-2-7b-chat	71.43	34.20	73.12	67.12	55.60	17.16	81.54	65.60	19.55	39.94	19.27	51.75
Llama-2-13b-chat	74.23	45.00	69.70	82.36	59.40	27.88	82.33	71.58	13.06	51.15	23.78	53.50
Llama-2-70b-chat	79.34	47.30	59.79	87.41	63.30	27.58	92.21	76.41	17.13	53.90	35.57	34.1

Table 3: Comparison of the effectiveness of our method with other black-box attack methods.

Models	SQuAD2.0			Math			GSM8K			
	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	Avg. Time
BertAttack	71.16	24.67	65.33	72.30	44.82	38.01	77.82	34.26	55.98	1.04s
DeepWordBug	71.16	65.68	7.70	72.30	48.36	33.11	77.82	25.67	67.01	1.18s
TextFooler	71.16	15.60	78.08	72.30	46.80	35.27	77.82	24.33	68.74	2.80s
TextBugger	71.16	60.14	16.08	72.30	47.75	33.96	77.82	52.61	32.40	1.57s
Stress Test	71.16	70.66	0.70	72.30	39.59	45.24	77.82	35.19	54.78	2.84s
Checklist	71.16	68.81	3.30	72.30	36.90	48.96	77.82	44.33	43.04	1.32s
Ours (Token Manipulation)	71.16	14.91	79.05	72.30	13.39	**81.48**	77.82	22.17	**71.51**	1.75s
Ours (Misleading Adversarial Attack)	71.16	12.08	**83.02**	72.30	33.39	53.82	77.82	32.04	58.83	1.73s

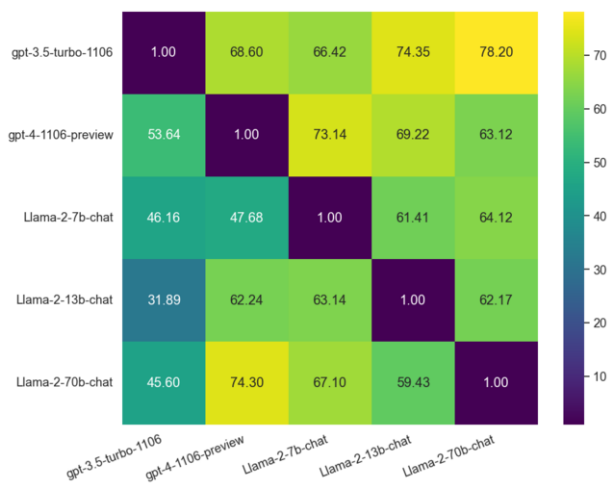

Figure 4: Transfer Success Rate (TSR) heatmap on our method with misleading adversarial attack. The rows and columns represent the attack model and defense model, respectively.

5 Conclusion

We propose a target-driven attack on LLMs, which transforms the attack problem into a KL-divergence maximization problem between the posterior probabilities of two conditions, the clean text and the attack text. Subsequently, the KL-divergence maximization problem is transformed into a well-defined convex optimization problem. At the same time, it is proved theoretically that maximizing the KL-divergence is approximately equivalent to maximizing the Mahalanobis distance between normal text and attack text. The projected gradient algorithm is used to find the optimal solution to the problem, which is the vector corresponding to the attack text. Two attack algorithms are designed including token manipulation attack and misinformation attack. Experimental results on multiple public datasets and LLM models verify the effectiveness of the proposed method.

Our attack can improve the robustness of model, e.g. an intuitive defense method (or adversarial training) is to find attack samples in a certain epsilon neighborhood of the clean samples, so that the Mahalanobis distance between the clean samples and the attack samples is minimized.

References

[1] F. Al-Turjman and B. D. Deebak. Privacy-aware energy-efficient framework using the internet of medical things for COVID-19. *IEEE Internet Things Mag.*, 3(3):64–68, 2020. doi: 10.1109/IOTM.0001.2000123. URL https://doi.org/10.1109/IOTM.0001.2000123.

[2] C. Chen and K. Shu. Combating misinformation in the age of llms: Opportunities and challenges. *CoRR*, abs/2311.05656, 2023. doi: 10.48550/ARXIV.2311.05656. URL https://doi.org/10.48550/arXiv.2311.05656.

[3] C. Chen and K. Shu. Can llm-generated misinformation be detected? *CoRR*, abs/2309.13788, 2023. doi: 10.48550/ARXIV.2309.13788. URL https://doi.org/10.48550/arXiv.2309.13788.

[4] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[5] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. Hotflip: White-box adversarial examples for text classification, 2018.

[6] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE SPW*, pages 50–56. IEEE, 2018.

[7] S. Goyal, S. Doddapaneni, M. M. Khapra, and B. Ravindran. A survey of adversarial defenses and robustness in nlp. *ACM Comput. Surv.*, 55 (14s), jul 2023. ISSN 0360-0300.

[8] C. Guo, A. Sablayrolles, H. Jégou, and D. Kiela. Gradient-based adversarial attacks against text transformers, 2021.

[9] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

[10] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the 2020 conference on Association for the Advancement of Artificial Intelligence*, volume 34, pages 8018–8025, 2020.

[11] E. Jones, A. Dragan, A. Raghunathan, and J. Steinhardt. Automatically auditing large language models via discrete optimization, 2023.

[12] J. Li, S. Ji, T. Du, B. Li, and T. Wang. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*, 2018.

[13] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*, 2020.

[14] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu. Bert-attack: Adversarial attack against bert using bert, 2020.

[15] Y. Liu, G. Deng, Y. Li, K. Wang, T. Zhang, Y. Liu, H. Wang, Y. Zheng, and Y. Liu. Prompt injection attack against llm-integrated applications, 2023.

[16] I. R. McKenzie, A. Lyzhov, M. Pieler, A. Parrish, A. Mueller, A. Prabhu, E. McLean, A. Kirtland, A. Ross, A. Liu, A. Gritsevskiy, D. Wurgaft, D. Kauffman, G. Recchia, J. Liu, J. Cavanagh, M. Weiss, S. Huang, T. F. Droid, T. Tseng, T. Korbak, X. Shen, Y. Zhang, Z. Zhou, N. Kim, S. R. Bowman, and E. Perez. Inverse scaling: When bigger isn't better, 2023.

[17] J. X. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp, 2020.

[18] OpenAI. Gpt-4 technical report, 2023.

[19] F. Perez and I. Ribeiro. Ignore previous prompt: Attack techniques for language models, 2022.

[20] A. Radford, J. Wu, R. Child, D. Luan, A. B. Santoro, S. Chaplot, A. Patra, and I. Sutskever. Chatgpt: A language model for conversational agents. *OpenAI*, 2020. URL https://openai.com/research/chatgpt.

[21] P. Rajpurkar, R. Jia, and P. Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

[22] M. T. Ribeiro, S. Singh, and C. Guestrin. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th ACL*, pages 856–865, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[23] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*, 2020.

[24] E. Shayegani, M. A. A. Mamun, Y. Fu, P. Zaree, Y. Dong, and N. Abu-Ghazaleh. Survey of vulnerabilities in large language models revealed by adversarial attacks, 2023.

[25] X. Shen, Z. Chen, M. Backes, Y. Shen, and Y. Zhang. "Do Anything Now": Characterizing and Evaluating In-The-Wild Jailbreak Prompts on Large Language Models. *CoRR abs/2308.03825*, 2023.

[26] T. Shin, Y. Razeghi, R. L. L. I. au2, E. Wallace, and S. Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts, 2020.

[27] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. 2023.

[28] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh. Universal adversarial triggers for attacking and analyzing nlp, 2021.

[29] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh. Benchmarking language models' robustness to semantic perturbations. In *NAACL*, pages 1793–1813, 2022.

[30] J. Wei and K. Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks, 2019.

[31] Z. Zhou, Q. Wang, M. Jin, J. Yao, J. Ye, W. Liu, W. Wang, X. Huang, and K. Huang. Mathattack: Attacking large language models towards math solving ability. *arXiv preprint arXiv:2309.01686*, 2023.

[32] K. Zhu, J. Wang, J. Zhou, Z. Wang, H. Chen, Y. Wang, L. Yang, W. Ye, N. Z. Gong, Y. Zhang, et al. Promptbench: Towards evaluating the robustness of large language models on adversarial prompts. *arXiv preprint arXiv:2306.04528*, 2023.