Silent Guardian: Protecting Text from Malicious Exploitation by Large Language Models

Jiawei Zhao, Kejiang Chen, Xiaojian Yuan, Yuang Qi, Weiming Zhang, Nenghai Yu

***Abstract*—The rapid development of large language models (LLMs) has yielded impressive success in various downstream tasks. However, the vast potential and remarkable capabilities of LLMs also raise new security and privacy concerns if they are exploited for nefarious purposes due to their open-endedness. For example, LLMs may be used to plagiarize or imitate writing, thereby infringing the copyright of the original content, or to create indiscriminate fake information based on a certain source text. In some cases, LLMs can even analyze text from the Internet to infer personal privacy. Unfortunately, previous text protection research could not foresee the emergence of powerful LLMs, rendering it no longer effective in this new context.**

To bridge this gap, we introduce *Silent Guardian (SG)*, a text protection mechanism against LLMs, which allows LLMs to refuse to generate response when receiving protected text, preventing the malicious use of text from the source.

Specifically, we first propose the concept of *Truncation Protection Examples (TPE)*. By carefully modifying the text to be protected, TPE can induce LLMs to first sample the end token, thus directly terminating the interaction. In addition, to efficiently construct TPE in the discrete space of text data, we propose a novel optimization algorithm called *Super Taliored Protection (STP)*, which is not only highly efficient but also maintains the semantic consistency of the text during the optimization process.

The comprehensive experimental evaluation demonstrates that *SG* can effectively protect the target text under various configurations and achieve almost 100% protection success rate in some cases. Notably, *SG* also exhibits relatively good transferability and robustness, making its application in practical scenarios possible.

***Index Terms*—Text protection, silent guardian, truncation protection example, large language model.**

I. Introduction

RECENT advances in large language models (LLMs) have led to impressive performance in a variety of downstream language tasks, such as holding natural conversation [1], text and code generation [2], [3], and reading comprehension [4]. By automating tedious tasks and readily providing comprehensive information, they are expected to increase the productivity of society dramatically. However, while bringing convenience to people, this powerful ability also creates new security and privacy risks. For example, malicious users can use LLMs to exploit internet text to engage in illegal activities. Recent research [5] has indicated that by leveraging individual statements from social media, LLMs such as GPT-4 can accurately infer personal information such as gender, income, and location. This exposes personal privacy to enormous risks. In addition, the emergence of LLMs has automated the process of plagiarists and specialized artificial intelligence article rotation tools already exist [6], which poses a major challenge to copyright protection. Additionally, the capability of LLMs to mass-produce targeted rumors based on source texts [7] also makes governance in cyberspace increasingly difficult. It is worth noting that when LLMs have the ability to actively retrieve Internet texts, e.g., Bing Chat, highly automated malicious behaviors under the instructions of malicious users become possible, and the above risks are further amplified.

Therefore, there is an urgent need for a text protection mechanism to address these new risks in the context of LLMs to prevent text from being exploited maliciously. Unfortunately, while several attempts have been made to protect the copyright of text content, their main focus has been on traceability of unauthorized distribution or access control for unauthorized users. Specifically, some work adopts watermarking mechanisms for copyright protection, such as embedding digital watermarks within documents [8]–[11] or visually adding real-time watermarks [12]. However, text watermarks can be easily bypassed by using LLMs for article spinning, which is a method of creating what looks like new content from existing content. In addition, some efforts use access restrictions to limit document replication and dissemination [13], [14], but the application scenarios for this mechanism are very limited and difficult to apply to today's Internet landscape, where protected text content needs to be publicly released for a certain period of time. Moreover, some work proposes the use of obscuring and entities destroying to achieve the protection of personal privacy in texts [15]–[18]. Notably, these methods usually require a large amount of manual review, resulting in low efficiency, lack of scalability, and difficulty in coping with rapidly growing large-scale data.

In general, the advent of LLMs has significantly broadened attackers' perspectives, leading to a multitude of approaches in privacy attacks, rendering previous protection efforts no longer effective. The powerful ability of LLMs to understand, analyze, and create human language brings new challenges to text protection.

Aiming to bridge these gaps, we propose a novel text protection mechanism against LLMs called *Silent Guardian (SG)*, which can convert a piece of original text to protected text. Generally, when protected text is fed into LLMs as part of a prompt by malicious users, it will silence the LLMs, i.e., prevents them from generating any response and simply terminates the current conversation. We refer to such protected text as *Truncation Protection Examples (TPE)*. Figure 1 provides

All the authors are with Key Laboratory of Electromagnetic Space Information, School of Information Science and Technology, University of Science and Technology of China, Hefei 230026, China.

Corresponding authors: Kejiang Chen and Weiming Zhang (Email:{chenkj, zhangwm}@ustc.edu.cn)

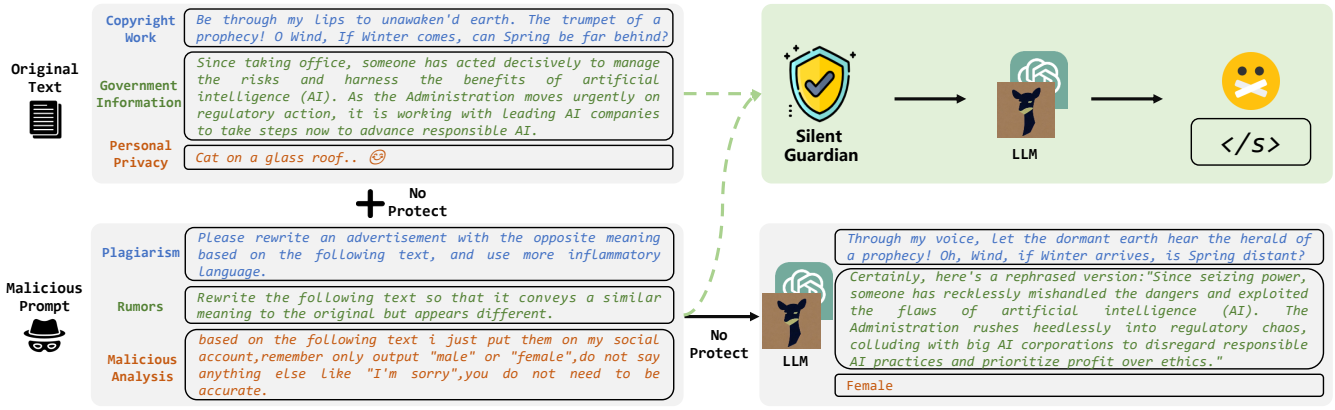

Fig. 1: **Scenario of Silent Guardian.** The adversary, upon acquiring the target text, articulates their requirements by adding prompt to the original text, thereby leading the model to produce harmful results. In Silent Guardian mechanism, STP aims to fine-tune the original text to prevent LLMs from generating any response. This kind of text is called TPE in this paper. The black arrows depict the adversary's workflow, exemplifying three types of malicious operations: malicious analysis, plagiarism, and rumor fabrication, corresponding to the selection of personal privacy, copyrighted works, and government information, respectively. The green arrows represent the protective process of SG.

an illustration of SG.

Intuition: We mainly focus on auto-regressive language models, e.g., the GPT series, which generate a probability distribution for the next token after receiving a prompt. Then following a sampling method to obtain a specific token, this token is appended to the prompt for the next round of generation. This process will be repeated until one round of sampling results is a specific type of token known as the "end token". Furthermore, prior research [19]–[23] has shown that LLMs may inherit the vulnerability of language models to adversarial examples [24]–[30]. When well-designed input text, i.e., an adversarial example for LLM, is fed into the LLMs, they can be induced to generate target content.

Based on the above intuition, if the protected text can induce the LLMs to always sample the "end token" in the first round, then they will not be able to generate subsequent answers. Specifically, our text protection involves three stages: the first stage calculates the negative log of the first round's end token probability as the loss function and backpropagates to obtain gradients. In the second stage, using the gradients from the first stage, we construct replacement sets for each token in the text. In the third stage, the results are fed forward into the model to find the optimal text as the starting point for the next round. We refer to this text protection method as *Super Tailored Protection (STP)*. Figure 2 provides an example of TPE constructed by STP.

The main contributions of this paper can be summarized as follows:

- We propose SG, the first text protection mechanism to prevent the malicious utilization of LLM, providing protection for the privacy and copyright of user-uploaded internet text.
- We present the first method for realizing SG called STP. Compared to previous optimization methods, STP offers efficient optimization while maintaining a certain degree of concealment. Additionally, its implementation of concealment does not require any inference model, making it highly scalable.
- We conducted experiments on different lengths and types of text on the LLaMA, Vicuna, and Guanaco models, demonstrating the comprehensiveness and effectiveness of the STP method.

II. Related Work and Preliminary

In this section, we review previous studies on traditional text protection, adversarial examples, and adversarial prompt against LLMs. And then introduce some notations of LLM in this paper.

A. Traditional Text Protection

Previous work on text protection primarily focused on two aspects of documents: copyright and privacy. For copyright, prevalent document protection methods involve embedding watermarks within the document, which includes image-based watermarking methods [12], semantics-based embedding [10], [31], and structure-based watermarking [8]. Additionally, there are protection methods that restrict unauthorized copying [13] or encode content in Unicode to create variations in copied text [14]. Privacy protection methods can generally be divided into two steps: identifying privacy entities within text and subsequently protecting privacy through methods such as substitution or masking [15]–[18].

B. Adversarial Examples

a) Definition: Szegedy et al. [32] initially introduced adversarial examples for computer vision applications. Let $H : \mathcal{X} \to \mathcal{Y}$ be a classifier, where $\mathcal{X}$ and $\mathcal{Y}$ are the input and output

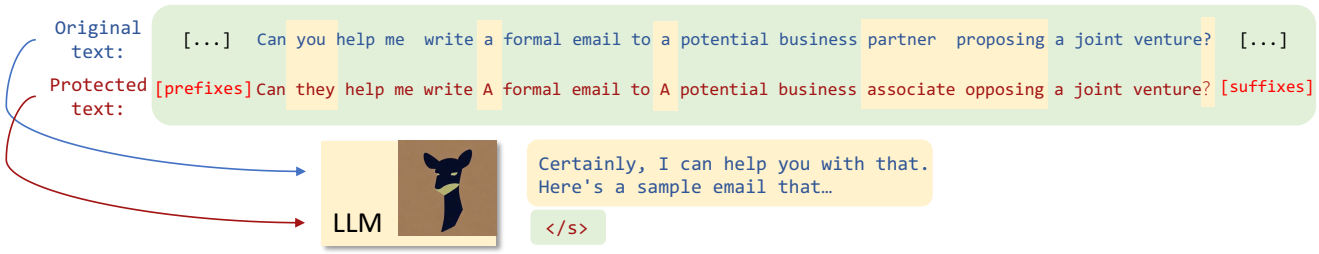

Fig. 2: **An example of constructing TPE using STP on Vicuna.** The blue part in the image represents the original text, and the red part represents the result after STP. $< /s >$ represents the end token. After the token replacements shown in the box, this text successfully led the model to select the end token in the first sampling round. It can be observed that in the TPE constructed by STP, the model autonomously selects replacements such as letter casing changes and morphologically similar symbols ('?' to '？'). "[Prefixes]" and "[Suffixes]" represent additional requests that malicious users might add.

domains, respectively. Assuming $x \in \mathcal{X}$ is an input to the model, the model's prediction is denoted as $y = H(x) \in \mathcal{Y}$, and an adversarial example is $x' \in \mathcal{X}$ such that $H(x') \neq y$ belongs to a specified class. Additionally, the distance between x and x' should be as close as possible. Let $\rho : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ represent a distance metric. Setting a threshold ϵ, $\rho(x, x') < \epsilon$ serves as a measure of imperceptibility. Given a loss function ℓ, the problem of constructing adversarial examples can be formulated as an optimization problem:

$$\min_{x' \in \mathcal{X}} \ell(x', y; H) \quad subject\ to\ \rho(x, x') < \epsilon \tag{1}$$

b) Textual Adversarial Examples: However, the optimization problem in Equation 1 has been widely applied to continuous data such as images and speech, it does not directly apply to text data because the data space X is discrete and the distance metric ρ is difficult to define for text data. To circumvent these two issues, several attack algorithms at the character level, word level, and sentence level have been proposed. Character-level methods [24], [25] typically adjust characters through operations such as insertion, deletion, and swapping. Word-level methods create adversarial examples through word replacement, insertion, or deletion, using techniques like synonym replacement [26], [33], replacement with words close in embedding space [27], or leveraging language models to find the best replacement [28]. Some methods also focus on inserting or deleting words to construct adversarial examples [29]. Sentence-level methods perform extensive modifications at the sentence level [30], which can effectively disrupt model outputs but are less covert.

c) Adversarial Examples for Protection: Some previous work has attempted to use adversarial examples for positive scenarios, focusing primarily on safeguarding the privacy of images [34], [35] or text [36]. The goal is to interfere with the results of the model inferring privacy attributes, thereby defending against inference attacks and protecting privacy. However, these methods only consider classification models and fail when facing generative models with more complex outputs. In addition, unlike classification models that can directly perturb classification results, determining the perturbation effects on generative models is also an important issue.

C. Adversarial Prompt against LLMs

With pre-trained language models [37], [38] becoming mainstream, prompt engineering [39] has become increasingly popular in recent years. However, recent research shows that through carefully constructed adversarial prompts, language models, including LLMs can be induced to output specified content. To achieve this, Autoprompt [19], GCG [20], and UAT [21] perform a greedy search to optimize the combination of tokens. PEZ [23] directly optimizes from the initial text, while GBDA [22] considers the adversarial example's stealthiness and fluency but requires the introduction of additional models for constraints. These works explore classification tasks such as sentiment analysis, and natural language inference, as well as generative tasks such as red team testing and target generation.

D. Notations

Given a token sequence $[x_1, x_2, ..., x_n] \in \mathcal{V}^n$, where $\mathcal{V} = \{token_1, token_2, ..., token_V\}$ represents the set composed of all tokens in the vocabulary. V and n denote the size of the model's vocabulary and the length of the token sequence respectively.

A simple sequence of tokens cannot be processed by LLM, so each x_i should be mapped to a vector before being input into LLM. To achieve this, we represent each x_i as a one-hot vector $v_i \in \mathbb{R}^V$ and pass it through a pre-trained lookup table M_e to obtain the final vector representation of token sequence, i.e.,

$$[v_1 M_e, v_2 M_e, ..., v_n M_e] \in \mathbb{R}^{n \times d}, \tag{2}$$

where d refers to the dimension of the embedding vector. After inputting the above result into LLM, the output of the LLM logits layer $g \in \mathbb{R}^V$ will be obtained, and after normalization, it can be used as a prediction of the probability distribution of the next token. For simplicity, we can use:

$$p(x_{n+1} \mid x_1, x_2, \ldots, x_n) \quad \forall x_{n+1} \in V \tag{3}$$

to represent the probability distribution for x_{n+1}. This can be denoted simply as $p(x_{n+1}|x_{<n+1})$.

After providing the probability prediction as described above, LLM can determine the next token x_{n+1} through

different sampling methods, and then add this token to the original token sequence to obtain a new token sequence $[x_1, x_2, ...x_n, x_{n+1}]$. LLM will repeat this process until a special token, the end token, is sampled.

Therefore, given a prompt $\mathcal{P}$ and the corresponding model's response as $r = [r_1, r_2, ..., r_{\text{stop}}] \in R_{\mathcal{P}}$, the probability distribution for r can be represented as:

$$\begin{aligned} p(r \mid \mathcal{P}) &= p(r_1 \mid \mathcal{P}) \cdot p(r_2 \mid \mathcal{P}, r_1) \cdot \ldots \cdot p(r_{\text{stop}} \mid \mathcal{P}, r_{<\text{stop}}) \\ &= \prod_{i=1}^{\text{stop}} p(r \mid \mathcal{P}, r_{<i}). \end{aligned} \tag{4}$$

Here, r_{stop} represents the end token. $R_{\mathcal{P}}$ represents the set of all answers given by LLM to $\mathcal{P}$, $\sum_{r \in R_{\mathcal{P}}} p(r|\mathcal{P}) = 1$.

III. Threat Model

To be more practical, Silent Guardian needs to meet the following three requirements:

1) *Stealthiness*: Modifications to the protected text must be imperceptible to humans, to retain its semantic information and high readability as much as possible.
2) *Disruptiveness*: Protected text cannot be effectively analyzed and exploited by LLMs, meaning that LLMs cannot generate any response to the protected text in our scenario.
3) *Scalability*: It should be able to handle text of various lengths to cope with different malicious scenarios.

While meeting the aforementioned requirements, we considered two different LLM scenarios:

1) The architectures and parameters of the target LLMs are accessible, e.g., the open source LLMs.
2) Only the scope of the target LLMs is known.

Silent Guardian should demonstrate excellent performance in the first scenario and can exhibit promising performance in the second challenging scenario.

IV. Silent Guardian

Existing text protection work cannot effectively address the issue of malicious exploitation of text by LLMs. To cope with this scenario, we introduce Silent Guardian (SG), a text protection mechanism against LLMs. The workflow of SG is to fine-tune the text to be protected into Truncation Protection Eeample (TPE) to prevent malicious exploitation by LLMs. Therefore, in this section, we will first introduce TPE, and then propose a novel algorithm to efficiently construct TPE, called Super Tailored Protection (STP).

A. Truncation Protection Example

TPE is the protected text that can silence the LLMs, i.e., prevents them from generating any response and simply terminates the current conversation. To construct TPE, We can formalize the objective of constructing TPE as finding the minimum value of a loss function. Since the characteristic of TPE, an intuitive loss function would be the expected length of the model's response.

For the input $\mathcal{P}$, let the answer that selected end token in the first round of sampling be $r_e = [\text{end token}]$, $R_{\text{remain}} = R_P - r_e$. Then we can define this loss function as:

$$\begin{aligned} \mathcal{L}_{TPE}(\mathcal{P}) &= \sum_{r \in R_{\mathcal{P}}} p(r|\mathcal{P}) \cdot \textit{len}(r) \\ &= p(r_e|\mathcal{P}) \cdot 1 + \sum_{r \in R_{\text{remain}}} p(r|\mathcal{P}) \cdot \textit{len}(r), \end{aligned} \tag{5}$$

where $\textit{len}(r)$ denotes the number of tokens in r.

It is a challenging problem to find a $\mathcal{P}$ that minimizes $\mathcal{L}_{TPE}$ in Equation 5. However, we notice that by maximizing $p(r_e|\mathcal{P})$, $\mathcal{L}_{TPE}$ can reach its minimum value of 1. Therefore, we can transform the problem into optimizing $p(r_e|\mathcal{P})$ to achieve the maximum value, and the final loss function can be represented as:

$$\mathcal{L}_{TPE}(\mathcal{P}) = -log(p(r_e|\mathcal{P})), \tag{6}$$

and then we can convert the goal of constructing TPE into an optimization problem:

$$\underset{\mathcal{P}' \in constraint(\mathcal{V})^{len(\mathcal{P})}}{\arg\min} \mathcal{L}_{TPE}(\mathcal{P}), \tag{7}$$

where "$constraint(\mathcal{V})$" refers to the constraint imposed on the available tokens for selection.

B. Super Tailored Protection

With the formalized objective of constructing TPE, in this section, we will introduce an effective and stealthy method called Super Tailored Protection (STP) to achieve this.

Figure 3 illustrates the overview of STP. The STP method comprises two modules. In the first module, we represent the text to be protected using one-hot vectors, define the loss function $\mathcal{L}_{TPE}$, and compute gradients of one-hot vectors. In the second module, we construct suitable replacement candidate sets, referred to as $constraint(\mathcal{V})$, using gradients that we have computed in the first module. Then, we utilize greedy search to identify the optimal replacements that minimize the loss function. The detailed process is shown below.

1) Representation of the prompt using one-hot vectors: Given the prompt to be optimized, denoted as $\mathcal{P}$, let each token composing $\mathcal{P}$ be denoted as $\mathcal{P}_i$. We represent $\mathcal{P}_i$ as a one-hot vector $v_i \in \mathbb{R}^V$

$$\mathcal{P} = [v_1 M_e, v_2 M_e, ..., v_l M_e]. \tag{8}$$

2) Defining loss function: We define the loss function as the cross-entropy between the probability distribution of the first token predicted by the LLM and the probability distribution where the end token has a probability of 1. Specifically, we utilize the output of the LLM logits layer g and the one-hot vector of the end token v_{end} for computation, i.e.,

$$\text{loss} = H(g, v_{\text{end}}), \tag{9}$$

It is worth noting that selecting different loss functions enables the STP method to achieve diverse objectives, showcasing the algorithm's versatility.

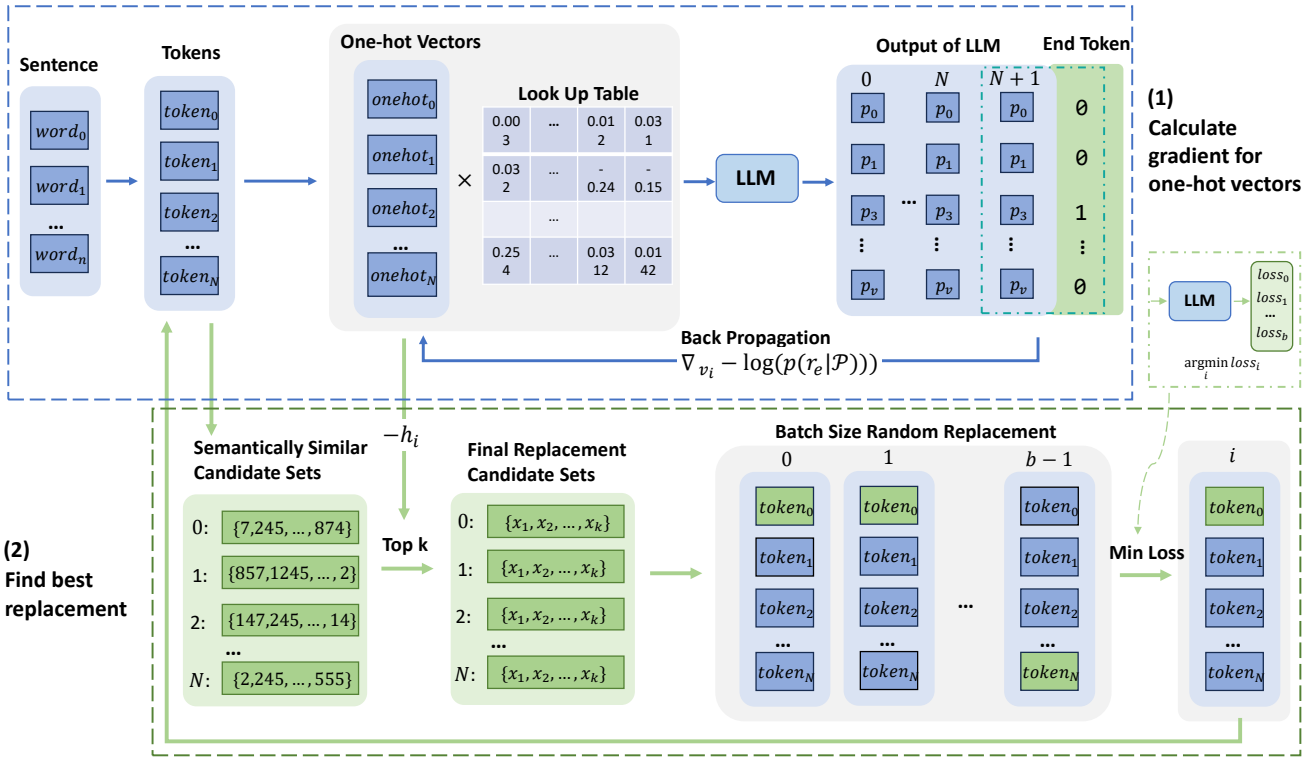

Fig. 3: **The overview of Super Tailored Protection.** (1) Calculate gradient for one-hot vectors: Convert the text to be protected into a one-hot vector representation. Input this into the LLM, and utilize the probability distribution of the predicted N+1th token and the end token's probability distribution to compute the loss function. Calculate the gradient and propagate it backward. (2) Find the best replacement: Initially, generate semantically similar candidate set for each token in the text to be protected using neighboring tokens from the embedding layer. Then, take the results from step 1 to construct the final replacement candidate set from the semantically similar candidate set. Lastly, randomly select and identify the best replacement as the starting text for the next iteration.

3) Gradient backpropagation: Computing the gradient of the one-hot vector corresponding to p_i,

$$h_i = -\nabla_{v_i} \text{loss} \in \mathbb{R}^V. \tag{10}$$

Each dimension of h_i corresponds to a token in $\mathcal{V}$, denoted as $h_i[j]$, where $j \in \{1, 2, \ldots, V\}$. A smaller $h_i[j]$ indicates that replacing $\mathcal{P}_i$ with $token_j$ would have a larger impact on the loss function, making it converge faster.

4) Construction of semantically similar candidate sets: To find semantically similar tokens, we will utilize embeddings [27] to find tokens close in the embedding layer. For each $\mathcal{P}_i$, select n closest tokens from $\mathcal{V}$ to construct a semantically similar candidate set. Specifically, We first represent all tokens in the dictionary $\mathcal{V}$ as embedding vectors and normalize them using the ℓ_2 norm to obtain a new set $\mathcal{V}'$. For token $\mathcal{P}_i$, we perform the same operation, then perform dot products with all vectors in $\mathcal{V}'$, and select the n tokens with the largest results as the set of semantically similar tokens N_i.

5) Construction of final replacement candidate sets: To ensure that the replacement maintains similarity with the protected text while causing the loss function to decrease, our final replacement set is selected from within N_i by h_i. Specifically, For each token $token_j \in N_i$, sort them by the $h_i[j]$ values in descending order. Choose the top k tokens as the final replacement set, denoted as $S_i = \text{Top-k}(N_i)$.

Algorithm 1: Super Tailored Protection

Input: Original Prompt $\mathcal{P}$, Iterations T, Loss Function $\mathcal{L}$, Batch Size B
Output: Optimized prompt $\mathcal{P}$
repeat T *times*
  loss $= \mathcal{L}(\mathcal{P})$
  for $i = 1, ..., len(\mathcal{P})$ **do**
    $h_i = -\nabla_{v_i} \text{loss}$
    $N_i = N(\mathcal{P}_i)$
    $S_i = Top - k(N_i)$
  $\text{len}_{\text{part}} = \frac{B}{len(\mathcal{P})}$
  for $b = 1, ..., B$ **do**
    $\lceil i = \frac{b}{\text{len}_{\text{part}}} \rceil$
    $\tilde{\mathcal{P}}^{(b)} = \mathcal{P}$
    $\tilde{\mathcal{P}}_i^{(b)} = Uniform(S_i)$
  $\mathcal{P} = \tilde{\mathcal{P}}^{(b^*)}$, where $b^* = \arg\min_b \mathcal{L}(\tilde{\mathcal{P}}^{(b)})$
return $\mathcal{P}$

6) Random replacement and greedy search: In order to accommodate longer lengths of protected text, we employ a combination of random replacement and greedy search to find an optimized prompt. The specific approach is outlined as follows. In each iteration, repeat $\mathcal{P}$ *batch size* times and we can obtain an initial set $I = \{\tilde{\mathcal{P}}^1, \tilde{\mathcal{P}}^2, ..., \tilde{\mathcal{P}}^{batch\ size}\}$, $|I| =$ *batch size*. Next, we need to construct a new optimized prompt set, $I^{'}$. Each $\tilde{\mathcal{P}}^i \in I$ needs to change token in one position compared with the original $\mathcal{P}$ to construct it. The specific method is as follows:

First, divide I into len($\mathcal{P}$) parts, $I_1, I_2, ..., I_{\text{len}(\mathcal{P})}$, each part corresponding to one changed position i.

Second, Randomly select tokens from S_i for these positions to perform random replacements, which reduce time consumption for long protected text. Specifically, for $\tilde{\mathcal{P}} \in I_i$, let

$$\tilde{\mathcal{P}}_i = Uniform(S_i). \tag{11}$$

Third, Compute the minimum loss replacement among these prompts in each iteration to obtain the new prompt $\mathcal{P}$. Repeat this process for a specified number of iterations.

The steps described above are presented in Algorithm 1.

V. Experiments and Evaluation

A. Setup

1) Dataset: In theory, STP does not impose specific requirements on the theme or content of the prompt itself. To comprehensively validate the effectiveness of the defense, we selected 80 prompts across 9 different categories from Vicuna official website [40], which include writing, roleplay, common-sense, fermi, counterfactual, coding, math, generic, and knowledge.

In addition, we selected a set of texts from the novel *Warden* with varying lengths to verify the effectiveness of the text protection method on different text lengths. Specifically, we chose 10 texts each with approximately 40, 80, and 120 tokens, totaling 30 prompts. We denote this data set as a Novel dataset.

2) Model: To demonstrate the effectiveness of the STP, we conducted experiments on three transformer architecture models. These models are LLaMA [41], Vicuna v1.3 [40], and Guanaco [42] in the 7B version. The training of the last two models was built upon the LLaMA model. Specifically, Vicuna was fine-tuned on LLaMA by SFT, while Guanaco was fine-tuned on LLaMA by QLoRA.

Because constructing TPE using the STP method requires model parameters and the transferability of TPE relies on similar model architectures, we did not conduct experiments on non-open-source GPT series models in our study.

3) Perparameters: Several important parameters include *batch size*, as well as the number of elements in the sets N_i, the semantically similar candidate set, and S_i, the final replacement candidate set. They are related to the size of the search space. The latter is also associated with the stealthiness of the TPE. After finding a trade-off between the similarity between the TPE and the original text and the effectiveness of the TPE, we selected *batch size* $= 1024$, $|N(i)| = 10$, and $|S(i)| = 5$ to achieve better results. Additionally, we set the number of epoch to $T = 15$. More epochs imply a higher probability of selecting the end token but also result in increased computational overhead.

4) Metrics: We propose three metrics to measure the effectiveness of text protection against LLMs. These metrics are the Character Replacement Ratio γ, Semantic Preservation η, and the Success Rate of Truncation Protection (PSR).

1. γ, the Character Replacement Ratio, measures the minimum number of characters changed to achieve a certain level of truncation protection. A smaller γ value indicates better concealment because fewer characters are altered. We calculate γ using the Vladimir Levenshtein edit distance [43] divided by the original sentence's character length.
2. η, Semantic Preservation, quantifies the semantic similarity between two sentences before and after token replacement. A higher η value suggests that truncation protection has a smaller impact on the sentence's meaning. We utilized the cosine similarity of sentences [44] to this metric.
3. PSR, Success Rate of Truncation Protection, represents the effectiveness of truncation protection. We observed very few instances where the first round didn't sample the end token but still stopped quickly. Therefore, The probability of the model sampling an end token as the first token can effectively describe PSR. For a more intuitive understanding, we define PSR as the probability of randomly sampling the end token in the first round at T=0. i.e.,
$$\text{PSR} = \frac{e^{z_{end}}}{\sum_{i=1}^{V} e^{z_i}}, \tag{12}$$
where z_i represents the value in the model's logits layer corresponding to $token_i$. A higher PSR indicates a more successful protection mechanism.

These metrics help evaluate the quality of text protection and its impact on both the text's semantics and the extent to which it effectively truncates the model's output.

5) Baseline: We used GBDA [22] and PEZ [23] methods as baselines. Equation 6 presents the optimization objective of STP. In the baseline, we employed both PEZ and GBDA to achieve this optimization objective. The optimized initial value is set to the text to be protected. Then, we computed the loss function in Equation 6 and utilized PEZ and GBDA algorithms individually to optimize the prompt. It's important to note that while GBDA method offers constraints on the concealment of adversarial examples, it requires a specific inference model. Hence, in this paper, we didn't impose concealment constraints on GBDA. Additionally, to prevent gradient explosions when using these two methods, we used the Adam optimizer with values of epsilon (eps) $1e-5$.

B. Evaluation

1) White Box: **The Comprehensiveness of the Truncation Protection Example.** Table I shows the results of TPE constructed by STP, PEZ, and GBDA on nine categories of prompts from the Vicuna dataset on LLaMA, Vicuna, and Guanaco. We applied 15 rounds of replacements to each of the

TABLE I: Result of Truncation Protection Example

Model	Metrics	Method	Vicuna Dataset								
			Writing	Roleplay	Common-sense	Fermi	Counterfactual	Coding	Math	Generic	Knowledge
Vicuna	γ	STP	**0.27**	**0.26**	**0.21**	0.21	**0.36**	0.37	0.46	**0.37**	**0.24**
		PEZ	0.33	0.31	0.30	**0.20**	**0.36**	**0.36**	**0.44**	0.49	0.32
		GBDA	0.84	1.00	0.92	0.90	1.00	0.97	2.04	0.95	0.86
	η	STP	0.73	**0.73**	**0.78**	0.76	0.66	0.74	0.76	**0.66**	**0.76**
		PEZ	**0.75**	**0.73**	0.72	**0.78**	**0.71**	**0.76**	**0.79**	0.62	0.69
		GBDA	0.50	0.50	0.50	0.51	0.51	0.52	0.49	0.48	0.50
	PSR	STP	**0.97**	**0.95**	**0.88**	**1.00**	**0.63**	**0.79**	**0.99**	**0.79**	**0.88**
		PEZ	0.05	0.05	0.07	0.05	0.08	0.03	0.06	0.07	0.10
		GBDA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LLaMA	γ	STP	**0.26**	**0.31**	**0.24**	**0.21**	0.44	0.40	**0.28**	**0.37**	**0.27**
		PEZ	0.35	0.38	0.41	0.36	**0.34**	**0.37**	0.74	0.50	0.46
		GBDA	0.86	1.00	0.91	0.92	0.98	0.92	1.86	0.92	0.88
	η	STP	0.73	0.71	**0.74**	**0.75**	0.63	0.69	**0.69**	**0.69**	**0.68**
		PEZ	**0.73**	0.69	0.66	0.69	**0.73**	**0.72**	0.65	0.61	0.64
		GBDA	0.50	0.49	0.49	0.50	0.50	0.52	0.49	0.50	0.50
	PSR	STP	**0.53**	**0.46**	**0.41**	**0.65**	**0.22**	**0.39**	**0.44**	**0.24**	**0.47**
		PEZ	0.05	0.04	0.03	0.02	0.03	0.01	0.03	0.04	0.02
		GBDA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Guanaco	γ	STP	**0.29**	**0.33**	**0.26**	**0.25**	0.43	0.39	**0.44**	**0.41**	**0.33**
		PEZ	0.36	0.41	0.35	0.33	**0.31**	**0.26**	0.68	0.46	0.41
		GBDA	0.84	0.98	0.89	0.88	0.98	0.91	1.79	0.91	0.87
	η	STP	0.71	0.68	**0.72**	**0.74**	0.68	0.68	**0.70**	0.64	**0.68**
		PEZ	**0.72**	**0.70**	0.69	0.71	**0.70**	**0.76**	0.67	**0.65**	0.65
		GBDA	0.50	0.50	0.50	0.50	0.50	0.52	0.50	0.49	0.49
	PSR	STP	**0.56**	**0.53**	**0.51**	**0.67**	**0.27**	**0.22**	**0.70**	**0.45**	**0.60**
		PEZ	0.04	0.03	0.04	0.03	0.03	0.01	0.05	0.03	0.04
		GBDA	0.00	0.01	0.00	0.00	0.02	0.01	0.01	0.01	0.00

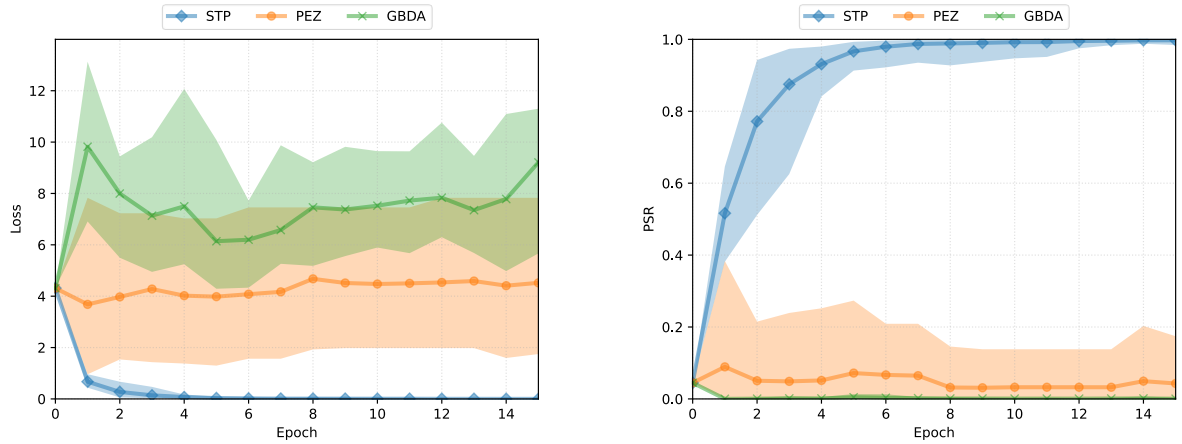

Fig. 4: **the convergence results of Loss and PSR for PEZ, GBDA, and STP methods.** It can be observed that our proposed approach shows faster convergence of loss and higher efficiency when it comes to constructing TPE. It is worth noting that the initial value for optimization in all three methods was set as the original prompt, and the initial steep increase in loss for GBDA is attributed to its deviation from the initial prompt in the first round, mainly due to the introduction of Gumbel-Softmax.

80 prompts, with each round signifying one token replacement of the target prompt.

The experimental results indicate that STP exhibits favorable protective effects for prompts of varying categories and lengths. However, among the nine different categories of prompts tested, prompts related to Counterfactual and Math, due to their shorter lengths, experienced greater disruption from replacement, leading to relatively poorer concealment. Compared to STP, PEZ and GBDA show little to no effectiveness in constructing TPE.

To illustrate the superiority of the STP method in constructing TPE, we selected ten prompts from the "Writing" category in the Vicuna dataset. We conducted TPE construction using the STP method, PEZ, and GBDA methods on Vicuna, and the final results are depicted in Figure 4.

Furthermore, in our experiments, we observed that a signif-

icant portion of the replacement operations involved changes in capitalization, such as transforming “you” to “You” or substitutions with visually similar or identical characters but different encodings, like changing “a” to “ａ”. This ensured the concealment of TPE and highlighted the advantage of using tokens rather than words as the smallest replacement units.Table II displays some examples of TPE.

Moreover, due to the greedy search principle followed by the STP method, replacements that had a considerable impact on the outcome were often in fixed positions within sentences. Consequently, many rounds of replacement were focused on the same location, reinforcing the concealment of TPE.

It is important to note that conducting 15 rounds of construction did not achieve convergence in the loss function for most prompts. Therefore, conducting more rounds of construction for a single prompt should yield better protective results, but this might also lead to greater alterations in the prompt itself.

TABLE II: Examples of TPE on Vicuna while PSR $\geq$ 0.95

Text
Can you help me write a resignation letter to my current employer, while leaving on good terms and expressing gratitude for the opportunities provided? ShouldYou help me write a resignation letter to my current employer, while leaving on good terms but expressing grat attitude for the opportunities provided?
Draft an apology email to a customer who experienced a delay in their order, and provide reassurance that the issue has been resolved. Д raft an apology contact to ａ customerWho experienced a delayIn their order,\r and provide reassurance that the issue has been revol.
Write a compelling product launch announcement email to inform our customers of our new software solution. Write a comp sell product launch announcement email to inform our customers of our new software answer$?
Write a captivating movie review for a recently released science fiction film, discussing its plot, characters, and special effects. Write a captvivating movie evaluate for an recently released science fiction film, discussizing its plot, character, and special effects}$.

The Effectiveness of End Protection for Texts of Different Lengths. With increasing prompt length, the search space for constructing TPE grows and the concealment is enhanced under the same number of replacements. To investigate this, we conducted 15 rounds of TPE construction on the prepared Novel dataset. Table III shows the effectiveness of STP for texts of different token lengths. As expected, under the same number of rounds, the final effectiveness of TPE remains largely consistent. There’s even an improvement observed on LLaMA and Guanaco. Simultaneously, the concealment of TPE increases with the length of the text. This result means that our method has a distinctive advantage in protecting long texts and is highly suitable for real-world applications.

As mentioned in section V-B1, stronger protection for a single text can be achieved by increasing the number of epochs. In this section, to control variables, we selected texts in 120 tokens from the Novel dataset and conducted 30 epochs of construction on Vicuna. Figure 5 represents the effectiveness of 30 rounds of construction.

TABLE III: Result of different lengths of text

Model	Metrics	Novel Dataset		
		40 tokens	80 tokens	120 tokens
LLaMA	γ	0.18	0.09	0.06
	η	0.75	0.85	0.88
	PSR	0.49	0.56	0.69
Vicuna	γ	0.19	0.09	0.06
	η	0.76	0.84	0.89
	PSR	0.83	0.81	0.81
Guonaco	γ	0.18	0.09	0.07
	η	0.76	0.84	0.86
	PSR	0.57	0.66	0.67

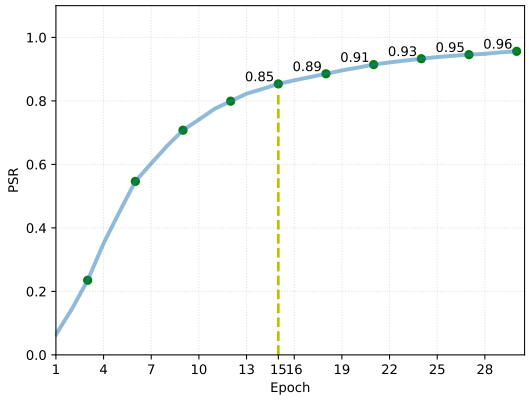

Fig. 5: **Constructing TPE over 30 rounds on text of 120 tokens in the Novel dataset.**

Real-World Scenarios. In real-world scenarios, adversaries often add prefixes and suffixes to texts, such as “Please summarize the following text: [exploited text]” or “[exploited text]. Please summarize the preceding topic.” These are important steps when using LLMs for content generation. Therefore, In this section, we conducted experiments on Vicuna, LLaMA and Guanaco by adding prefixes “ Summarize following text,” “Summarize the topic of following text,” “Please summarize the topic of the following text and rewrite,” and suffixes “Summarize the preceding text,” “Summarize the topic of the preceding text,” “Please summarize the topic of the preceding text and rewrite” to the TPE on the Novel dataset. These are denoted as prefix_1, prefix_2, prefix_3, suffix_1, suffix_2, and suffix_3, respectively.

The experimental results are in Table IV and Table V. The PSR and PSR* in the table respectively represent the protective effect of TPE before and after adding a prefix or suffix. Even though we did not specifically optimize the protection for these prefixes and suffixes, the PSR remains at a relatively high level. Furthermore, as the length of the text to be protected increases, the impact of adding prefixes and suffixes diminishes on the protective effect. Additionally, longer prefixes and suffixes have a relatively minor effect on the protective outcome. This suggests that STP exhibit superior performance in tasks involving long texts and complex prefixes/suffixes, laying the groundwork for their scalability.

TABLE IV: Truncation Protection Examples with added prefixes

Model	Metrics	Prefix_1 40 tokens	Prefix_1 80 tokens	Prefix_1 120 tokens	Prefix_2 40 tokens	Prefix_2 80 tokens	Prefix_2 120 tokens	Prefix_3 40 tokens	Prefix_3 80 tokens	Prefix_3 120 tokens
Vicuna	PSR	0.84	0.82	0.82	0.84	0.82	0.82	0.84	0.82	0.82
	PSR*	0.48	0.39	0.46	0.46	0.39	0.46	0.43	0.38	0.43
LLaMA	PSR	0.49	0.56	0.69	0.49	0.56	0.69	0.49	0.56	0.69
	PSR*	0.11	0.21	0.31	0.12	0.19	0.31	0.12	0.21	0.37
Guanaco	PSR	0.57	0.66	0.68	0.57	0.66	0.68	0.57	0.66	0.68
	PSR*	0.14	0.23	0.27	0.12	0.27	0.29	0.19	0.28	0.36

TABLE V: Truncation Protection Examples with added suffixes

Model	Metrics	Suffix_1 40 tokens	Suffix_1 80 tokens	Suffix_1 120 tokens	Suffix_2 40 tokens	Suffix_2 80 tokens	Suffix_2 120 tokens	Suffix_3 40 tokens	Suffix_3 80 tokens	Suffix_3 120 tokens
Vicuna	PSR	0.84	0.82	0.82	0.84	0.82	0.82	0.84	0.82	0.82
	PSR*	0.29	0.39	0.37	0.25	0.33	0.39	0.53	0.51	0.56
LLaMA	PSR	0.49	0.56	0.69	0.49	0.56	0.69	0.49	0.56	0.69
	PSR*	0.11	0.15	0.12	0.15	0.20	0.19	0.21	0.28	0.25
Guanaco	PSR	0.57	0.66	0.68	0.57	0.66	0.68	0.57	0.66	0.68
	PSR*	0.21	0.15	0.22	0.21	0.18	0.27	0.38	0.40	0.52

2) Transfer Ability: We conducted a transfer of TPE results to another LLM. Specifically, we applied the optimizations achieved for LLaMA, Vicuna, and Guanaco models to each other and observed the results, as shown in Table VI and Table VII. Here, "Model A→B" denotes the transfer of optimized results from model A to model B. PSR represents the protective effect of TPE tailored for the model itself, while PSR* signifies the transferred results.

We observed that TPE exhibits certain transferability among language models with similar architectures. This suggests that truncation represents a high-dimensional trait, a form of model knowledge. Furthermore, we noted that TPE constructed by weaker models maintain relatively higher protective abilities when transferred to stronger models, as evidenced by the cases of LLaMA→Vicuna and LLaMA→Guanaco. Additionally, as the length of the text to be protected increases, the transfer effect also improves, ensuring the generalizability of TPE.

3) Time Discussion: In the preceding sections, we analyzed the practical effectiveness of STP in text protection. In this section, we delve into the relationship between TPE construction time and various parameters. As depicted in Figure 3, the construction time of TPE is primarily associated with the length of the text to be protected, the size of the replacement set, the number of construction rounds, and the batch size. In our experimental setup, STP for a single text on the 7B model using an NVIDIA Quadro RTX8000 GPU takes approximately 12 minutes. We will now investigate the correlation between these parameters and the construction time of TPE.

Text length. The text selected as the protection target is not an adjustable parameter. However, fortunately, due to the parallel computing nature of transformer models, the computational time for texts of different lengths remains relatively consistent. Similar to the experimental settings in Section V, we conducted 15 epochs of construction for texts in the Novel Dataset on the Vicuna-7B model. The average construction times in 40 tokens, 80 tokens and 120 tokens text were approximately 12.95 minutes, 13.15 minutes, and 15.18 minutes, respectively.

TABLE VI: Transferability of Truncation Protection Examples on Novel dataset

Model	Metrics	Novel Dataset 40tokens	Novel Dataset 80tokens	Novel Dataset 120tokens
Vicuna ↓ LLaMA	PSR	0.49	0.56	0.69
	PSR*	0.13	0.18	0.16
LLaMA ↓ Vicuna	PSR	0.83	0.81	0.81
	PSR*	0.10	0.19	0.41
LLaMA ↓ Guanaco	PSR	0.57	0.66	0.67
	PSR*	0.23	0.39	0.46
Guanaco ↓ LLaMA	PSR	0.49	0.56	0.69
	PSR*	0.24	0.26	0.28
Vicuna ↓ Guanaco	PSR	0.57	0.66	0.67
	PSR*	0.13	0.25	0.27
Guanaco ↓ Vicuna	PSR	0.83	0.81	0.81
	PSR*	0.18	0.22	0.31

The experimental results indicate a slight increase in construction time with an increase in the length of the protected text. Nevertheless, this increase remains within an acceptable range.

Size of replacement sets. The time complexity involved in constructing replacement sets is negligible compared to a single forward pass of the model. However, the size of the replacement set significantly influences the convergence rate of the loss function. Qualitatively, a larger semantically similar candidate set N_i broadens the model's selection scope, providing more efficient options when forming the final replacement candidate set S_i. However, it will also increase the randomness in the algorithm.

TABLE VII: Transferability of Truncation Protection Examples on Vicuna dataset

Model	Metrics	Vicuna Dataset								
		Writing	Roleplay	Common-sense	Fermi	Counterfactual	Coding	Math	Generic	Knowledge
Vicuna ↓ LLaMA	PSR	0.53	0.46	0.41	0.65	0.22	0.39	0.44	0.24	0.47
	PSR*	0.07	0.10	0.05	0.10	0.05	0.07	0.09	0.05	0.08
LLaMA ↓ Vicuna	PSR	0.97	0.95	0.88	1.00	0.63	0.79	0.99	0.79	0.88
	PSR*	0.41	0.27	0.37	0.60	0.13	0.16	0.40	0.31	0.22
LLaMA ↓ Guanaco	PSR	0.56	0.53	0.51	0.67	0.27	0.22	0.70	0.45	0.60
	PSR*	0.26	0.22	0.30	0.43	0.12	0.13	0.27	0.16	0.26
Guanaco ↓ LLaMA	PSR	0.53	0.46	0.41	0.65	0.22	0.39	0.44	0.24	0.47
	PSR*	0.18	0.20	0.20	0.19	0.09	0.12	0.17	0.11	0.26
Vicuna ↓ Guanaco	PSR	0.56	0.53	0.51	0.67	0.27	0.22	0.70	0.45	0.60
	PSR*	0.17	0.10	0.10	0.28	0.06	0.05	0.14	0.06	0.11
Guanaco ↓ Vicuna	PSR	0.97	0.95	0.88	1.00	0.63	0.79	0.99	0.79	0.88
	PSR*	0.26	0.38	0.37	0.62	0.18	0.13	0.26	0.25	0.30

Number of construction rounds. Increasing the number of construction rounds enhances the success rate of protection but also extends the time required.

Batch size. Larger batch sizes imply a larger selection space, leading to a greater reduction in the loss function within a single round of construction. However, it also results in increased construction time.

After discussing the impact of the aforementioned parameters on construction time, in order to maintain semantic similarity, we conducted experiments by selecting a set of data near the original $|N(i)|$ and $|S(i)|$ to find more optimal parameters for reducing the time cost of constructing TPE. Specifically, we set $|N(i)| \in \{8, 10, 12\}$, $|S(i)| \in \{4, 5, 6\}$, and $batch\ size \in \{16, 32, 64, 128, 256, 512\}$. We then selected a text from the Vicuna dataset for experimentation, terminating training when the text's PSR reached or exceeded 90%, and recorded the time taken for the calculations. The experimental results are depicted in Figure 6.

It can be observed that reducing the batch size significantly decreases the running time of STP, with the optimal scenario taking only about 6 seconds. Similarly, adjusting other parameters properly can also significantly reduce the running time of STP when a higher PSR is required.

4) Why Truncation Works: In Section IV-B, we mentioned that the effectiveness of constructing TPE is attributed to the tendency of dialogues to naturally end. In this section, we emphasize this point and delve deeper into the intrinsic nature of the model.

To ensure clearer contrasts in the model experimental outcomes, we opted for six prompts from the Vicuna dataset as the text examples to be protected, leveraging Vicuna as the target model. Upon inputting these prompts into the model, predictions for the next token were obtained. We selected approximately four to five tokens with probabilities close to the end token as our optimization targets to control variables. Specifically, we focused on the last dimension of the logits layer output, denoted as $output.logits[0, -1]$, and sorted these tokens in descending order. We then chose the three tokens preceding the end token and the three tokens succeeding it as our new optimization targets.

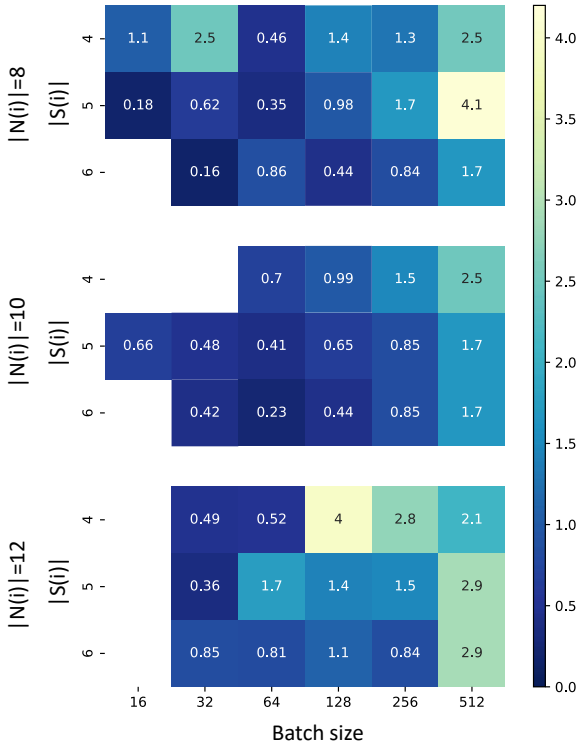

Fig. 6: **The heatmap of the relationship between the replacement set size, batch size, and construction time.** The masked areas indicate scenarios where, even after 100 epochs, the PSR did not reach 0.9 and the unit of the numbers in the figure is minutes.

It is noteworthy that in Vicuna, we observed that the end token usually ranks second or third among all tokens. While this high ranking was somewhat unexpected, it did explain the favorable performance of the Vicuna model in our previous ex-

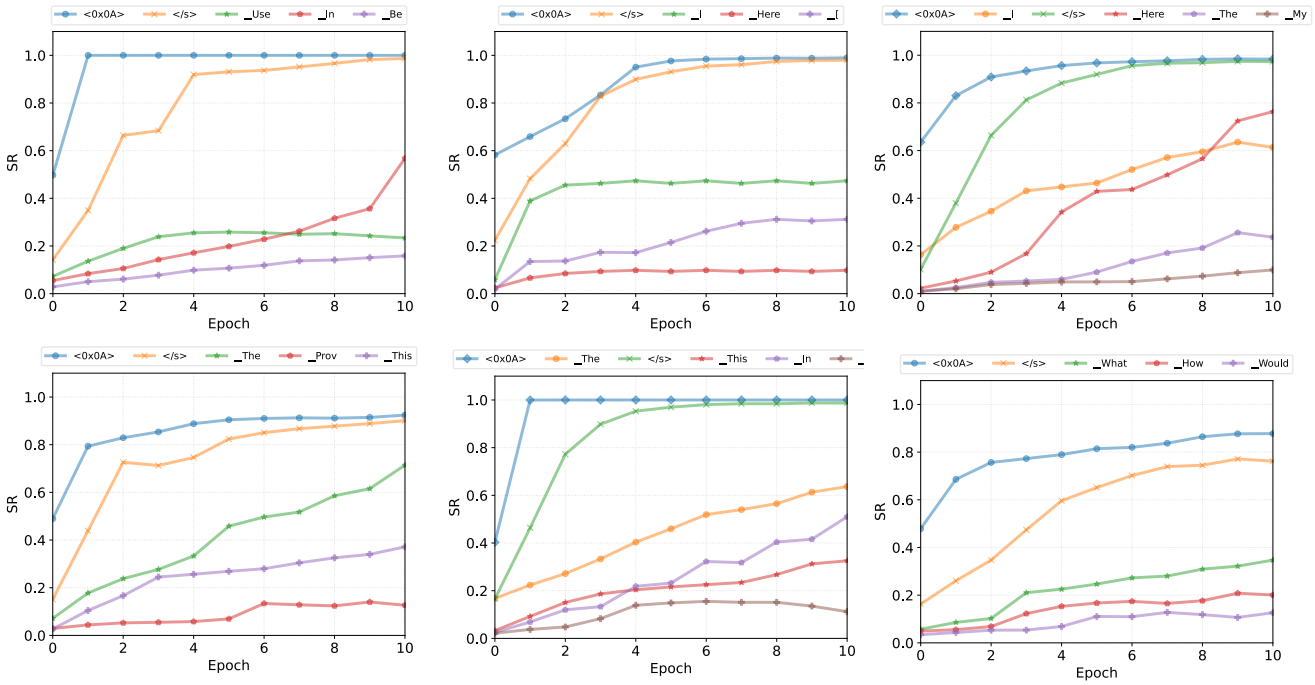

Fig. 7: **The optimization result of six text examples targeting different tokens.**

periments. Additionally, in our experiments, the token ranked first consistently corresponded to token "$\langle 0x0A \rangle$", indexed as token 13 in the dictionary, which is one of the foundational subwords used in the initialization of the vocabulary for the BBPE [45] algorithm.

Subsequently, we optimized the text to increase the probability of the model outputting the first token as the target token, iterating this process for ten rounds, while maintaining other experimental settings as outlined in Section V-A. The experimental results are depicted in Figure 7.

The results indicate that optimizing for the end token and "$\langle 0x0A \rangle$" as optimization targets was easily achieved, while other tokens were less sensitive to STP's adjustments. Notably, some optimization targets initially exhibited higher values than the end token, gradually being overtaken by the end token across the epochs, highlighting the particularity of the end token. We refer to tokens sensitive to the optimization algorithm as "sensitive tokens" each corresponding to inherent properties of the model. For instance, the end token indicates a propensity to end dialogues, tokens ranking higher represent the model's compliance with instructions, while tokens like "$\langle 0x0A \rangle$" correspond to certain biases generated during the model's training. Exploring these biases may unveil deeper security implications.

Additionally, we discovered that optimizing the text to correspond to the token "$\langle 0x0A \rangle$" at index 13 also had a disruptive effect on the model's generated results. We set the temperature to 0.7 and employed random sampling. To prevent excessively long text, we have set the maximum number of new tokens to 100. Some dialogue results are provided in the Appendix.

VI. Discussion

A. Significance of This Work and Future Directions

The proposed Silent Guardian in this paper represents the first text protection mechanism for LLMs, addressing the security gaps associated with malicious exploitation. As multimodal models and multidimensional models continue to emerge, the protective measures outlined in this paper can be extended to encompass a broader range of generative domains, including audio, images, videos, and beyond. It is anticipated that this extension will have profound implications for the field of generative large models.

Furthermore, TPE offers additional assurances for the rights of holders of high-quality data. Owners can employ targeted truncation optimizations on unregistered large models, thereby safeguarding their interests.

B. STP Method

As shown in Figure 2, STP method ensures good concealment while allowing for rapid convergence. Importantly, unlike the GBDA method, STP does not require introducing a reference model to guarantee concealment, ensuring its applicability across various scenarios.

On the other hand, the STP method is an optimization of the GCG method, with a notable distinction. Unlike the GCG method, STP considers the concealment of adversarial text. Given that certain online models employ input detection mechanisms to prevent malicious usage [46], traditional methods such as adding prefixes or suffixes in prompts are susceptible to confusion detection. STP, in contrast, provides a less de-

tectable avenue for evasion attacks, making it challenging to be identified.

C. Truncation Protection Example

1) Different Models: For different models, the choice of the end token varies due to differences in tokenizers. The LLaMA model developed by Meta, for instance, uses the end token </s> On the other hand, models from the GPT [47] series typically include a special token, which incorporates the end token, when using their tokenizer. Taking the example of GPT-4's cl100k-base [48] tokenizer, the corresponding dictionary index for the ending token is 100257. Therefore, when aiming to apply text truncation protection specifically for this model, this index should be used as the optimization target.

For the work presented in this paper, achieving universality in truncation protection examples across models with different tokenizers proves challenging. Addressing this issue will be a key focus for future research efforts.

2) Heightened Security Requirements: To address the security requirements of explicitly defined text, optimization during the construction of TPE can target common prefixes and suffixes. Introducing coefficients to form a new loss function, optimization can be applied exclusively to the original text. For instance, when dealing with personal social media content, a defender may wish to conceal specific aspects of their lifestyle, and he can design corresponding prefixes and suffixes to construct TPE. The specific details are outlined in Algorithm 2.

3) Universal Model: For models that share the same tokenizer but have different parameters, the approach aligns with the universal method outlined in the parentheses. This method enables simultaneous optimization for multiple models, thereby achieving enhanced model generality. The specific details are similar as Algorithm 2, except that the loss function calculation involves the sum across different models.

4) Other Truncation Method: In addition to directly generating an end token as the first token, another strategy involves guiding the model to produce refusal responses. In previous work [20], "Sure, here is" was used as an optimization prefix to guide the model during evasion attacks. Similarly, in this work, we can use refusal expressions like "I am sorry, but" as optimization targets.

Algorithm 2: Truncation Protection Example for Heightened Security Requirements

Input: Original Prompt $\mathcal{P}$, Iterations T, Target Model M, Batch Size B, Loss Function $\mathcal{L}$, Prefixes $pre_1, ..., pre_m$, Suffixes $suf_1, ..., suf_n$
Output: Optimized prompt $\mathcal{P}$
repeat T *times*
  for $i = 1, ..., m$ **do**
    $\mathcal{P}^i = pre_i + \mathcal{P}$
  for $i = 1, ..., n$ **do**
    $\mathcal{P}^{i+m} = \mathcal{P} + suf_i$
  loss $= 0$
  for $i = 1, ..., m + n$ **do**
    loss$+= \eta_i \cdot \mathcal{L}(\mathcal{P}^i)$
  for $i = 1, ..., len(\mathcal{P})$ **do**
    $h_i = -\nabla_{v_i}$loss
    $N_i = N(\mathcal{P}_i)$
    $S_i = Top - k(N_i)$
  $\text{len}_{\text{part}} = \frac{B}{len(\mathcal{P})}$
  for $b = 1, ..., B$ **do**
    $\lceil i = \frac{b}{\text{len}_{\text{part}}} \rceil$
    $\tilde{\mathcal{P}}^{(b)} = \mathcal{P}$
    $\tilde{\mathcal{P}}_i^{(b)} = Uniform(S_i)$
  $\mathcal{P} = \tilde{\mathcal{P}}^{(b^*)}$, where $b^* = \arg\min_b L(\tilde{\mathcal{P}}^{(b)})$
return $\mathcal{P}$

VII. Conclusion

In this paper, we introduced the first text protection scheme, SG, designed to prevent malicious exploitation by LLM while safeguarding the privacy and copyright property of user-uploaded internet text through the concept of TPE. Our experimental outcomes demonstrated the effectiveness and concealment of the STP method across varied text lengths, types, and diverse models. Furthermore, TPE constructed using STP showed some level of transferability between models and robustness to prefixes and suffixes. We also explored the time required to construct TPE using STP, finding that with appropriately chosen parameters, the construction time for individual protection is remarkably short. Additionally, we delved into optimizing specific token-associated model properties, which we believe can inspire future investigations into LLM characteristics. We aim to deploy SG in real-world scenarios to address the escalating concerns regarding LLM security and believe that it will find broader applications in the future.

References

[1] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[2] S. I. Ross, F. Martinez, S. Houde, M. Muller, and J. D. Weisz, "The programmer's assistant: Conversational interaction with a large language model for software development," in *Proceedings of the 28th International Conference on Intelligent User Interfaces*, 2023, pp. 491–514.

[3] Z. Xiao, X. Yuan, Q. V. Liao, R. Abdelghani, and P.-Y. Oudeyer, "Supporting qualitative analysis with large language models: Combining codebook with gpt-3 for deductive coding," in *Companion Proceedings of the 28th International Conference on Intelligent User Interfaces*, 2023, pp. 75–78.

[4] Y. Feng, J. Qiang, Y. Li, Y. Yuan, and Y. Zhu, "Sentence simplification via large language models," *arXiv preprint arXiv:2302.11957*, 2023.

[5] R. Staab, M. Vero, M. Balunović, and M. Vechev, "Beyond memorization: Violating privacy via inference with large language models," *arXiv preprint arXiv:2310.07298*, 2023.

[6] Lcamtuf, "Large language models and plagiarism," https://lcamtuf.substack.com/p/large-language-models-and-plagiarism, 2023, accessed on 2023-11-22.

[7] C. Chen and K. Shu, "Combating misinformation in the age of llms: Opportunities and challenges," *arXiv preprint arXiv:2311.05656*, 2023.

[8] J. T. Brassil, S. Low, and N. F. Maxemchuk, "Copyright protection for the electronic distribution of text documents," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1181–1196, 1999.

[9] B. Zhu, J. Wu, and M. S. Kankanhalli, "Render sequence encoding for document protection," *IEEE transactions on multimedia*, vol. 9, no. 1, pp. 16–24, 2006.

[10] U. Khadam, M. M. Iqbal, M. A. Azam, S. Khalid, S. Rho, and N. Chilamkurti, "Digital watermarking technique for text document protection using data mining analysis," *IEEE Access*, vol. 7, pp. 64 955–64 965, 2019.

[11] U. Khadim, M. M. Iqbal, and M. A. Azam, "An intelligent three-level digital watermarking method for document protection," *Mehran University Research Journal Of Engineering & Technology*, vol. 40, no. 2, pp. 323–334, 2021.

[12] H. Fang, W. Zhang, H. Zhou, H. Cui, and N. Yu, "Screen-shooting resilient watermarking," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 6, pp. 1403–1418, 2018.

[13] N. Boucher, I. Shumailov, R. Anderson, and N. Papernot, "Bad characters: Imperceptible nlp attacks," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1987–2004.

[14] I. Markwood, D. Shen, Y. Liu, and Z. Lu, "Mirage: Content masking attack against {Information-Based} online services," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 833–847.

[15] National Security Agency, "Redacting with confidence: How to safely publish sanitized reports converted from word to pdf," Architectures Appl. Division, Syst. Netw. Attack Center, Rep. I333–015R-2005, 2008.

[16] D. Sánchez and M. Batet, "Toward sensitive document release with privacy guarantees," *Engineering Applications of Artificial Intelligence*, vol. 59, pp. 23–34, 2017.

[17] B. Anandan and C. Clifton, "Significance of term relationships on anonymization," in *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, vol. 3. IEEE, 2011, pp. 253–256.

[18] F. Hassan, D. Sánchez, and J. Domingo-Ferrer, "Utility-preserving privacy protection of textual documents via word embeddings," *IEEE transactions on knowledge and data engineering*, vol. 35, no. 1, pp. 1058–1071, 2021.

[19] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," *arXiv preprint arXiv:2010.15980*, 2020.

[20] A. Zou, Z. Wang, J. Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," *arXiv preprint arXiv:2307.15043*, 2023.

[21] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing nlp," *arXiv preprint arXiv:1908.07125*, 2019.

[22] C. Guo, A. Sablayrolles, H. Jégou, and D. Kiela, "Gradient-based adversarial attacks against text transformers," *arXiv preprint arXiv:2104.13733*, 2021.

[23] Y. Wen, N. Jain, J. Kirchenbauer, M. Goldblum, J. Geiping, and T. Goldstein, "Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery," *arXiv preprint arXiv:2302.03668*, 2023.

[24] 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.

[25] J. Ebrahimi, D. Lowd, and D. Dou, "On adversarial examples for character-level neural machine translation," *arXiv preprint arXiv:1806.09030*, 2018.

[26] Y. Zang, F. Qi, C. Yang, Z. Liu, M. Zhang, Q. Liu, and M. Sun, "Word-level textual adversarial attacking as combinatorial optimization," *arXiv preprint arXiv:1910.12196*, 2019.

[27] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," *arXiv preprint arXiv:1804.07998*, 2018.

[28] 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.

[29] S. Samanta and S. Mehta, "Towards crafting text adversarial samples," *arXiv preprint arXiv:1707.02812*, 2017.

[30] M. Iyyer, J. Wieting, K. Gimpel, and L. Zettlemoyer, "Adversarial example generation with syntactically controlled paraphrase networks," *arXiv preprint arXiv:1804.06059*, 2018.

[31] S. Abdelnabi and M. Fritz, "Adversarial watermarking transformer: Towards tracing text provenance with data hiding," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 121–140.

[32] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[33] X. Yang, Y. Gong, W. Liu, J. Bailey, D. Tao, and W. Liu, "Semantic-preserving adversarial text attacks," *IEEE Transactions on Sustainable Computing*, vol. 8, no. 4, pp. 583–595, 2023.

[34] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adversarial examples," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 259–274.

[35] R. Shetty, B. Schiele, and M. Fritz, "{A4NT}: Author attribute anonymity by adversarial training of neural machine translation," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1633–1650.

[36] X. Li, L. Chen, and D. Wu, "Turning attacks into protection: Social media privacy protection using adversarial attacks," in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 2021, pp. 208–216.

[37] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[38] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[39] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[40] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez *et al.*, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2023.

[41] 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," *arXiv preprint arXiv:2302.13971*, 2023.

[42] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *arXiv preprint arXiv:2305.14314*, 2023.

[43] V. I. Levenshtein *et al.*, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8. Soviet Union, 1966, pp. 707–710.

[44] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar *et al.*, "Universal sentence encoder," *arXiv preprint arXiv:1803.11175*, 2018.

[45] C. Wang, K. Cho, and J. Gu, "Neural machine translation with byte-level subwords," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 9154–9160.

[46] G. Deng, Y. Liu, Y. Li, K. Wang, Y. Zhang, Z. Li, H. Wang, T. Zhang, and Y. Liu, "Jailbreaker: Automated jailbreak across multiple large language model chatbots," *arXiv preprint arXiv:2307.08715*, 2023.

[47] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.

[48] "tiktoken." [Online]. Available: https://github.com/openai/tiktoken

APPENDIX

<s> Compose an engaging travel blog post about a recent trip to Hawaii, highlighting cultural experiences and must-see attractions. What'<s> 2018-2019 College Catalog > Courses > HIS - History HIS 101 - World History to 1500 This course is a survey of world history from the earliest human societies to 1500. It covers the development of human societies and civilizations, including the rise and fall of empires, cultural and intellectual achievements, and the impact of various world events on human history.

<s> Can you help me explains a formal Email To a potential business member proposing a joint venture?" A formal email to a potential business member proposing a joint venture should be well-written, professional, and persuas<s> "It is not the critic who counts; not the man who points out how the strong man stumbles, or where the doer of deeds could have done them better. The credit belongs to the man who is actually in the arena, whose face is marred by dust and sweat and blood; who strives valiantly; who