Goal-guided Generative Prompt Injection Attack on Large Language Models

Chong Zhang*,1, Mingyu Jin*,2, Qinkai Yu³, Chengzhi Liu¹, Haochen Xue¹, and Xiaobo Jin†,1

¹ Xi'an Jiaotong-Liverpool University
² Rutgers University
³ University of Liverpool
chong.zhang19@student.xjtlu.edu.cn, mj939@scarletmail.rutgers.edu,
xiaobo.jin@xjtlu.edu.cn *

Abstract. Current large language models (LLMs) provide a strong foundation for large-scale user-oriented natural language tasks. A large number of users can easily inject adversarial text or instructions through the user interface, thus causing LLMs model security challenges. Although there is currently a large amount of research on prompt injection attacks, most of these black-box attacks use heuristic strategies. It is unclear how these heuristic strategies relate to the success rate of attacks and thus effectively improve model robustness. To solve this problem, we redefine the goal of the attack: to maximize the KL divergence between the conditional probabilities of the clean text and the adversarial text. Furthermore, we prove that maximizing the KL divergence is equivalent to maximizing the Mahalanobis distance between the embedded representation x and x' of the clean text and the adversarial text when the conditional probability is a Gaussian distribution and gives a quantitative relationship on x and x'. Then we designed a simple and effective goalguided generative prompt injection strategy (G2PIA) to find an injection text that satisfies specific constraints to achieve the optimal attack effect approximately. It is particularly noteworthy that our attack method is a guery-free black-box attack method with low computational cost. Experimental results on seven LLM models and four datasets show the effectiveness of our attack method.

Keywords: Prompt Injection \cdot KL-divergence \cdot Robustness \cdot Mahalanobis Distance.

1 Introduction

Large Language Models (LLMs) [19] are rapidly evolving in terms of architecture and functionality. As they become more deeply integrated into complex systems, the urgency to review their security properties also increases. Previous work

^{* *} Equal contribution.

[†] Corresponding authors.

has shown that even LLMs whose instructions are tuned through reinforcement learning with human feedback (RLHF) are vulnerable to adversarial attacks. Therefore, it is significant to study adversarial attacks on large language models. It can help researchers understand the security and robustness of large language models [26] to design more robust models to prevent such attacks.

Previous attacks on large models 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. [10] 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. However, these methods can only attack open-source large language models but are powerless against more widely used closed-source LLMs such as ChatGPT.

A black-box attack limits the attacker to access only API-type services. 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 [12] to identify the most critical words in the text and replace them, or use some simple and general text enhancement methods [18], 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. However, 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.

In our work, we assume that the clean text representation x and the adversarial text representation x' satisfy the conditional probability distribution p(y|x) and p(y|x') respectively, and the goal of the black-box attack is to maximize the KL divergence $\mathrm{KL}(p(y|x),p(y|x'))$, then we prove that maximizing the KL divergence is equivalent to maximizing the Mahalanobis distance between x and x' under the assumption of Gaussian distribution. Furthermore, we give the quantitative relationship between optimal attack text representation x'^* and x. Based on the above theoretical results, we designed a simple and effective prompt text injection method to search for attack texts that meet approximately optimal conditions.

Overall, **our contributions** are as follows: **1)** We propose a new objective function based on KL divergence between two conditional probabilities for black-box attacks to maximize the success rate of black-box attacks; **2)** We theoretically prove that under the assumption that the conditional probabilities are Gaussian distributions, the KL divergence maximization problem based on the posterior probability distributions of clean text and adversarial text, respectively, is equivalent to maximizing the Mahalanobis distance between clean text and adversarial text. **3)** We propose a simple and effective injection attack strategy for generating adversarial injection texts, and the experimental

results verify the effectiveness of our method. Note that our attack method is a query-free black-box attack method with low computational cost.

2 Related Work

In this section, we mainly review white-box and black-box attacks. Gradient-based attacks are in the white-box category, while jailbreak prompting, token manipulation, and prompt injection are all black-box attacks.

2.1 Gradient-based Attack

Gradient-based Distributional Attack (GBDA) [10] uses the Gumbel-Softmax approximation technique to make the adversarial loss optimization differentiable while using BERTScore and perplexity to enhance perceptibility and fluency. HotFlip [4] maps text operations into vector space and measures the loss derivatives with respect to these vectors. AutoPrompt [27] uses a gradient-based search strategy to find the most effective prompt templates for different task sets. Autoregressive stochastic coordinate ascent (ARCA) [13] considers a broader optimization problem to find input-output pairs that match a specific pattern of behaviour. Wallace et al. [29] 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 not suitable for these large language models. Our work belongs to the black-box attack method.

2.2 Jailbreak Prompting Attack

Jailbreak prompts hostile triggering of LLM to output harmful content. Jailbreaking is a black-box attack, so the wording combination is based on heuristics and manual exploration. Wei et al. [31] proposed two failure modes of LLM security to guide the design of jailbreak attacks: competing objectives and mismatched generalization. They also experiment with many jailbreak methods, including combined strategies such as the composition of prefix injection, style injection and the Base64 attack, etc. When observing prompt injection attacks, Greshak et al. [8] found that even if the attack does not provide detailed methods but only targets, the model may have access to more information that brings more risks such as phishing, private probing, and even proprietary information. It is worth noting that the goal of a jailbreaking prompting attack is to guide LLM to generate answers that exceed the safe range or alignment, while our attack method is a prompt injection attack, and the goal is to insert an adversarial prompt to make LLM generate wrong answers.

2.3 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. [24] manually define heuristic Semantic Equivalent Adversary Rules (SEAR) to perform minimal labelling operations, thereby preventing the model from generating correct answers. In contrast, Easy Data Augmentation (EDA) [32] 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 natural use case for masked language models, BERT-Attack [15] 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.

2.4 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 [17] to embed instructions in the data and trick the model into understanding it. Perez&Ribeiro [21] 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 behaviour and change the user experience. Liu et al. [16] 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. 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.

2.5 Black-box Attack Paradigm

The black box attack paradigm refers to a broad range of attack methods that exploit vulnerabilities in machine learning models. Boundary query free attack BadNets [9] and boundary query related attack [6] are two common black box attack methods, whose difference lies in whether the boundaries of the model need to be input in advance to observe the output of the model. Other black box attack methods include methods based on model substitution such as practical black box attack[20] and context injection-based attacks, e.g. BERT-Attack [15].

3 Methodology

3.1 Threat Model with Black-box Attack

Adversarial scope and goal. Given a text t containing multiple sentences, we generate a text t' to attack the language model, ensuring that the meaning

of the original text t is preserved. Here, we use $\mathcal{D}(t',t)$ to represent the distance between the semantics of text t and t'. If the LLM outputs M(t) and M(t') differ, then t' is identified as an adversarial input for M. Our objective is to generate a text t' that satisfies the following conditions

$$M(t) = r, \quad M(t') = r', \quad r \neq r', \quad \mathcal{D}(t', t) = 0,$$
 (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. Note that here we assume that the output text r and r' come from some candidate set, so equality = makes sense. 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 M(t') = r' such that $r' \neq r$ on one hand and maintains high benign accuracy with M(t) = r on the other hand.
- Imperceptible: Prompt injection attacks are often detected easily by inserting garbled code that can disrupt large models. We try to ensure that the adversarial text fits better into the problem context so that the model's active defence mechanisms make it difficult to detect the presence of our prompt injections. The semantic constraint $\mathcal{D}(t',t)$ can be replaced by a more relaxed constraint

$$\mathcal{D}(t',t) < \epsilon. \tag{2}$$

That is, the distance $\mathcal{D}(\cdot,\cdot)$ between the clean text t and the adversarial text t' is less than a given small threshold ϵ with $\epsilon > 0$.

Input-dependent: Compared to fixed triggers, input-dependent triggers are imperceptible and difficult to detect from humans in most cases [30]. It is obvious the adversarial text (or trigger) t' is input-dependent by Eqn. (2). We will insert trigger t' into t through a prompt injection manner to form an attack prompt.

3.2 Analysis on Objective

Below, we first discuss the necessary conditions for the LLM model to output different values $(M(t) \neq M(t'))$ 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 model M are both texts. To facilitate analysis, we discuss the 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** embedded function from text to vector. Now, we re-formulate the output of the LLM model M as a maximization problem of posterior probability in an enumerable discrete space \mathcal{Y}

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

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

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

So, we get its converse proposition

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

That is, LLM has different posterior probability distributions under different input conditions, which is a necessary condition for LLM to output different values such that $M(t) \neq M(t')$. 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

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

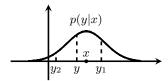


Fig. 1. Assumption that the output y of LLM under the condition of x satisfies the discrete Gaussian distribution: Answers (output) y that are close to question (input) x are usually more relevant to x and have a higher probability of being sampled.

First, we assume that the output distribution p(y|x) of LLM satisfies the discrete Gaussian distribution [2] under the condition of input x,

$$p(y|x) = \frac{e^{-\frac{1}{2}(y-x)^T \Sigma^{-1}(y-x)}}{\sum_{\hat{y} \in \mathcal{Y}} e^{-\frac{1}{2}(\hat{y}-x)^T \Sigma^{-1}(\hat{y}-x)}},$$
(7)

as shown in Fig. 1, that is, the output of LLM is defined on a limited candidate set \mathcal{Y} , although this candidate set \mathcal{Y} may be very large. Since the input x and output y of LLM are questions and answers, the spaces where they are located generally do not intersect with each other, so there are $y = \arg\max_{\hat{y} \in \mathcal{Y}} p(\hat{y}|x) \neq x$, which is different from the commonly used continuous Gaussian distribution y = x, as can be seen in Fig. 1. For the same question x, LLM usually outputs different answers and the answer y most relevant to x has a higher probability of being sampled. In the embedding space, the distance between y and x is usually closer. Similarly, answers y that are almost uncorrelated with x and that are far away from x in the embedding space have a smaller probability of being sampled.

In order to solve the problem (6) theoretically, we do the following processing: 1) Replace the discrete Gaussian distribution with a continuous Gaussian

distribution to facilitate the calculation of KL divergence although the output text on condition of input text still obey discrete distribution in practical applications; (2) 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 in NLP. Subsequently, we present the following theorem (see Appendix Section A.2 and A.1).

Theorem 1. Assuming that variable x' is a unit vector, and p(y|x) and p(y|x') respectively follows the Gaussian distribution $\mathcal{N}_1(y;x,\Sigma)$ and $\mathcal{N}_2(y;x',\Sigma)$, then the maximization problem of KL(p(y|x),p(y|x')) is equivalent to the following optimization problem

$$\min_{x'} \|x'\|_2, \quad s.t. \quad (x'-x)^T \Sigma^{-1} (x'-x) \le 1, \tag{8}$$

which has an optimal solution of the form (λ is Lagrange multiplier)

$$x'^* = (\Sigma + \lambda I)^{-1} \lambda x, \quad \lambda > 0. \tag{9}$$

3.3 Solution of Problem (8) and Cos Similarity

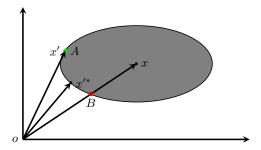


Fig. 2. Assuming $x'^* = (x'_1, x'_2)$ is the optimal solution to problem (8), then when x' moves from A through x'^* to B on the ellipse, $\cos(x', x)$ first increases and then decreases, while $||x'||_2$ first decreases and then increases.

Note that our method is a black-box attack and does not know the model parameters Σ , so we cannot solve the problem (8). Below, we try to approximately solve the problem (8) using cos similarity, which does not contain any parameters. Fig. 2 shows the optimal solution x'^* of the problem (8) in two-dimensional space. When the vector x' moves from A to B along the ellipse through the optimal point x'^* , $\cos(x',x)$ first increases and then decreases, while $\|x'\|_2$ first decreases and then increases. Therefore, we introduce the hyperparameter γ to approximate the solution to problem (8)

$$\cos(x', x) = \gamma, \quad 0 \le \gamma \le 1, \tag{10}$$

where x (known) and x' (unknown) are the embedded representations of clean input and adversarial input respectively. Note that in our implementation, we relax the constraint satisfaction problem of the optimal solution x'^* as

$$|\cos(x', x) - \gamma| < \delta$$
, δ is a small positive constant. (11)

Below we discuss the problems of $\gamma \neq 0$ and $\gamma \neq 1$ from two perspectives. First, we prove this conclusion mathematically. We compute $\cos(x^{\prime*}, x)$ to obtain

$$\cos(x'^*, x) = \frac{{x'^*}^T x}{\|x'^*\|_2 \|x\|_2} = \frac{\lambda x^T (\Sigma + \lambda I)^{-1} x}{\|x'^*\|_2 \|x\|_2}.$$
 (12)

If $\cos(x'^*,x)=0$ $(x\neq 0)$, then $\lambda=0$, that is, $x'^*=0$, which is meaningless to LLM. If $\cos(x'^*,x)=1$, then x'^* and x are in the same direction, i.e. $x'^*=\lambda(\Sigma+\lambda I)^{-1}x=tx$, where t is a ratio value. So x'^* must be the eigenvector of the matrix $\lambda(\Sigma+\lambda I)^{-1}$. However, x'^* can be an embedding representation of any input. So we arrive at a contradiction.

From the perspective of a black-box attack, when $\cos(x',x) = 1$, then the vectors x and x' will have the same direction. Since in NLP, when using vectors (often using unit vectors) to represent text, we care more about the direction of the vector, so x = x'. In addition, note that $w(\cdot)$ is a bijective function, then for clean text t and adversarial text t', there is t = t'. There we have r = M(t) = M(t') = r', which contradicts the condition $r \neq r'$ in problem (1). When $\cos(x', x) = 0$, then w(t') and w(t) are linearly uncorrelated, which is conflicted with the condition D(t', t) = 0 in problem (1).

3.4 Goal-guided Generative Prompt Injection Attack

Note that w(t) = x and w(t') = x', based on the previous discussion we can simplify our problem (1) into the following constraint satisfaction problem (CSP)

$$\min_{\mathcal{U}} 1, \tag{13}$$

$$s.t. \ \mathcal{D}(t',t) < \epsilon,$$
 (14)

$$|\cos(w(t'), w(t)) - \gamma| < \delta, \tag{15}$$

where x and x' represent clean input (known) and adversarial input (unknown) respectively, while $w(\cdot)$ represents the embedded representation of the text (literal meaning) and $\mathcal{D}(\cdot,\cdot)$ represents the distance between the semantics (intrinsic meaning) of the two texts. The hyperparameters δ and ϵ are used to control the difficulty of searching constraint, where δ or ϵ is smaller, the search accuracy is higher.

In our method, we implement a black-box attack through prompt injection: generate an adversarial text t' that satisfies conditions (15) and (14), and then mix t' into the text t to obtain a prompt \bar{t} . The advantage of using prompt insertion is that since the prompt \bar{t} contains both clean input t and adversarial input t', on one hand, the concealment of the adversarial input (or trigger) t'

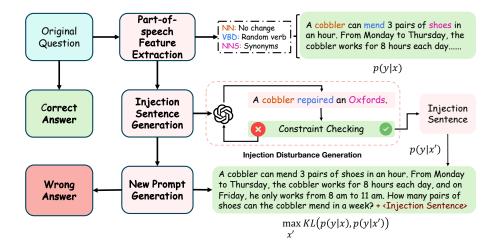


Fig. 3. Overview of Goal-guided Generative Prompt Injection Attack: 1) We use the part-of-speech method to find the subject, predicate and object of the question in the clean text x and fetch synonyms of the predicate and object plus a random number as core words; 2) Put the core words into assistant LLM to generate an adversarial text x' that satisfies the constraints; 3) Insert the generated adversarial text into the clean text x to form the final attack text; 4) Enter the attack text into the LLM victimization model to test the effectiveness of our attack strategy.

is enhanced, and on the other hand the adversarial input t' plays a very good interference role to the output of the LLM model.

Next, we first find the core word set that determines the text semantics through the semantic constraint condition (14), and then use the core word set to generate adversarial text that satisfies the cos similarity condition (15). It is worth noting that the embedded representation is defined on texts, so we use the BERT model to convert any text t into w(t). However, in our work, the semantic distance between texts is defined on the core words of the text, so we use the word2vec model to define the semantic distance $\mathcal{D}(t',t)$ between text t' and t.

3.5 Solving Semantic Constraint(14)

Usually, the semantic meaning of text r is determined by a few core words $C(r) = \{\omega_1(r), \omega_2(r), \dots, \omega_n(r)\}$, which will not be interfered by noise words in the text r. Based on the core word set C(r), we will use cos similarity to define the semantic distance $\mathcal{D}(t',t)$ between two texts t and t'

$$\mathcal{D}(t',t) = 1 - \cos(s(\omega_i(t')), s(\omega_i(t))), \quad i = 1, 2, \dots, n,$$
(16)

where $s(\cdot)$ represents the word2vec representation of the word.

In a text paragraph, usually the first sentence is a summary of the entire paragraph (may be some exceptions that we ignore), where the meaning of a text

will be determined by its subject, predicate and object. Therefore, the subject S_t , predicate P_t and object O_t that appear first in text t will serve as the core words of the text (n = 3)

$$\omega_1(t) = S_t, \quad \omega_2(t) = P_t, \quad \omega_3(t) = O_t. \tag{17}$$

Because the change of the subject will have a great impact on the meaning of the text, the subject $S_{t'}$ in the adversarial text t' is directly set to the subject S_t in the clean text t. Through the wordnet tool, we randomly select a word from the synonym lists of P_t or O_t to check whether the constraints (14) are met. Once the conditions are met, the search process will stop. Finally, we obtain the core word set of the adversarial text t' that satisfies the semantic constraints

$$C(t') = \{\omega_1(t') = S_{t'} = S_t, \quad \omega_2(t') = P_{t'}, \quad \omega_3(t') = O_{t'}\}.$$
 (18)

3.6 Solving Cos Similarity Constraint (15)

Next, we will generate adversarial text that satisfies constraint (15) through the core vocabulary C(t') of adversarial text. Note that in order to increase the randomness of the sentence, we add another random number $N_{t'}$ between 10 and 100 as the core word. Now the core vocabulary of the adversarial text t' becomes (n=4)

$$C(t') = \{\omega_1(t') = S_{t'}, \quad \omega_2(t') = P_{t'}, \quad \omega_3(t') = O_{t'}, \quad \omega_4(t') = N_{t'}\}.$$
 (19)

The core word set is embedded into the prompt template to generate a sentence text t' with LLM. We iterate N times to randomly generate multiple sentence texts t' until the text t' satisfies Eqn.(15). Finally, we insert the adversarial text t' after the text t to attack the LLM. In fact, it is also feasible to insert t' after any sentence in t. In Appendix Section A.5, we will see that the difference in attack effectiveness at different locations is minimal.

4 Experiments

4.1 Experimental Details

Below we describe some details of the prompt insertion-based attack method, including the victim model, dataset, and evaluation metrics. In particular, ChatGPT-4-Turbo (gpt-4-0125-preview) is used as our auxiliary model to generate random sentences that comply with grammatical rules. Unless otherwise stated, all results of our algorithm use the parameter settings $\epsilon=0.2,\,\delta=0.05$ and $\gamma=0.5$. We randomly selected 300 examples from the following dataset and tested them using the following two large model families.

Victim Models

- ChatGPT. ChatGPT is a language model created by OpenAI that can produce conversations that appear to be human-like [22]. 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 [28] 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 knowledge tests. Llama 2-7B, Llama 2-13B and Llama 2-70B are transformer framework-based models.

Q&A Datasets We chose datasets for plain text and mathematical Q&A scenarios.

- GSM8K The GSM8K dataset is an extensive language model training resource that comprises 800 billion words [1], making it the most massive language dataset available today.
- web-based QA The dataset [3] is mostly obtained from online Question Answering communities or forums through Web crawlers.
- MATH dataset The MATH dataset [11] has 12,000+ question-answer pairs for researchers to develop and evaluate problem-solving models.
- SQuAD2.0 SQuAD2.0 [23] has 100K+ question-answer pairs from Wikipedia for reading comprehension.

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 $\mathcal{A}_{\text{clean}} = \frac{|D|}{|T|}$.
- Attack Accuracy The Attack Accuracy metric measures the accuracy on adversarial attack inputs $\mathcal{A}_{\text{attack}} = \frac{|\sum_{(x,y) \in D} f(a(x)) = y|}{|T|}$.
 Attack Success Rate (ASR) The attack success rate indicates the rate
- 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 = $\frac{|\sum_{(x,y)\in D} f(a(x))\neq y|}{|D|}$. It is worth noting that for the above three measurements, we have the following relationship ASR = $1 \frac{A_{\text{attack}}}{A_{\text{clean}}}$.

4.2 Main Results

The experimental results are as shown in Tab. 1. Although we set the insertion position at the end of the clean text, we also give the attack effects of different insertion positions in Appendix Section A.5.

Table 1. Comparison of attack effects of our method G2PIA on seven LLM models and four data sets: including 4 ChatGPT models and 3 Llama models

Models		$\overline{\mathbf{GSM8}}$	K	Web-based QA			
Models	$\mathcal{A}_{ ext{clean}}$	$\mathcal{A}_{\mathrm{attack}}$	$ASR\uparrow$	$\mathcal{A}_{ ext{clean}}$	$\mathcal{A}_{ ext{clean}}$	$\overline{ASR\uparrow}$	
text-davinci-003	71.68	36.94	48.47	41.87	17.97	57.19	
${\tt gpt\text{-}3.5\text{-}turbo\text{-}0125}$	72.12	37.80	47.60	41.98	24.17	42.42	
gpt-4-0613	76.43	41.67	45.48	53.63	33.72	37.12	
${ m gpt} ext{-}4 ext{-}0125 ext{-}{ m preview}$	77.10	43.32	43.81	54.61	34.70	32.80	
llama-2-7b-chat	44.87	27.51	38.69	47.67	24.26	49.10	
llama-2-13b-chat	49.54	35.51	28.33	58.67	36.14	38.40	
llama-2-70b-chat	56.48	39.90	29.36	70.20	48.18	31.47	
Models	SQuAD2.0 Dataset			Math Dataset			
Wiodels	$\mathcal{A}_{\mathrm{clean}}$	$\mathcal{A}_{\mathrm{attack}}$	$ASR\uparrow$	$\mathcal{A}_{ ext{clean}}$	$\mathcal{A}_{\mathrm{attack}}$	$\overline{ASR\uparrow}$	
text-davinci-003	68.30	14.00	79.50	21.33	11.76	44.87	
${ m gpt} ext{-}3.5 ext{-}{ m turbo} ext{-}0125$	68.33	12.67	81.46	21.33	15.99	29.72	
gpt-4-0613	71.87	19.71	72.58	41.66	28.33	32.00	
${ m gpt} ext{-}4 ext{-}0125 ext{-}{ m preview}$	71.94	24.03	69.34	44.64	32.83	26.49	
llama-2-7b-chat	78.67	37.66	52.13	79.33	52.44	33.90	
llama-2-13b-chat	94.67	52.70	44.33	89.67	56.72	36.75	
llama-2-70b-chat	93.33	40.78	56.31	94.67	71.82	24.14	

The results on four public datasets show that the first-generation ChatGPT-3.5 and ChatGPT-3.5-Turbo have the lowest defence capabilities. Obviously, when ChatGPT first came out, it didn't think too much about being attacked. Similarly, the small model 7b of Llama-2 is also very weak in resisting attacks. Of course, it is indisputable that the clean accuracy of the models of the Llama series is also very low. The output of small models is more susceptible to noise.

On the other hand, taking ChatGPT-4 as an example, if we compare the ASR values on the 4 data sets, we can conclude that our attack algorithm is more likely to succeed on the data set SQuAD 2.0, while mathematical problems are the most difficult to attack. In contrast to [33] ASR 41.15 with ChatGPT-3.5 on GSM8k, our attack algorithm with ASR 44.87 is a general attack strategy and is not specifically designed for problems involving mathematical reasoning.

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 [23] and Math dataset [11]. Microsoft Prompt Bench [34] uses the following black box attack methods to attack the ChatGPT-3.5 language model, including BertAttack [15], DeepWordBug [5], TextFooler [12], TextBugger [14], Prompt Bench [34], Semantic and CheckList [25]. For the sake of fairness, we also use our method to attack ChatGPT 3.5. Tab. 2 gives the comparison of the results of these methods on the three measurements of Clean Acc, Attack Acc and ASR.

Multiple attack strategies attack ChatGPT-3.5 on two data sets, the SQuAD2.0 dataset and the Math dataset, respectively. As can be seen from Tab. 2, our at-

SQuAD2.0 dataset Math Dataset Models Query $A_{\text{clean}} A_{\text{attack}} ASR \uparrow A_{\text{clean}} A_{\text{attack}} ASR \uparrow$ BertAttack [15] Dependent 71.16 24.67 65.33 $22.27 \quad 14.82$ $\label{eq:DeepWordBug} \mbox{ [5] Dependent } 70.41 - 65.68$ 6.7222.07 18.3616.83 TextFooler [12] Dependent 72.87 21.7116.80 26.02 15.60 78.59 TextBugger [14] Both 71.6660.1416.08 21.7317.7518.31 Stress Test [25] Free 71.9470.66 1.78 21.33 19.59

68.81

14.00

3.64

79.50

22.07

21.33

16.90

11.76

23.41

44.87

71.41

68.30

Free

Free

Table 2. Our method is compared with other methods on two datasets

tack strategy achieves the best results on both data sets. It is worth noting that we count Clean Acc and Attack Acc for each algorithm at the same time, so there are subtle differences between the multiple Clean Acc shown in Tab. 2, but since Clean Acc and Attack Acc are calculated in the same attack algorithm, therefore it has little effect on the value of ASR. Especially on the Math dataset, our algorithm is significantly better than other algorithms, with an ASR of 44.87% compared to BertAttack's 33.46%. However, our algorithm is a general attack method and is not specifically designed for mathematical problems. To some extent, it is shown that our algorithm has good transfer ability on different types of data sets.

5 Ablation Study

CheckList [25]

Ours

In this section, we will analyze our baseline approach by conducting ablation studies based on two strategies. The first strategy involves extracting sentence components, while the second strategy involves traversing insertion positions. To extract sentence components, we randomly replaced all three components with synonyms. We also randomly performed an ablation study with random breakpoint insertion. The results show that the average ASRs of random location prompt injection and random sentence component replacement multiple times are lower than our method.

Table 3. Ablation studies on datasets of GSM8K and Web-based QA with gpt-3.5-turbo (gpt-3.5-turbo-0125).

Models		GSM8I	K	Web-based QA		
Wiodels	$\overline{\mathcal{A}_{ ext{clean}}}$	$\mathcal{A}_{\mathrm{attack}}$	ASR↑	$\mathcal{A}_{ ext{clean}}$	$\mathcal{A}_{\mathrm{attack}}$	ASR↑
Random position prompt injection	72.12	51.08	29.18	41.98	36.20	14.20
Random component replacement	72.12	58.90	18.33	41.98	37.16	11.49
Our Method	72.12	37.80	47.60	41.98	24.17	42.42

5.1 Transferability

We use the adversarial examples generated by model A to attack model B, thereby obtaining the transferability [33] of the attack on model A. The attack success rate obtained on model B can, to some extent, demonstrate the transferability of the attack on model A. We list the ASR values of the offensive and defensive pairs of LLM into a correlation matrix, as shown in Fig. 4. It can be found that the ChatGPT-4-Turbo attack model has the strongest transferability, while Llama-2-7b has the weakest defensive ability. Obviously, this is because ChatGPT-4-Turbo is the strongest LLMs model among them, while Llama-2-7b is the weakest model. This can be found in the results in Tab. 1.

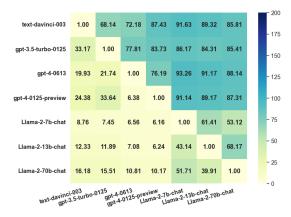
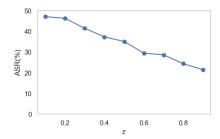


Fig. 4. Transfer Success Rate(TSR) heatmap. The rows and columns represent the attack model and defence model respectively.

5.2 Parameter sensitivity analysis

In our method, the parameters ϵ and δ are two important parameters. The former controls the distance between the adversarial text and the clean text in the semantic space, while the latter will affect the optimality of the approximate optimal solution. We selected a total of 9 values from $\{0.1, 0.2 \cdots, 0.9\}$ for the two parameters to attack ChatGPT-3.5 on the GSM8K data set and calculated their ASR values to obtain the curve as shown in Fig. 5 and 6. As shown in Fig. 5, ASR is simply a decreasing function of the distance threshold ϵ . That is, the farther the distance, the worse the attack effect. This aligns with our intuition: injected text that is too far away from the clean text will be treated as noise by LLM and ignored. Fig. 6 shows that when $\gamma = 0.5$, our attack strategy achieves the best attack effect. When the gamma value is greater than 0.5 or less than 0.5, the attack effect will be attenuated to some extent. It is worth noting that the value of parameter γ may vary depending on the model or data. See Appendix sections A.5 and A.6 for more results and discussions.



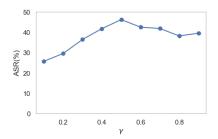


Fig. 5. ASR metric changes with the parameter ϵ

Fig. 6. ASR metric changes with the parameter γ

6 Conclusion

In our work, we propose a new goal-oriented generative prompt injection attack (G2PIA) method. In order to make the injected text mislead the large model as much as possible, we define a new objective function to maximize, which is the KL divergence value between the two posterior conditional probabilities before injection (clean text) and after injection (attack text). Furthermore, we proved that under the condition that the conditional probability follows the multivariate Gaussian distribution, maximizing the KL divergence value is equivalent to maximizing the Mahalanobis distance between clean text and adversarial text. Then we establish the relationship between the optimal adversarial text and clean text. Based on the above conclusions, we design a simple and effective attack strategy with an assisted model to generate injected text that satisfies certain constraints, which will maximize the KL divergence. Experimental results on multiple public datasets and popular LLMs demonstrate the effectiveness of our method.

References

- Brown, T., Chinchilla, J., Le, Q.V., Mann, B., Roy, A., Saxton, D., Wei, E., Ziegler, M.: Gsm8k: A large-scale dataset for language model training (2023)
- 2. Canonne, C.L., Kamath, G., Steinke, T.: The discrete gaussian for differential privacy (2021)
- 3. Chang, Y., Narang, M., Suzuki, H., Cao, G., Gao, J., Bisk, Y.: Webqa: Multihop and multimodal qa. In: CVPR. pp. 16495–16504 (2022)
- 4. Ebrahimi, J., Rao, A., Lowd, D., Dou, D.: Hotflip: White-box adversarial examples for text classification (2018)
- 5. Gao, J., Lanchantin, J., Soffa, M.L., Qi, Y.: Black-box generation of adversarial text sequences to evade deep learning classifiers. In: SPW. pp. 50–56. IEEE (2018)
- Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
- Goyal, S., Doddapaneni, S., Khapra, M.M., Ravindran, B.: A survey of adversarial defenses and robustness in nlp. ACM Comput. Surv. 55(14) (2023)
- 8. Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., Fritz, M.: Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection (2023)

- 9. Gu, T., Dolan-Gavitt, B., Garg, S.: Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733 (2017)
- Guo, C., Sablayrolles, A., Jégou, H., Kiela, D.: Gradient-based adversarial attacks against text transformers (2021)
- 11. Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., Steinhardt, J.: Measuring mathematical problem solving with the math dataset. NeurIPS (2021)
- 12. Jin, D., Jin, Z., Zhou, J.T., Szolovits, P.: Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 8018–8025 (2020)
- 13. Jones, E., Dragan, A., Raghunathan, A., Steinhardt, J.: Automatically auditing large language models via discrete optimization (2023)
- 14. Li, J., Ji, S., Du, T., Li, B., Wang, T.: Textbugger: Generating adversarial text against real-world applications. arXiv preprint arXiv:1812.05271 (2018)
- 15. Li, L., Ma, R., Guo, Q., Xue, X., Qiu, X.: Bert-attack: Adversarial attack against bert using bert. arXiv preprint arXiv:2004.09984 (2020)
- Liu, Y., Deng, G., Li, Y., Wang, K., Zhang, T., Liu, Y., Wang, H., Zheng, Y., Liu,
 Y.: Prompt injection attack against llm-integrated applications (2023)
- McKenzie, I.R., Lyzhov, A., Pieler, M., Parrish, A., Mueller, A., Prabhu, A., McLean, E., Kirtland, A., Ross, A., Liu, A., et al.: Inverse scaling: When bigger isn't better. arXiv preprint arXiv:2306.09479 (2023)
- 18. Morris, J.X., Lifland, E., Yoo, J.Y., Grigsby, J., Jin, D., Qi, Y.: Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp (2020)
- 19. OpenAI: Gpt-4 technical report (2023)
- Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: ACCC. pp. 506–519 (2017)
- 21. Perez, F., Ribeiro, I.: Ignore previous prompt: Attack techniques for language models (2022)
- Radford, A., Wu, J., Child, R., Luan, D., Santoro, A.B., Chaplot, S., Patra, A., Sutskever, I.: Chatgpt: A language model for conversational agents. OpenAI (2020)
- 23. Rajpurkar, P., Jia, R., Liang, P.: Know what you don't know: Unanswerable questions for squad. arXiv preprint arXiv:1806.03822 (2018)
- 24. Ribeiro, M.T., Singh, S., Guestrin, C.: Semantically equivalent adversarial rules for debugging NLP models. In: ACL. pp. 856–865 (2018)
- 25. Ribeiro, M.T., Wu, T., Guestrin, C., Singh, S.: Beyond accuracy: Behavioral testing of nlp models with checklist. arXiv preprint arXiv:2005.04118 (2020)
- 26. Shayegani, E., Mamun, M.A.A., Fu, Y., Zaree, P., Dong, Y., Abu-Ghazaleh, N.: Survey of vulnerabilities in large language models revealed by adversarial attacks (2023)
- 27. Shin, T., Razeghi, Y., au2, R.L.L.I., Wallace, E., Singh, S.: Autoprompt: Eliciting knowledge from language models with automatically generated prompts (2020)
- 28. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. corr, abs/2302.13971, 2023. doi: 10.48550. arXiv preprint arXiv.2302.13971 (2023)
- 29. Wallace, E., Feng, S., Kandpal, N., Gardner, M., Singh, S.: Universal adversarial triggers for attacking and analyzing nlp (2021)
- Wallace, E., Feng, S., Kandpal, N., Gardner, M., Singh, S.: Benchmarking language models' robustness to semantic perturbations. In: NAACL: Human Language Technologies. pp. 1793–1813 (2022)

- 31. Wei, A., Haghtalab, N., Steinhardt, J.: Jailbroken: How does llm safety training fail? (2023)
- 32. Wei, J., Zou, K.: Eda: Easy data augmentation techniques for boosting performance on text classification tasks (2019)
- 33. Zhou, Z., Wang, Q., Jin, M., Yao, J., Ye, J., Liu, W., Wang, W., Huang, X., Huang, K.: Mathattack: Attacking large language models towards math solving ability. arXiv preprint arXiv:2309.01686 (2023)
- 34. Zhu, K., Wang, J., Zhou, J., Wang, Z., Chen, H., Wang, Y., Yang, L., Ye, W., Gong, N.Z., Zhang, Y., et al.: Promptbench: Towards evaluating the robustness of large language models on adversarial prompts. arXiv preprint arXiv:2306.04528 (2023)

A Appendix

A.1 Proof on Theorem 1

Assuming that the output value of LLM is a continuous value, below we discuss the maximization problem of $\mathrm{KL}(p(y|x),p(y|z))$ under conditional Gaussian distribution

$$\max_{\|z\|=1} \text{KL}(p(y|x), p(y|z))$$

s.t. $D(s(x), s(z)) < \epsilon$. (20)

Let y be an $n \times 1$ random vector and two multivariate Gaussian distributions of y under two different conditions are

$$p(y|x) = \mathcal{N}_1(y; x, \Sigma)$$
$$p(y|z) = \mathcal{N}_2(y; z, \Sigma)$$

Then

$$KL(\mathcal{N}_{1}||\mathcal{N}_{2})$$

$$= \int \log \frac{\mathcal{N}_{1}(y; x, \Sigma)}{\mathcal{N}_{2}(y; z, \Sigma)} \mathcal{N}_{1}(y; x, \Sigma) dy$$

$$= \int [-\frac{1}{2}(y - x)^{T} \Sigma^{-1}(y - x)$$

$$+ \frac{1}{2}(y - z)^{T} \Sigma^{-1}(y - z)] \mathcal{N}_{1}(y; x, \Sigma) dy$$

$$= -\frac{1}{2} tr(\mathbb{E}_{\mathcal{N}_{1}}(y - x)(y - x)^{T} \Sigma^{-1})$$

$$+ \frac{1}{2} tr(\mathbb{E}_{\mathcal{N}_{1}}(y - z)(y - z)^{T} \Sigma^{-1})$$

$$= -\frac{1}{2} tr(I_{n}) + \frac{1}{2} tr((z - x)^{T} \Sigma^{-1}(z - x))$$

$$+ \frac{1}{2} tr(\Sigma^{-1} \Sigma)$$

$$= \frac{1}{2} (z - x)^{T} \Sigma^{-1}(z - x)$$
(21)

Therefore, assuming that the predicted value of the large model follows a Gaussian distribution, our optimization problem is equivalent to finding a certain text z that maximizes the Mahalanobis distance to x. So we optimize the following problems

$$\max_{\|z\|_2^2=1} (z-x)^T \Sigma^{-1} (z-x), \tag{22}$$

which is equivalent to the following problem

$$\max_{z \neq 0} \frac{(z-x)^T \Sigma^{-1} (z-x)}{\|z\|_2^2}.$$
 (23)

Further, we transform the above optimization problem as follows

$$\min_{z} \|z\|_{2}^{2}, \quad s.t. \quad (z-x)^{T} \Sigma^{-1}(z-x) = 1, \tag{24}$$

A.2 Solution of Problem (8)

Below we first give the form of the solution in a two-dimensional space to explain the problem (8) more intuitively, and then we will give the solution to the problem in the general case.

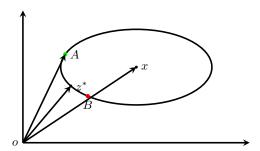


Fig. 7. When $z^* = (z_1, z_2)$ is the optimal solution to problem (8), then the vector \overrightarrow{zb} and the normal vector of the ellipse at point z^* have the same direction.

For the convenience of discussion, let Σ be a diagonal matrix

$$\Sigma = \begin{bmatrix} \sigma_1^2 \\ \sigma_2^2 \end{bmatrix} \tag{25}$$

so the constraint $(x-z)^T \Sigma^{-1}(x-z) = 1$ becomes an elliptic equation $G(z_1, z_2) = 0$ such as

$$G(z_1, z_2) = \frac{(z_1 - x_1)^2}{\sigma_1^2} + \frac{(z_2 - x_2)^2}{\sigma_2^2} - 1 = 0.$$
 (26)

It can be seen from Fig. 7 that when the vector \overrightarrow{zb} (the point o is the origin) and the normal vector at point $z=(z_1,z_2)$ have the same direction, e.g. $\overrightarrow{zb}=\lambda\nabla G(z_1,z_2)$, then point z is the optimal solution to the problem (8).

The normal vector at point $z = (z_1, z_2)$ on the ellipse is

$$\nabla G(z_1, z_2) \propto \left(\frac{z_1 - x_1}{\sigma_1^2}, \frac{z_2 - x_2}{\sigma_2^2}\right),\tag{27}$$

so we have

$$(0 - z_1, 0 - z_2) = \lambda \left(\frac{z_1 - x_1}{\sigma_1^2}, \frac{z_2 - x_2}{\sigma_2^2} \right).$$
 (28)

Finally, we get

$$z_1 = \frac{\lambda}{\lambda + \sigma^1} x_1, \quad z_2 = \frac{\lambda}{\lambda + \sigma^2} x_2, \tag{29}$$

Obviously, we can solve the value of λ by substituting (29) into Eqn. (27) and z^* . Note that when $\lambda = 0$, then λ is on the ellipse, $\|\lambda\| = 0$. When $\lambda > 0$, then λ is outside the ellipse. Otherwise, it is inside the ellipse.

Below we directly derive the relationship between z and x in general based on the KKT condition. The Lagrangian function of the optimization problem is

$$\mathcal{L}(z,\lambda) = z^{T}z + \lambda((z-x)^{T}\Sigma^{-1}(z-x) - 1). \tag{30}$$

According to the KKT condition, we have

$$\frac{\partial \mathcal{L}}{\partial z} = 2z + 2\lambda \Sigma^{-1}(z - x) = 0 \tag{31}$$

So

$$z^* = (\Sigma + \lambda I)^{-1} \lambda x. \tag{32}$$

It can be seen that Eqn. (29) is a result of applying Eqn. (32) on a twodimensional space.

Finally, if we replace the condition $(\hat{z} - x)^T \Sigma^{-1} (\hat{z} - x) = 1$ with $(\hat{z} - x)^T \Sigma^{-1} (\hat{z} - x) \leq 1$, then we have Lagrange multiplier $\lambda \geq 0$, that is

$$0 < \cos(x, z^*) = \frac{\lambda x^T (\Sigma + \lambda I)^{-1} x}{\|z^*\| \|x\|} < 1.$$
 (33)

A.3 Relationship between KL Divergence and Cos Similarity in 2-dimensional space

When point z moves on the ellipse, its length corresponds to the solution of the optimization problem: when it takes the minimum value, the optimization problem (20) takes the maximum value. It can be seen from Fig. 7 that when pointing z moves from A to B, $\cos(x',x)$ gradually increases, but the length of z first decreases and then increases. This process corresponds to that as the cos similarity increases, the KL divergence value first increases from a non-optimal value to the maximum value, and then decreases. We experimentally observed the attack effect as shown in Fig. 8, which also verified our conclusion.

A.4 Discussion on KL Divergence and Cos Similarity

In this section, we will explain the reasons for using cos similarity instead of KL divergence from another perspective.

We expand the KL divergence distance function with the first-order Taylor expansion at x to get

$$f(x') = \text{KL}(p(y|x), p(y|x, x'))$$

$$\approx \text{KL}(p(y|x), p(y|x)) + \nabla_{x'} f(x')^{T} (x' - x)$$

$$= \|\nabla_{x'} f(x')\|_{2} \cos \theta \|x' - x\|_{2},$$

$$\propto \|x' - x\|_{2}$$
(34)

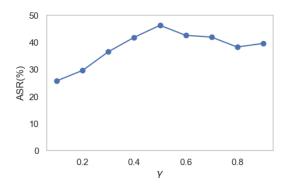


Fig. 8. The measurement ASR varies with the changing parameter γ .

where θ is the angle between the gradient $\nabla_{x'} f(x')$ and x' - x. In our problem, we assume that $z = \begin{bmatrix} x & x' \end{bmatrix}^T$ has unit length, so we have

$$||x||_2^2 + ||x'||_2^2 = 1. (35)$$

There are further

$$||x' - x||_2^2 = ||x'||_2^2 + ||x||_2^2 - 2||x'||_2 ||x||_2 \cos(x', x)$$

= 1 - 2||x'||_2 ||x||_2 \cos(x', x) (36)

Therefore, we can choose $\cos(x',x)$ as the surrogate objective function of KL divergence.

A.5 Discussion on Insertion Position of Injection Text

Although our algorithm is set to always insert the injection text at the end of the normal text, from our experiments, the insertion position of the injection text has minimal impact on the attack performance. Tab. 4 takes the most difficult-to-attack problem, such as the Math dataset, as an example and lists the prediction results of inserting injection text into the same normal text at different positions, where the injection text is highlighted in green. In this question, the output of this normal text should be the number 105. It can be seen that although the position of the injection text affects the output of LLM, these outputs are all wrong answers. This also shows that our algorithm generates injection text that has a strong ability to interfere with normal text, and the effect of the attack is not affected by the position of the injection text.

A.6 Impact of Parameters (ϵ, γ) on Attack Effects

Tab. 5 shows the joint impact of two important parameters ϵ and γ on attack performance. These results are based on the ChatGPT3.5 attack on the dataset

Table 4. Impact of the insertion position of the injected text on the attack effect: Different insertion positions of LLM output different results, but they are not equal to the correct answer.

Original Questions	Questions Injected	Answers	Attack Results
A cobbler can mend 3 pairs of shoes in an hour. From Monday to Thursday, the cobbler works for 8 hours each day, and on Friday, he only works from 8 am to 11 am. How many pairs of shoes can the cobbler mend in a week?	A cobbler can mend 3 pairs of shoes in an hour. Cobbler obviate the need to go to the office on Friday. From Monday to Thursday, the cobbler works 8 hours each day; on Friday, he only works from 8 am to 11 am. How many pairs of shoes can the cobbler mend in a week?	104	Attack success!
A cobbler can mend 3 pairs of shoes in an hour. From Monday to Thursday, the cobbler works for 8 hours each day, and on Friday, he only works from 8 am to 11 am. How many pairs of shoes can the cobbler mend in a week?	A cobbler can mend 3 pairs of shoes in an hour. From Monday to Thursday, Cobbler obviate the need to go to the office on Friday, the cobbler works for 8 hours each day, and on Friday, he only works from 8 am to 11 am. How many pairs of shoes can the cobbler mend in a week?	96	Attack success!
A cobbler can mend 3 pairs of shoes in an hour. From Monday to Thursday, the cobbler works for 8 hours each day, and on Friday, he only works from 8 am to 11 am. How many pairs of shoes can the cobbler mend in a week?	A cobbler can mend 3 pairs of shoes in an hour. From Monday to Thursday, Cobbler obviate the need to go to the office on Friday, the cobbler works for 8 hours each day, and on Friday, he only works from 8 am to 11 am. How many pairs of shoes can the cobbler mend in a week?	96	Attack success!
A cobbler can mend 3 pairs of shoes in an hour. From Monday to Thursday, the cobbler works for 8 hours each day, and on Friday, he only works from 8 am to 11 am. How many pairs of shoes can the cobbler mend in a week?	A cobbler can mend 3 pairs of shoes in an hour. From Monday to Thursday, the cobbler works for 8 hours each day, and on Friday, Cobbler obviates the need to go to the office on Friday. He only works from 8 am to 11 am. How many pairs of shoes can the cobbler mend in a week?	92	Attack success!
A cobbler can mend 3 pairs of shoes in an hour. From Monday to Thursday, the cobbler works for 8 hours each day, and on Friday, he only works from 8 am to 11 am. How many pairs of shoes can the cobbler mend in a week?	A cobbler can mend 3 pairs of shoes in an hour. From Monday to Thursday, the cobbler works 8 hours each day; on Friday, he only works from 8 am to 11 am. Cobbler obviates the need to go to the office on Friday. How many pairs of shoes can the cobbler mend in a week?	88	Attack success!
A cobbler can mend 3 pairs of shoes in an hour. From Monday to Thursday, the cobbler works for 8 hours each day, and on Friday, he only works from 8 am to 11 am. How many pairs of shoes can the cobbler mend in a week?	A cobbler can mend 3 pairs of shoes in an hour. From Monday to Thursday, the cobbler works for 8 hours each day, and on Friday, he only works from 8 am to 11 am. How many pairs of shoes can the cobbler mend in a week? Cobbler obviates the need to go to the office on Friday.	96	Attack success!

GSM8K. It can be seen that when $\epsilon=0.2$ and $\gamma=0.5$, the ASR value of the attack is the highest. In fact, this combination of parameters works very well when we apply it to other datasets.

Table 5. Impact of parameter pair (ϵ, γ) on the performance of the attack algorithm: when $(\epsilon, \gamma) = (0.2, 0.5)$, the attack algorithm achieves the highest performance.

ϵ	γ	Clean Acc	Attack Acc	ASR	ϵ	γ	Clean Acc	Attack Acc	ASR
0.2	0.1	72.12	53.50	25.82	0.9	0.5	72.12	56.57	21.56
0.2	0.2	72.12	59.72	29.67	0.8	0.5	72.12	54.50	24.43
0.2	0.3	72.12	45.74	36.58	0.7	0.5	72.12	51.41	28.71
0.2	0.4	72.12	41.90	41.89	0.6	0.5	72.12	50.82	29.53
0.2*	0.5*	72.12	38.69	46.35	0.5	0.5	72.12	46.76	35.17
0.2	0.6	72.12	41.71	42.64	0.4	0.5	72.12	45.16	37.38
0.2	0.7	72.12	41.84	41.98	0.3	0.5	72.12	42.17	41.53
0.2	0.8	72.12	44.48	38.32	0.2*	0.5*	72.12	38.69	46.35
0.2	0.9	72.12	43.53	39.64	0.1	0.5	72.12	37.80	47.60