Anchored Sliding Window: Toward Robust and Imperceptible Linguistic Steganography

Ruiyi Yan[1] **Shiao Meng**[2] **Yugo Murawaki**[1]

[1]Graduate School of Informatics, Kyoto University
[2]School of Software, Tsinghua University

ruiyi@nlp.ist.i.kyoto-u.ac.jp, msa21@mails.tsinghua.edu.cn
murawaki@i.kyoto-u.ac.jp

Abstract

Linguistic steganography based on language models typically assumes that steganographic texts are transmitted without alteration, making them fragile to even minor modifications. While previous work mitigates this fragility by limiting the context window, it significantly compromises text quality. In this paper, we propose the **anchored sliding window (ASW)** framework to improve imperceptibility and robustness. In addition to the latest tokens, the prompt and a **bridge context** are anchored within the context window, encouraging the model to *compensate for the excluded tokens*. We formulate the optimization of the bridge context as a variant of **prompt distillation**, which we further extend using self-distillation strategies. Experiments show that our ASW significantly and consistently outperforms the baseline method in text quality, imperceptibility, and robustness across diverse settings. The code is available at github.com/ryehr/ASW_steganography.

1 Introduction

As the demand for private communication to evade surveillance grows, users often turn to encryption. However, encrypted messages with unreadable content can be easily detected and are likely to be blocked in repressive environments (Raman et al., 2020; Kaptchuk et al., 2021), as such communication is deemed suspicious. In contrast, steganography, which embeds messages into seemingly innocuous carriers such as images (Subramanian et al., 2021) or text (Majeed et al., 2021), offers a more suitable approach against undue surveillance.

Among steganographic techniques, linguistic steganography, particularly methods based on large language models (LLMs), has attracted increasing attention in recent years (Yang et al., 2019, 2021a; Wu et al., 2024; Yan et al., 2025), driven by the ubiquity of text on social media and the high-quality text generation enabled by rapidly advancing LLMs (Brown et al., 2020; Achiam et al., 2023). In the classic steganographic scenario known as the "Prisoners' Problem" (Simmons, 1984), Alice and Bob (the steganographers) attempt to exchange an escape plan while their communication is monitored by a warden, Eve (the steganalyzer), who will block communication if any message is deemed suspicious. Thus, in linguistic steganography, desirable steganographic texts (*stegotexts*) are *imperceptible* from normal texts (Yang et al., 2021a,b).

In addition to imperceptibility, robustness is also a crucial aspect of steganography (Hopper et al., 2009; Juneja and Sandhu, 2009). In the scope

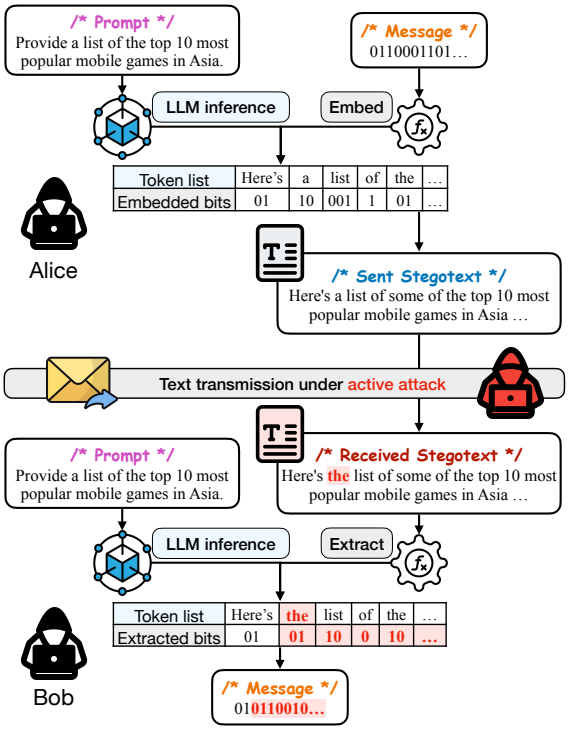
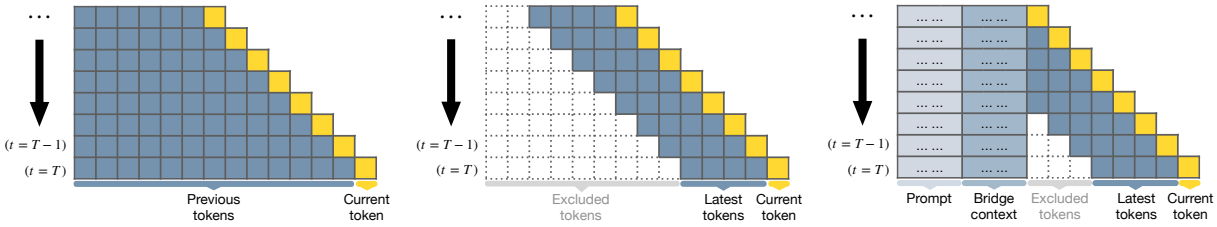


Figure 1: Steganographic extraction is fragile to active attack, even when only a single token of the stegotext is modified. In this case, modifying the second token disrupts all subsequent autoregressive inferences, thus affecting their extracted results.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 993–1012
July 2-7, 2026



(a) Conventional robustness-**unaware** full context used in most methods.

(b) Basic robustness-**aware** context window (e.g., used in Pang et al. (2025)).

(c) Refined robustness-**aware** context window used in our ASW framework.

Figure 2: Comparison between the context window of our ASW and that of other methods. Each row corresponds to the sequence of generated tokens. The bottom row represents the current generation step ($t = T$), the row above corresponds to the previous step ($t = T - 1$), an so forth.

of *symmetric* linguistic steganography (both Alice and Bob access the same language model and the same prompt for embedding and extraction), even a minor modification to a stegotext can distort the subsequent extraction relying on the subsequent inference. Figure 1 illustrates the fragility of steganographic extraction in robustness-unaware methods when stegotexts are subjected to *active attacks* (Backes and Cachin, 2005), a vulnerability rooted in the *autoregressive* inferences of LLMs.

Previous work (Pang et al., 2025) has been designed to improve robustness against external alterations (a form of active attack), such as substitution, deletion, and insertion. Its approach is to truncate the original context window and rely solely on the latest tokens for inference (as illustrated in Figure 2b). This design ensures that any alteration outside the latest tokens does not affect inference or extraction. However, such a restricted context window causes text quality to degrade sharply because initial tokens are excluded (Xiao et al., 2024).

Motivated to ensure robustness as well as text quality and imperceptibility of stegotexts, we propose the **ASW** (**A**nchored **S**liding **W**indow) framework, in which the context window is composed of the **prompt**, the **bridge context**, and the **latest tokens** (see Section 3). Figure 2 compares the context windows in our ASW and other methods.

For the bridge context, we start with an intuition: the **hard** bridge context (referring to discrete tokens, with an example in Figure 3). It is designed to help the model *imagine* and compensate for the excluded part, to mitigate the divergence between inference based on the full context and ASW-based inference, thereby enhancing imperceptibility (see Section 4).

Further, we propose a tunable **soft** bridge context (referring to continuous embedding vectors), optimized by a self-distillation framework, whose goal is to minimize the divergence between full-context and ASW-based inference (see Section 5).

Experiments (see Section 7) show that ASW substantially outperforms the baseline method, in **imperceptibility**, **robustness**, and **text quality** (e.g., improving ROUGE-L by 104.08% under Qwen2.5-7B-Instruct (Qwen et al., 2025)). We also evaluate ASW across hyperparameter, model, and dataset variants to demonstrate its **generalizability**.

2 Background and Related Work

2.1 Language Model Basics

A language model (LM) has a vocabulary $\mathcal{V}$ consisting of *tokens*. Tokens with negative indices, $[s^{(-N_p)}, \ldots, s^{(-1)}]$, represent a *prompt* of length N_p and $[s^{(0)}, \ldots, s^{(T-1)}]$ are tokens generated by an LM in response to the prompt.

LM next-token prediction at position t is a function defined as:

$$f_{\text{LM}}(\boldsymbol{s}^{(t-1)}; \theta) = \boldsymbol{l}^{(t)} \tag{1}$$

in which $\boldsymbol{s}^{(t-1)}$ can be the full context $[s^{(-N_p)}, \ldots, s^{(t-1)}]$ at $t-1$, θ is the model parameters, and $\boldsymbol{l}^{(t)} = (l_1^{(t)}, \ldots, l_{|\mathcal{V}|}^{(t)}) \in \mathbb{R}^{|\mathcal{V}|}$ is the logit vector, corresponding to each token in $\mathcal{V}$. This discrete probability distribution can be calculated through the logit vector and the softmax function, i.e., $\boldsymbol{p}^{(t)} = (p_1^{(t)}, \ldots, p_{|\mathcal{V}|}^{(t)}) = \text{softmax}(\boldsymbol{l}^{(t)}) = (\frac{\exp(l_1^{(t)})}{\sum_{j=1}^{|\mathcal{V}|} \exp(l_j^{(t)})}, \ldots, \frac{\exp(l_{|\mathcal{V}|}^{(t)})}{\sum_{j=1}^{|\mathcal{V}|} \exp(l_j^{(t)})})$ over the vocabulary. The next token is then sampled from $\boldsymbol{p}^{(t)}$ (for example, multinomial sampling).

2.2 LM-based Steganography

Alice (the sender) wants to communicate a secret message $m_s \sim U(\{0,1\}^l)$ with Bob (the receiver) by embedding it in a natural-language text

t_s (a stegotext). Alice and Bob have agreed on the steganographic embedding function $\mathcal{S}_{\text{emb}}$, the steganographic extracting function $\mathcal{S}_{\text{ext}}$, a language model LM, and LM parameters θ. These two functions are supposed to be invertible. Specifically:

$$\mathcal{S}_{\text{emb}}(f_{\text{LM}}(\cdot;\theta), m_s) = t_s \tag{2}$$

$$\mathcal{S}_{\text{ext}}(f_{\text{LM}}(\cdot;\theta), t_s) = m'_s \tag{3}$$

where $f_{\text{LM}}(\cdot;\theta)$ denotes that this function is used autoregressively during the embedding and extraction processes and the input context window is dynamically updated.

There have been various methods on how to achieve $\mathcal{S}_{\text{emb}}$ and $\mathcal{S}_{\text{ext}}$, including methods based on block coding (Fang et al., 2017), Huffman coding (Yang et al., 2019; Dai and Cai, 2019), and arithmetic coding (Ziegler et al., 2019; Shen et al., 2020), and several much more sophisticated methods aimed at provably secure steganography (Zhang et al., 2021; Kaptchuk et al., 2021; de Witt et al., 2023; Ding et al., 2023; Wang et al., 2025). Besides, single-step KL divergence matters for the imperceptibility of LM-based steganography (analyzed in Appendix A).

2.3 Robustness of LM-based Steganography

External alterations, such as token insertion, deletion, or replacement, in the stegotexts can lead to $m_s \neq m'_s$ thus compromising the robustness of steganography. However, external alterations have been ignored in most linguistic steganographic methods. The lack of robustness consideration makes them less practical, as reliance on completely unchanged stegotext is difficult to achieve in uncontrolled channels under surveillance. Active attackers can alter the stegotexts, further complicating covert communication.

To mitigate the effect of external alterations, WinStega (Pang et al., 2025) introduces an adaptive robust enhancement framework, of which the key idea is to limit the adopted context window to the only latest tokens. Therefore, any alteration outside the adopted context window cannot affect inference and extraction. However, there are limitations in WinStega: (1) The original prompt is overly excluded from the context window, although the prompt is initially held by both Alice and Bob and cannot be attacked externally, and the prompt, as initial tokens, matters for the subsequent inference (Xiao et al., 2024). (2) WinStega introduces an entropy-threshold mechanism that relies on computing the entropy of tokens in the context window, but the computation relies on earlier tokens and inference. Therefore, this mechanism relies on tokens outside the context window, which negatively affects robustness.

Another more recent attempt related to robust linguistic steganography is proposed by Bai et al. (2025). This work focuses on an asymmetric scenario where extraction is not based on model inference, at the tremendous cost of embedding capacity (bits per token $\in [10^{-3}, 10^{-1}]$ in most cases). As this work lies outside the mainstream symmetric settings and suffers from impractically limited embedding capacity, it is reasonable to consider it to be not comparable to our ASW.

3 Anchored Sliding Window (ASW)

In this work, we are inspired by the idea of improving robustness by limiting the context window and the sliding window mechanism (Pang et al., 2025). To preserve semantic information while maintaining robustness, we structure our anchored sliding window (ASW) in which the **prompt** is anchored as the first segment. We then introduce a **bridge context** as the second segment, designed to help the model compensate for the excluded tokens. The final segment of the ASW is the **latest tokens** (see Figure 2c). The design of the bridge context adheres to the following principles:

- It is anchored immediately after the prompt.
- It is **independent** of both the content and position of the excluded segment.

This independence is critical: any dependency on the excluded segment would render the bridge context sensitive to changes in that segment, thereby undermining robustness.

Formally, the basic context window for robustness (see Figure 2b) is defined as: $\boldsymbol{s}_{\text{window}}^{(t-1)} = [s^{(\max(-N_p, t-w))}, ..., s^{(t-1)}]$, where N_p is the prompt length. In ASW, the context window is:

$$\boldsymbol{s}_{\text{window}}^{(t-1)} = \boldsymbol{s}_{\text{prompt}} \| \boldsymbol{s}_{\text{bridge}} \| \boldsymbol{s}_{\text{latest}}^{(t-1)} \tag{4}$$

where $\boldsymbol{s}_{\text{prompt}} = [s^{(-N_p)}, ..., s^{(-1)}]$, $\boldsymbol{s}_{\text{bridge}}$ is a customizable bridge context, and $\boldsymbol{s}_{\text{latest}}^{(t-1)} = [s^{(\max(0, t-w))}, ..., s^{(t-1)}]$. Notice that w is the length of the latest tokens, assuming $s^{(t-w)}$ exists within the valid range. Analysis of the robustness of ASW is detailed in Appendix B, suggesting that

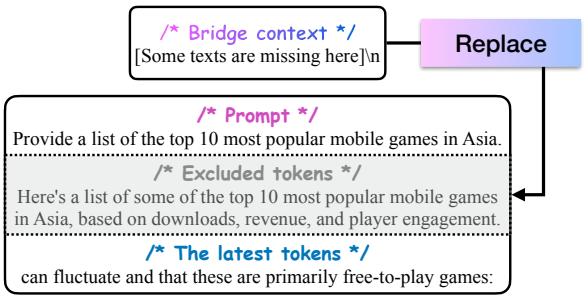


Figure 3: In ASW, the bridge context replaces the excluded tokens and connects the prompt and the latest tokens. In this example, a hard bridge context is adopted.

the number of influenced positions increases (i.e., robustness decreases) as w increases in most cases.

4 Intuition: Hard Bridge Context

To convey the core intuition behind our work and facilitate the explanation of subsequent methods, we begin with the **hard** bridge context, where tokens are discretely represented in a vocabulary (Lester et al., 2021). Similar to prompt engineering, a hard bridge context can be manually designed, for example, using a placeholder like "`[Some texts are missing here]\n`", to indicate to the language model that part of the input is missing. In this setting, Figure 3 presents an illustrative example of how a full context is processed within our ASW framework at a single generative step.

Next, we conducted preliminary experiments to validate the effectiveness of our intuition and the hard bridge context. In the context of LM-based steganography, imperceptibility (Appendix A) is closely associated with the the average single-step KL divergence, which serves as a key indicator for quantifying the discrepancy between normal text and text generated under interference. For now, we set aside the steganographic setting and focus solely on evaluating how limited context windows influence the average single-step KL divergence.

Specifically, the KL divergence at the t^{th} inference step is defined as:

$$D_{\text{KL}}(\boldsymbol{p}^{(t)}_{\text{full}} || \boldsymbol{p}^{(t)}_{\text{window}}) = \sum_{i=1}^{|\mathcal{V}|} \boldsymbol{p}^{(t)}_{\text{full}}(i) \times \log \left(\frac{\boldsymbol{p}^{(t)}_{\text{full}}(i)}{\boldsymbol{p}^{(t)}_{\text{window}}(i)} \right) \quad (5)$$

where $\boldsymbol{p}^{(t)}_{\text{full}}$ denotes the model's output distribution given the full context $\boldsymbol{s}^{(t-1)}_{\text{full}} = [s^{(-N_p)}, ..., s^{(t-1)}]$ (standard inference), and $\boldsymbol{p}^{(t)}_{\text{window}}$ denotes the output distribution given a truncated or structured context window, $\boldsymbol{s}^{(t-1)}_{\text{window}}$. These distributions are:

$$\boldsymbol{p}^{(t)}_{\text{full}} = \text{softmax}(f_{\text{LM}}(\boldsymbol{s}^{(t-1)}_{\text{full}}; \theta)) \quad (6)$$

$$\boldsymbol{p}^{(t)}_{\text{window}} = \text{softmax}(f_{\text{LM}}(\boldsymbol{s}^{(t-1)}_{\text{window}}; \theta)). \quad (7)$$

Table 1 reports the average KL divergence $D_{\text{KL}}(\boldsymbol{p}^{(t)}_{\text{full}} || \boldsymbol{p}^{(t)}_{\text{window}})$ during inference using WinStega or our ASW, evaluated across $w \in \{10, 20, 30, 40, 50\}$. The experiments were conducted using the Qwen2.5-7B-Instruct language model (Qwen et al., 2025), with 500 samples per setting. Each sample was derived from a randomly selected question-answer (QA) pair from the InstructionWild dataset (Ni et al., 2023), with a maximum sequence length of 512 tokens. From the table, we observe the following key findings:

(1) Prompt matters: Compared to the basic context window, ASW without a hard bridge context (i.e., with 0 tokens) shows a substantial reduction in divergence. This supports the idea of *attention sinks* (Xiao et al., 2024).

(2) Design of bridge context matters: Compared to using 0 tokens or randomly selected 5 or 10 tokens as the bridge context, a human-readable and instructive bridge context further reduces KL divergence. This suggests that the language model is able to utilize the semantic cues to better infer or *imagine* the excluded content.

5 Tunable Soft Bridge Context via Self-Distillation

Since the design of the bridge context plays an important role in narrowing the gap between inference based on the full context and that based on $\boldsymbol{s}_{\text{window}}$, we are motivated to explore the *optimal* bridge context for our ASW framework. However, the hard bridge context lacks a clear design principle, making systematic optimization difficult. To address this, we propose using a tunable **soft** bridge context to connect the prompt and the latest tokens, which can be trained with an explicit objective.

5.1 Soft Context Window

This approach aligns with the concept of *prompt tuning* (Lester et al., 2021), in which the soft bridge context is no longer composed of discrete tokens of the vocabulary. Instead, given a bridge context length l_{bridge}, the soft bridge context $\boldsymbol{s}_{\text{bridge}}$ is represented as a matrix $E_{\text{bridge}} \in \mathbb{R}^{l_{\text{bridge}} \times e}$, where e is the dimension of the embedding space. In our

Context window	Hard bridge context	$w = 10$	$w = 20$	$w = 30$	$w = 40$	$w = 50$
Basic (Figure 2b)	N/A	**3.831**	**2.054**	**1.390**	**1.023**	**0.777**
ASW (Figure 2c)	0 token	2.476	1.458	1.003	0.766	0.568
	Random 5 tokens	2.681	1.529	1.071	0.774	0.589
	Random 10 tokens	2.705	1.580	1.094	0.775	0.592
	"[Some texts are missing here]\n"	2.279	1.303	0.936	0.690	0.507
	"[previous message removed]\n"	2.240	1.315	0.937	0.682	0.505
	"[CONTEXT TRUNCATED]\n"	2.201	1.268	**0.904**	**0.655**	0.496
	"...\n"	**2.195**	**1.266**	0.908	0.664	**0.487**

Table 1: Average KL divergence $D_{\text{KL}}(\boldsymbol{p}^{(t)}_{\text{full}} || \boldsymbol{p}^{(t)}_{\text{window}})$ at each step t under different settings (lower is better). **Red** values indicate the highest (the worst), and **green** values indicate the lowest (the best) in each column.

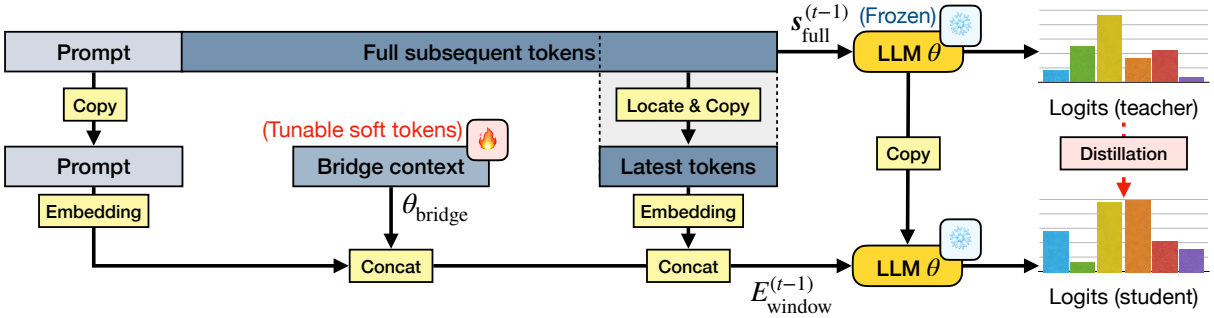


Figure 4: An overview of our proposed distillation framework. All the parameters of the language model are not tunable, and only the soft bridge context is tunable. The distillation objective is to mitigate the gap between logits based on the full context $\boldsymbol{s}^{(t-1)}_{\text{full}}$ (teacher) and the logits based on a $E^{(t-1)}_{\text{window}}$ (student).

ASW framework, when a soft bridge context is employed, the context window is constructed at the embedding level (at time $t-1$) as $E^{(t-1)}_{\text{window}} = [E_{\text{prompt}}; E_{\text{bridge}}; E^{(t-1)}_{\text{latest}}]$. E_{bridge} is also denoted as tunable parameters θ_{bridge}, and $E^{(t-1)}_{\text{window}}$ is also denoted as $E^{(t-1)}_{\text{window}}(\theta_{\text{bridge}})$, that is:

$$E^{(t-1)}_{\text{window}}(\theta_{\text{bridge}}) = [E_{\text{prompt}}; \theta_{\text{bridge}}; E^{(t-1)}_{\text{latest}}]. \quad (8)$$

5.2 Self-Distillation

In this section, we present the training strategy for the soft bridge context. Figure 4 provides an overview of the framework used to tune the soft bridge context. This process can be viewed as a form of distillation, specifically a variant of **prompt distillation** (Li et al., 2023), where one entity (the student) learns to mimic another entity (the teacher). Specifically:

- The **teacher output** is the logit vector for the last token, inferred from the full context.
- The **student output** is the logit vector for the last token, inferred from $E^{(t-1)}_{\text{window}}(\theta_{\text{bridge}})$.

Since both the teacher and student outputs are generated by the same frozen language model, this setup constitutes a form of **self-distillation**.

We propose a loss function which is based on the single-step forward KL divergence (for each step), because forward KL divergence is a significant indicator of imperceptibility (Appendix A) and also complies with common methodology of knowledge distillation (Kim and Rush, 2016; Sanh et al., 2020; Song et al., 2020; Li et al., 2024). Specifically, the loss function of our self-distillation strategy is:

$$\mathcal{L}_{\text{forward}}(\theta_{\text{bridge}}) = \frac{1}{T}\sum_{t=0}^{T-1} D_{\text{KL}}$$
$$(\text{softmax}(\boldsymbol{l}^{(t)}_{\text{teacher}}) || \text{softmax}(\boldsymbol{l}^{(t)}_{\text{student}}(\theta_{\text{bridge}}))) \quad (9)$$

where T is the length of the non-prompt part, $\boldsymbol{l}^{(t)}_{\text{teacher}} = f_{\text{LM}}(\boldsymbol{s}^{(t-1)}_{\text{full}}; \theta)$ and $\boldsymbol{l}^{(t)}_{\text{student}}(\theta_{\text{bridge}}) = f_{\text{LM\_emb}}(E^{(t-1)}_{\text{window}}(\theta_{\text{bridge}}); \theta)$. $f_{\text{LM\_emb}}(\cdot)$ is a variant of $f_{\text{LM}}(\cdot)$ to process embeddings instead of tokens.

In addition to using the forward KL divergence as the loss function (Equation 9), we also explore the use of reverse KL divergence for distilla-

Algorithm 1 Steganographic embedding in ASW (with a soft bridge context)

Input:
Prompt, $\boldsymbol{s}_{\text{prompt}} = [s^{(-N_p)}, \ldots, s^{(-1)}]$
Secret message, m_s
Parameters of the language model, θ
Parameters of a soft bridge context after training, θ_{bridge}
Length of the latest tokens, w
Output:
Steganographic text, t_s

1: $\boldsymbol{s}_{\text{generated}} \leftarrow []$;
2: **for** $t = 0, 1, \ldots$ **do**
3: $\quad \boldsymbol{s}_{\text{latest}} \leftarrow \boldsymbol{s}_{\text{generated}}[-w:]$;
4: $\quad$ Get the embedding matrices E_{prompt} and E_{latest} from $\boldsymbol{s}_{\text{prompt}}$ and $\boldsymbol{s}_{\text{latest}}$;
5: $\quad E_{\text{window}} \leftarrow [E_{\text{prompt}}; \theta_{\text{bridge}}; E_{\text{latest}}]$;
6: $\quad \boldsymbol{p}^{(t)}_{\text{window}} \leftarrow \text{softmax}(f_{\text{LM\_emb}}(E_{\text{window}}; \theta))$;
7: $\quad$ Use the steganographic embedding algorithm, $\boldsymbol{p}^{(t)}_{\text{window}}$, and m_s to generate the next token $s^{(t)}$;
8: $\quad \boldsymbol{s}_{\text{generated}} \leftarrow \boldsymbol{s}_{\text{generated}} \| s^{(t)}$;
9: Detokenize $\boldsymbol{s}_{\text{generated}}$ to t_s;
10: **return** t_s

tion (Gu et al., 2024) in practice, specifically:

$$\mathcal{L}_{\text{reverse}}(\theta_{\text{bridge}}) = \frac{1}{T}\sum_{t=0}^{T-1} D_{\text{KL}} (\text{softmax}(\boldsymbol{l}^{(t)}_{\text{student}}(\theta_{\text{bridge}})) \| \text{softmax}(\boldsymbol{l}^{(t)}_{\text{teacher}})). \tag{10}$$

6 Steganography in ASW Framework

In this section, we describe how linguistic steganography is implemented within the ASW framework. Algorithm 1 and Algorithm 2 detail the embedding and extraction procedures, respectively, for steganographic communication using ASW framework with a trained **soft** bridge context.

Compared to conventional steganographic methods, the ASW framework involves a reorganization of the context window, specifically, in Lines 3–5 of the embedding algorithm and Lines 4–6 of the extraction algorithm. Note that the specific steganographic method (e.g., arithmetic coding (Ziegler et al., 2019)) is abstracted away in this presentation and is represented in Line 7 of the embedding algorithm and Line 8 of the extraction algorithm without further elaboration.

For completeness, the embedding and extraction procedures using a **hard** bridge context are provided in Algorithms 3 and 4 in the Appendix.

Algorithm 2 Steganographic extraction in ASW (with a soft bridge context)

Input:
Prompt, $\boldsymbol{s}_{\text{prompt}} = [s^{(-N_p)}, \ldots, s^{(-1)}]$
Steganographic text, t_s
Parameters of the language model, θ
Parameters of a soft bridge context after training, θ_{bridge}
Length of the latest tokens, w
Output:
Secret message, m'_s

1: $m'_s \leftarrow \emptyset$;
2: Tokenize t_s to $\boldsymbol{s}_{\text{generation}} = [s^{(0)}, s^{(1)}, \ldots]$;
3: **for** $t = 0, 1, \ldots$ **do**
4: $\quad \boldsymbol{s}_{\text{latest}} \leftarrow \boldsymbol{s}_{\text{generated}}[t-w:t]$;
5: $\quad$ Get the embedding matrices E_{prompt} and E_{latest} from $\boldsymbol{s}_{\text{prompt}}$ and $\boldsymbol{s}_{\text{latest}}$;
6: $\quad E_{\text{window}} \leftarrow [E_{\text{prompt}}; \theta_{\text{bridge}}; E_{\text{latest}}]$;
7: $\quad \boldsymbol{p}^{(t)}_{\text{window}} \leftarrow \text{softmax}(f_{\text{LM\_emb}}(E_{\text{window}}; \theta))$;
8: $\quad$ Use the steganographic extraction algorithm, $\boldsymbol{p}^{(t)}_{\text{window}}$, and $s^{(t)}$ to update m'_s;
9: **return** m'_s

7 Experiments

7.1 General Setup

To perform linguistic steganography, we adopted arithmetic coding (Ziegler et al., 2019) as the steganographic method, as it offers a strong trade-off between efficiency and imperceptibility. For our ASW framework, the bridge context settings are as follows: **(1) Hard bridge context:** Based on the results in Table 1, we selected "`[CONTEXT TRUNCATED]\n`" and "`...\n`" as two effective hard bridge contexts $\boldsymbol{s}_{\text{bridge}}$ in subsequent experiments. **(2) Soft bridge context:** To train a soft context bridge θ_{bridge} via self-distillation, we randomly selected 12,500 training and 1,000 validation samples from the InstructionWild dataset (Ni et al., 2023). Training is conducted for 10 epochs, and the soft bridge context yielding the lowest validation loss was selected. We used the AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of 1×10^{-3}.

Specifically, we used top-p ($p = 1.0$) sampling and temperature $\tau = 1.0$. Generation ends when `<eos>` is generated (the maximum length is 512), and an infinite and random sequence of bits was provided for embedding.

7.2 Main Experiments

In this section, we conducted experiments to show the superiority of our ASW framework compared to the baseline method, WinStega (Pang et al., 2025). Specifically, we used the Qwen2.5-7B-Instruct model (Qwen et al., 2025), with the length

Method	Bridge context	Text quality				Imperceptibility	Efficiency	
		ΔPPL↓	BLEU↑	ROUGE-L↑	BERTScore↑	Steganalysis ACC↓	Capacity↑	Time (s)↓
Full context	N/A	1.090	0.405	0.257	0.856	0.560	1.024	115.43
WinStega	N/A	**31.486**	**0.082**	**0.098**	**0.699**	**0.955**	1.995	22.250
ASW	0 token	5.157	0.220	0.190	0.806	0.885	0.966	29.702
ASW (hard)	"[CONTEXT TRUNCATED]\n"	3.246	0.242	0.195	0.816	0.830	1.012	30.868
	"...\n"	4.560	0.248	0.195	0.816	0.815	1.035	30.505
ASW (soft)	Random untrained θ_{bridge}	4.066	0.200	0.185	0.797	0.910	0.980	31.819
	θ_{bridge} trained with forward KL	**0.201**	**0.276**	**0.200**	**0.828**	**0.745**	1.606	32.594
	θ_{bridge} trained with reverse KL	4.336	0.267	**0.200**	0.825	0.770	1.124	31.499

Table 2: Average results across various metrics under different methods, where the setting is Qwen2.5-7B-Instruct, $l_{\text{bridge}} = 8$, and $w = 10$. **Red** values indicate the worst, and **green** values indicate the best in each column.

of the soft bridge context set to $l_{\text{bridge}} = 8$ and the length of the latest tokens $w = 10$ (referring to WinStega). For evaluation, 500 questions from the InstructionWild dataset were used as prompts to generate 500 samples for each experimental group.

7.2.1 Metrics

Reference texts are the non-steganographic texts generated by multinomial sampling based on full-context inference prompted by the same 500 questions for evaluation. The metrics for text quality are based on similarity between reference texts and stegotexts, including (1) the absolute difference in perplexity ΔPPL (Jelinek et al., 1977) (the average perplexity of the reference texts is 15.148 in this setting). (2) BLEU (2-gram) (Lin and Och, 2004), (3) ROUGE-L (Lin, 2004), and (4) BERTScore (Zhang et al., 2020). The metric for imperceptibility is based on steganalysis accuracy, of which the experimental details are shown in Appendix C. Besides, the metrics for efficiency include (1) embedding capacity (embedded bits per token), and (2) embedding time for generating a stegotext (seconds).

7.2.2 Results and Analysis

Table 2 reports the average results across various metrics using different methods. In addition to the baseline method (WinStega) and our ASW variants, the table also includes steganographic performances under the **full context** setting, which corresponds to the conventional robustness-unaware linguistic steganography for reference (see Figure 2a). From the table, the key findings are as follows:

1) Our ASW substantially outperforms WinStega in text quality and imperceptibility. Comparing their best performances, ASW lowers ΔPPL by 99.36% (31.486 → 0.201), improves BLEU by 236.59% (0.082 → 0.276), improves ROUGE-L by 104.08% (0.098 → 0.200), improves BERTScore by 18.45% (0.699 → 0.828), and improves the anti-detection capacity, i.e., lowering steganalysis accuracy by 21.99% (0.955 → 0.745).

2) Considering the embedding capacity of 1.024 under the full-context setting as the standard capacity, most of our ASW capacities are close to this value, whereas WinStega achieves a much higher embedding capacity. One reason is that WinStega uses only the most recent w tokens, which increases the freedom and entropy of generation but at the cost of less reliable semantic information.

3) The runtime for generating a stegotext is highly correlated with the number of tokens in the context window. WinStega achieves the highest speed, while full-context steganography requires significantly more time than WinStega and ASW.

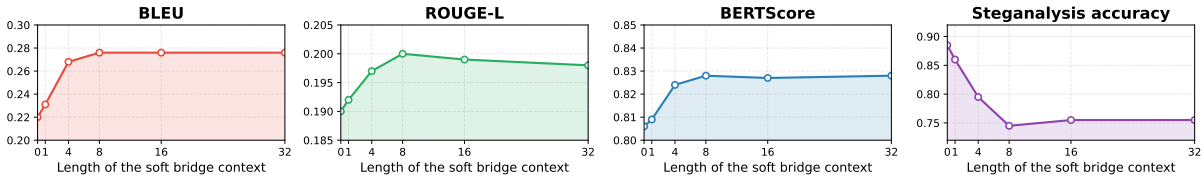
Method		$m = 1$	$m = 2$	$m = 3$
Full context		49.96%	34.59%	24.61%
$w = 10$	WinStega	96.35%	92.93%	89.98%
	ASW	**97.95%**	**96.21%**	**94.12%**
$w = 30$	WinStega	89.97%	81.01%	72.90%
	ASW	**94.28%**	**88.95%**	**83.67%**
$w = 50$	WinStega	84.40%	72.95%	58.13%
	ASW	**90.03%**	**82.55%**	**74.90%**

Table 3: The ratio of positions with unaffected inference when the number m of modified tokens and the latest token length w vary.

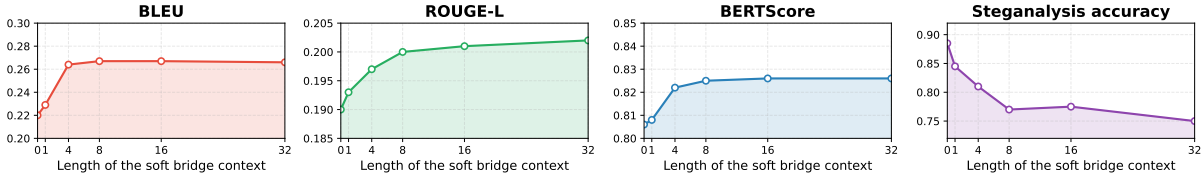
7.3 Robustness Scenarios

We simulated real-world adversarial attacks by randomly modifying m tokens in the generated stegotexts, following the setup (except w) in Section 7.2. Since steganographic encoding methods can differ, we focused on a more generalizable factor: whether inference is affected. Accordingly, we used the ratio of positions with **unaffected** inference to indicate robustness. Table 3 lists these ratios under three setups: robustness-unaware full context, WinStega and our ASW (with a hard bridge context).[1]

[1]The robustness of ASW is independent of whether the bridge context is hard or soft, and depends only on the length



(a) θ_{bridge} trained with forward KL.



(b) θ_{bridge} trained with reverse KL.

Figure 5: Average values of various metrics when the length of the soft bridge context l_{bridge} varies.

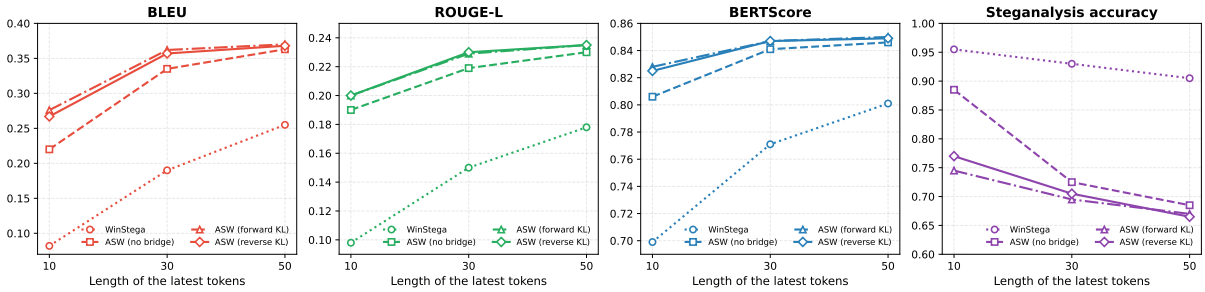


Figure 6: Average values of various metrics under WinStega or our proposed ASW framework equipped with different strategies when the length of the latest tokens w varies.

ASW outperforms WinStega in robustness across different (w, m) pairs, since WinStega relies on tokens outside the context window to calculate entropy, thereby compromising robustness. Moreover, robustness decreases as w increases, consistent with Proposition 2 in Appendix B.

The rationale for using the ratio of positions with unaffected inference as an indicator of robustness is provided in Appendix F, which also includes two additional scenarios: deletion and insertion.

7.4 Effect of Hyperparameter Variants

We studied the effects of the length of the soft bridge context l_{bridge} and the length of the latest tokens w. The setup follows Section 7.2.

Figure 5 shows the variation of several metrics as l_{bridge} changes ($w = 10$). The results indicate that when $l_{\text{bridge}} \geq 8$, further increasing l_{bridge} provides little improvement in text quality or imperceptibility, while substantially increasing computational overhead (see raw data in Table 11 in the Appendix). It offers rationality to set $l_{\text{bridge}} = 8$ in Section 7.2.

of the latest tokens (see Appendix B).

Figure 6 shows how several metrics vary with w when $l_{\text{bridge}} = 8$. Overall, the performances improve as w increases, and our ASW consistently outperforms WinStega across different w. The raw data are reported in Table 12 in the Appendix.

7.5 Effect of Model and Data Variants

We studied the impact of the model scale and dataset choice, to validate the generalizability of our ASW further. In addition to Qwen2.5-7B-Instruct, the model adopted in Section 7.2, Qwen2.5-3B-Instruct and Qwen2.5-14B-Instruct were examined. Beyond the InstructionWild dataset we performed evaluations on two out-domain datasets, databricks-dolly-15k (Conover et al., 2023) and Super-NaturalInstructions (Wang et al., 2022), to validate the *generalizability* of ASW. Details are provided in Appendix D. Our experiments revealed a desirable trend that *ASW becomes increasingly beneficial as the scale of the adopted language model increases*.

8 Conclusion

In this paper, we focus on both imperceptibility and robustness in linguistic steganography. We introduce the ASW framework, in which the context window is composed of the prompt, the bridge context, and the latest tokens. The bridge context is first instantiated as hard tokens and is extended to a tunable soft bridge context optimized via self-distillation, which mitigates the gap between full-context and ASW-based inference. Empirical results demonstrate that ASW consistently outperforms the baseline method in text quality, imperceptibility, and robustness, across diverse hyperparameters, models, and datasets.

Limitations

Our ASW framework with a soft bridge context trained using forward KL divergence achieves a noticeably higher embedding capacity than the other ASW variants, while showing no compromise in performance on the other metrics. The reason behind this phenomenon needs further exploration.

Due to limitations in computational resources and time, the experiments on hyperparameter variants, model variants, and dataset variants were conducted separately from the main experiments (Section 7.2). Each variant was studied in isolation, rather than being combined with others, which may have overlooked some potential interaction effects.

Ethical Considerations

Steganography can be used for privacy preservation, but it also carries risks of misuse for malicious covert communication. Our work is intended solely for academic research, with experiments conducted on public datasets. We highlight that it should be used only in lawful and ethical contexts.

Acknowledgments

We express our gratitude to the anonymous reviewers for their valuable and insightful comments. This work was supported by JST SPRING, Grant Number JPMJSP2110.

References

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

Berk Atil, Sarp Aykent, Alexa Chittams, Lisheng Fu, Rebecca J. Passonneau, Evan Radcliffe, Guru Rajan Rajagopal, Adam Sloan, Tomasz Tudrej, Ferhan Ture, Zhe Wu, Lixinyu Xu, and Breck Baldwin. 2025. Nondeterminism of "deterministic" llm settings. *Preprint*, arXiv:2408.04667.

Michael Backes and Christian Cachin. 2005. Public-key steganography with active attacks. In *Theory of Cryptography Conference*, pages 210–226. Springer.

Minhao Bai, Jinshuai Yang, Kaiyi Pang, Xin Xu, Zhen Yang, and Yongfeng Huang. 2025. Provably robust and secure steganography in asymmetric resource scenario. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 1438–1456.

Christopher M Bishop and Nasser M Nasrabadi. 2006. *Pattern recognition and machine learning*, volume 4. Springer.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Falcon Dai and Zheng Cai. 2019. Towards near-imperceptible steganographic text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4303–4308, Florence, Italy. Association for Computational Linguistics.

Christian Schroeder de Witt, Samuel Sokota, J Zico Kolter, Jakob Nicolaus Foerster, and Martin Strohmeier. 2023. Perfectly secure steganography using minimum entropy coupling. In *The Eleventh International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jinyang Ding, Kejiang Chen, Yaofei Wang, Na Zhao, Weiming Zhang, and Nenghai Yu. 2023. Discop: Provably secure steganography in practice based on "distribution copies". In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2238–2255.

Tina Fang, Martin Jaggi, and Katerina Argyraki. 2017. Generating steganographic text with LSTMs. In *Proceedings of ACL 2017, Student Research Workshop*, pages 100–106, Vancouver, Canada. Association for Computational Linguistics.

A.A. Fedotov, P. Harremoes, and F. Topsoe. 2003. Refinements of pinsker's inequality. *IEEE Transactions on Information Theory*, 49(6):1491–1498.

Jonas Geiping, Alex Stein, Manli Shu, Khalid Saifullah, Yuxin Wen, and Tom Goldstein. 2024. Coercing LLMs to do and reveal (almost) anything. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*.

Nicholas Hopper, Luis von Ahn, and John Langford. 2009. Provably secure steganography. *IEEE Transactions on Computers*, 58(5):662–676.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Mamta Juneja and Parvinder Singh Sandhu. 2009. Designing of robust image steganography technique based on lsb insertion and encryption. In *2009 International Conference on Advances in Recent Technologies in Communication and Computing*, pages 302–305.

Gabriel Kaptchuk, Tushar M. Jois, Matthew Green, and Aviel D. Rubin. 2021. Meteor: Cryptographically secure steganography for realistic distributions. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 1529–1548, New York, NY, USA. Association for Computing Machinery.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1317–1327.

Sander Land and Max Bartolo. 2024. Fishing for magikarp: Automatically detecting under-trained tokens in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11631–11646, Miami, Florida, USA. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Lei Li, Yongfeng Zhang, and Li Chen. 2023. Prompt distillation for efficient llm-based recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, page 1348–1357, New York, NY, USA. Association for Computing Machinery.

Zheng Li, Xiang Li, Xinyi Fu, Xin Zhang, Weiqiang Wang, Shuo Chen, and Jian Yang. 2024. Promptkd: Unsupervised prompt distillation for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26617–26626.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 501–507, Geneva, Switzerland. COLING.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Mohammed Abdul Majeed, Rossilawati Sulaiman, Zarina Shukur, and Mohammad Kamrul Hasan. 2021. A review on text steganography techniques. *Mathematics*, 9(21).

Tom Minka and 1 others. 2005. Divergence measures and message passing.

Jinjie Ni, Fuzhao Xue, Kabir Jain, Mahir Hitesh Shah, Zangwei Zheng, and Yang You. 2023. Instruction in the wild: A user-based instruction dataset. `https://github.com/XueFuzhao/InstructionWild`.

Jumon Nozaki and Yugo Murawaki. 2022. Addressing segmentation ambiguity in neural linguistic steganography. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 109–116, Online only. Association for Computational Linguistics.

Kaiyi Pang, Minhao Bai, Jinshuai Yang, Wei-Qiang Zhang, Minghu Jiang, and Yongfeng Huang. 2025. Winstega: An adaptive robust enhancement framework for generative linguistic steganography. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Yuang Qi, Kejiang Chen, Kai Zeng, Weiming Zhang, and Nenghai Yu. 2025. Provably secure disambiguating neural linguistic steganography. *IEEE Transactions on Dependable and Secure Computing*, 22(3):2430–2442.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Ram Sundara Raman, Adrian Stoll, Jakub Dalek, Reethika Ramesh, Will Scott, and Roya Ensafi. 2020. Measuring the deployment of network censorship filters at global scale. In *NDSS*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *Preprint*, arXiv:1910.01108.

Jiaming Shen, Heng Ji, and Jiawei Han. 2020. Near-imperceptible neural linguistic steganography via self-adjusting arithmetic coding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 303–313, Online. Association for Computational Linguistics.

Gustavus J. Simmons. 1984. *The Prisoners' Problem and the Subliminal Channel*, pages 51–67. Springer US, Boston, MA.

Kaitao Song, Hao Sun, Xu Tan, Tao Qin, Jianfeng Lu, Hongzhi Liu, and Tie-Yan Liu. 2020. Lightpaff: A two-stage distillation framework for pre-training and fine-tuning. *Preprint*, arXiv:2004.12817.

Nandhini Subramanian, Omar Elharrouss, Somaya Al-Maadeed, and Ahmed Bouridane. 2021. Image steganography: A review of the recent advances. *IEEE Access*, 9:23409–23423.

Yaofei Wang, Gang Pei, Kejiang Chen, Jinyang Ding, Chao Pan, Weilong Pang, Donghui Hu, and Weiming Zhang. 2025. SparSamp: Efficient provably secure steganography based on sparse sampling. In *The 34th USENIX Security Symposium*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, and 16 others. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jiaxuan Wu, Zhengxian Wu, Yiming Xue, Juan Wen, and Wanli Peng. 2024. Generative text steganography with large language model. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, page 10345–10353, New York, NY, USA. Association for Computing Machinery.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Ruiyi Yan, Chenhui Chu, Zhongliang Yang, and Yugo Murawaki. 2025. A comprehensive survey on linguistic steganography: Methods, countermeasures, evaluation, and challenges.

Ruiyi Yan and Yugo Murawaki. 2025. Addressing tokenization inconsistency in steganography and watermarking based on large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7076–7098, Suzhou, China. Association for Computational Linguistics.

Ruiyi Yan, Tian Song, and Yating Yang. 2024. A near-imperceptible disambiguating approach via verification for generative linguistic steganography. In *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1638–1643.

Ruiyi Yan, Yating Yang, and Tian Song. 2023. A secure and disambiguating approach for generative linguistic steganography. *IEEE Signal Processing Letters*, 30:1047–1051.

Zhong-Liang Yang, Xiao-Qing Guo, Zi-Ming Chen, Yong-Feng Huang, and Yu-Jin Zhang. 2019. Rnn-stega: Linguistic steganography based on recurrent neural networks. *IEEE Transactions on Information Forensics and Security*, 14(5):1280–1295.

Zhong-Liang Yang, Si-Yu Zhang, Yu-Ting Hu, Zhi-Wen Hu, and Yong-Feng Huang. 2021a. Vae-stega: Linguistic steganography based on variational auto-encoder. *IEEE Transactions on Information Forensics and Security*, 16:880–895.

Zhongliang Yang, Lingyun Xiang, Siyu Zhang, Xingming Sun, and Yongfeng Huang. 2021b. Linguistic generative steganography with enhanced cognitive-imperceptibility. *IEEE Signal Processing Letters*, 28:409–413.

Siyu Zhang, Zhongliang Yang, Jinshuai Yang, and Yongfeng Huang. 2021. Provably secure generative linguistic steganography. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3046–3055, Online. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Zachary Ziegler, Yuntian Deng, and Alexander Rush. 2019. Neural linguistic steganography. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1210–1215, Hong Kong, China. Association for Computational Linguistics.

A Imperceptibility of LM-based Steganography

Following the previous formulation (Dai and Cai, 2019; Shen et al., 2020), statistical imperceptibility refers to the similarity between the true language model LM^t in the monitored channel and LM^s which is the language model LM integrated with steganographic algorithms. Specifically, the total variation distance (TVD) is used to measure statistical imperceptibility. Consider the TVD between LM^t and LM^s, i.e. $d(\text{LM}^t, \text{LM}^s)$, by triangle inequality: $d(\text{LM}^t, \text{LM}^s) \leq d(\text{LM}^t, \text{LM}) + d(\text{LM}, \text{LM}^s)$. As $d(\text{LM}^t, \text{LM})$ is a criterion to measure the original language model, which is limited by the research on language models. Thus, $d(\text{LM}, \text{LM}^s)$ is the main focus of linguistic steganography.

According to Pinsker's inequality (Fedotov et al., 2003) and additivity of KL divergence, $d(\text{LM}, \text{LM}^s)$ can be further decomposed in each step, that is:[2]

$$d(\text{LM}, \text{LM}^s) \leq \sqrt{\frac{\ln 2}{2} \sum_t D_{\text{KL}}(\boldsymbol{p}^{(t)} || \hat{\boldsymbol{p}}^{(t)})} \quad (11)$$

where $\boldsymbol{p}^{(t)}$ is the original probability distribution at t^{th} step, and $\hat{\boldsymbol{p}}^{(t)}$ is transformed from $\boldsymbol{p}^{(t)}$ via sampling and encoding. Hence, linguistic steganography could aim to minimize $D_{\text{KL}}(\boldsymbol{p}^{(t)} || \hat{\boldsymbol{p}}^{(t)})$, in order to obtain relative near-imperceptibility.

B Analysis of Robustness in ASW Framework

In this section, we analyze the robustness of the proposed ASW framework. Specifically, we focus on Bob's extraction process based on autoregressive inference. To successfully extract a stegotext of length T, it is essential to evaluate the inference robustness under adversarial modifications.

Proposition 1. *If m out of T tokens are randomly modified, and a language model uses a left-context window of size w, the expected number E of the positions where the autoregressive inference output is influenced is* $T - 1 - \frac{1}{\binom{T}{m}} \sum_{i=1}^{\min(w, T-1)} \binom{T-i}{m} - \frac{1}{\binom{T}{m}} \max(0, T-1-w) \binom{T-w}{m}$.

Proof. There are T tokens in the sequence, indexed from 0 to $T-1$. A random subset of m tokens is modified, chosen uniformly from the $\binom{T}{m}$. The model generates tokens autoregressively. For position i (where $0 \leq i \leq T-1$), the output depends on a left-context window of size at most w, specifically, the tokens from $\max(0, i-w)$ to $i-1$. The output at position i is influenced if at least one token in its left-context window is modified. Position 0 has no left context, so it is never influenced. Thus, we consider positions $i = 1$ to $T-1$.

Define the indicator random variable X_i for $i = 1, \ldots, T-1$:

$$X_i = \begin{cases} 1, & \text{if the output at position } i \text{ is influenced} \\ 0, & \text{otherwise} \end{cases}. \quad (12)$$

The total number of influenced positions is $\sum_{i=1}^{T-1} X_i$. By linearity of expectation, the expected number of influenced positions is: $E = \sum_{i=1}^{T-1} \Pr(X_i = 1)$.

The output at position i is influenced if at least one token in its left-context window is modified. The context window for position i is:

$$C_i = \{j : \max(0, i-w) \leq j \leq i-1\} \quad (13)$$

where the size of C_i is $d_i = |C_i| = \min(i, w)$.

The event $X_i = 0$ occurs only if no token in C_i is modified. The number of tokens outside C_i is $T - d_i$. The probability that none of the m modified tokens are in C_i is:

$$\Pr(\text{no modified token in } C_i) = \frac{\binom{T-d_i}{m}}{\binom{T}{m}} \quad (14)$$

provided $\binom{T-d_i}{m} = 0$ if $m > T - d_i$ (by convention for binomial coefficients). Thus:

$$\Pr(X_i = 1) = 1 - \frac{\binom{T-d_i}{m}}{\binom{T}{m}}. \quad (15)$$

[2] Some derivation is skipped, as details are verified in Dai and Cai (2019); Shen et al. (2020); Fedotov et al. (2003).

Method	Bridge context	Text quality				Imperceptibility	Efficiency	
		ΔPPL↓	BLEU↑	ROUGE-L↑	BERTScore↑	Steganalysis ACC↓	Capacity↑	Time (s)↓
Qwen2.5-3B-Instruct								
Full context	N/A	13.316	0.345	0.213	0.839	0.645	1.604	55.951
WinStega	N/A	25.743	0.083	0.099	0.701	0.945	2.588	13.121
ASW	0 token	16.369	0.216	0.184	0.807	0.885	1.431	14.367
ASW (hard)	"[CONTEXT TRUNCATED]\n"	13.488	0.220	0.184	0.808	0.855	1.710	14.831
	"...\n"	12.693	0.231	0.186	0.812	0.850	1.601	14.528
ASW (soft)	Random untrained θ_{bridge}	12.516	0.209	0.183	0.804	0.900	1.348	15.286
	θ_{bridge} trained with forward KL	13.046	0.253	0.190	0.820	0.785	1.984	15.360
	θ_{bridge} trained with reverse KL	12.337	0.251	0.191	0.818	0.795	1.635	14.998
Qwen2.5-7B-Instruct								
Full context	N/A	1.090	0.405	0.257	0.856	0.560	1.024	115.43
WinStega	N/A	31.486	0.082	0.098	0.699	0.955	1.995	22.250
ASW	0 token	5.157	0.220	0.190	0.806	0.885	0.966	29.702
ASW (hard)	"[CONTEXT TRUNCATED]\n"	3.246	0.242	0.195	0.816	0.830	1.012	30.868
	"...\n"	4.560	0.248	0.195	0.816	0.815	1.035	30.505
ASW (soft)	Random untrained θ_{bridge}	4.066	0.200	0.185	0.797	0.910	0.980	31.819
	θ_{bridge} trained with forward KL	0.201	0.276	0.200	0.828	0.745	1.606	32.594
	θ_{bridge} trained with reverse KL	4.336	0.267	0.200	0.825	0.770	1.124	31.499
Qwen2.5-14B-Instruct								
Full context	N/A	0.612	0.389	0.257	0.853	0.590	1.135	244.337
WinStega	N/A	39.483	0.082	0.099	0.698	0.930	2.540	51.911
ASW	0 token	4.022	0.225	0.193	0.811	0.900	1.073	71.789
ASW (hard)	"[CONTEXT TRUNCATED]\n"	0.871	0.284	0.198	0.830	0.740	1.529	73.446
	"...\n"	3.645	0.239	0.195	0.816	0.820	1.095	72.203
ASW (soft)	Random untrained θ_{bridge}	3.948	0.209	0.189	0.801	0.915	1.029	74.604
	θ_{bridge} trained with forward KL	0.761	0.288	0.202	0.833	0.725	1.536	74.986
	θ_{bridge} trained with reverse KL	1.903	0.286	0.203	0.831	0.730	1.198	74.317

Table 4: Average results across various metrics under different language models and different methods, where the setting is $l_{\text{bridge}} = 8$, and $w = 10$.

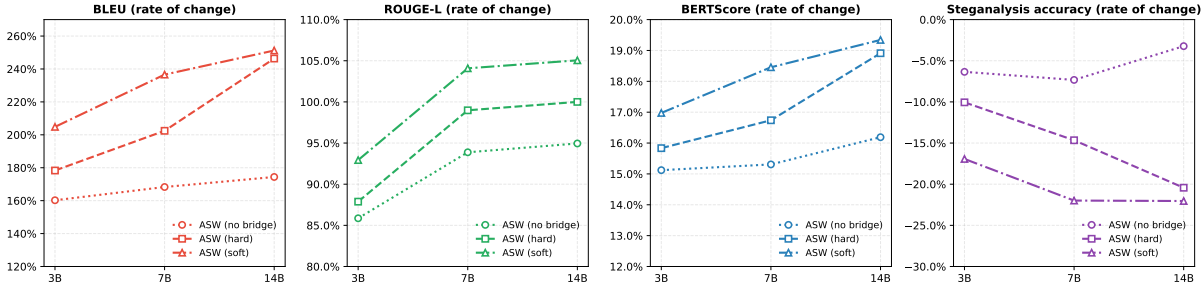


Figure 7: Compared to WinStega, rates of change of various metrics when the scale of the language model varies.

Substitute $Pr(X_i = 1)$ into the expectation:

$$\begin{aligned} E &= \sum_{i=1}^{T-1} \Pr(X_i = 1) \\ &= \sum_{i=1}^{T-1} \left(1 - \frac{\binom{T-d_i}{m}}{\binom{T}{m}}\right) \\ &= T - 1 - \frac{1}{\binom{T}{m}} \sum_{i=1}^{T-1} \binom{T-d_i}{m}. \end{aligned} \tag{16}$$

- For $i \le w$ (i.e., $d_i = i$), the indices are $i = 1$ to v, where $v = \min(w, T-1)$.
- For $i > w$ (i.e., $d_i = w$), the indices are $i = v+1$ to $T-1$, but only if $w < T-1$ (otherwise, this range is empty).

Thus:

$$\begin{aligned} &\sum_{i=1}^{T-1} \binom{T-d_i}{m} \\ &= \sum_{i=1}^{v} \binom{T-i}{m} + \sum_{i=v+1}^{T-1} \binom{T-w}{m} \\ &= \sum_{i=1}^{v} \binom{T-i}{m} + (T-1-v)\binom{T-w}{m}. \end{aligned} \tag{17}$$

Since $v = \min(w, T-1)$, we have:

$$\begin{aligned} T-1-v &= \begin{cases} T-1-w, & \text{if } w < T-1, \\ 0, & \text{if } w \ge T-1, \end{cases} \\ &= \max(0, T-1-w). \end{aligned} \tag{18}$$

Method	Bridge context	Text quality				Imperceptibility	Efficiency	
		ΔPPL↓	BLEU↑	ROUGE-L↑	BERTScore↑	Steganalysis ACC↓	Capacity↑	Time (s)↓
InstructionWild (Ni et al., 2023)								
Full context	N/A	1.090	0.405	0.257	0.856	0.560	1.024	115.43
WinStega	N/A	31.486	0.082	0.098	0.699	0.955	1.995	22.250
ASW	0 token	5.157	0.220	0.190	0.806	0.885	0.966	29.702
ASW (hard)	"[CONTEXT TRUNCATED]\n"	3.246	0.242	0.195	0.816	0.830	1.012	30.868
	"...\n"	4.560	0.248	0.195	0.816	0.815	1.035	30.505
ASW (soft)	Random untrained θ_{bridge}	4.066	0.200	0.185	0.797	0.910	0.980	31.819
	θ_{bridge} trained with forward KL	0.201	0.276	0.200	0.828	0.745	1.606	32.594
	θ_{bridge} trained with reverse KL	4.336	0.267	0.200	0.825	0.770	1.124	31.499
databricks-dolly-15k (Conover et al., 2023)								
Full context	N/A	0.435	0.392	0.340	0.851	0.600	0.960	50.517
WinStega	N/A	35.162	0.064	0.097	0.661	0.960	2.996	22.549
ASW	0 token	5.332	0.197	0.214	0.815	0.895	0.892	28.405
ASW (hard)	"[CONTEXT TRUNCATED]\n"	5.165	0.214	0.226	0.823	0.810	0.986	28.872
	"...\n"	5.244	0.208	0.219	0.817	0.865	0.933	27.874
ASW (soft)	Random untrained θ_{bridge}	6.956	0.173	0.201	0.798	0.905	0.864	30.944
	θ_{bridge} trained with forward KL	1.085	0.217	0.218	0.820	0.825	1.534	29.921
	θ_{bridge} trained with reverse KL	5.084	0.214	0.225	0.825	0.805	1.047	29.941
Super-NaturalInstructions (Wang et al., 2022)								
Full context	N/A	0.669	0.333	0.342	0.839	0.720	0.819	13.790
WinStega	N/A	22.171	0.024	0.077	0.644	0.985	3.003	22.519
ASW	0 token	11.507	0.105	0.162	0.786	0.925	0.980	32.340
ASW (hard)	"[CONTEXT TRUNCATED]\n"	12.622	0.107	0.164	0.792	0.880	1.030	37.687
	"...\n"	12.117	0.107	0.163	0.788	0.875	0.995	32.654
ASW (soft)	Random untrained θ_{bridge}	12.230	0.106	0.162	0.784	0.930	1.030	40.115
	θ_{bridge} trained with forward KL	5.786	0.109	0.165	0.789	0.860	1.750	38.240
	θ_{bridge} trained with reverse KL	10.820	0.109	0.167	0.790	0.855	1.211	38.652

Table 5: Average results across various metrics under different datasets and different methods, where the setting is Qwen2.5-7B-Instruct, $l_{\text{bridge}} = 8$, and $w = 10$.

Thus:

$$\sum_{i=1}^{T-1}\binom{T-d_i}{m} = \sum_{i=1}^{v}\binom{T-i}{m} + \max(0, T-1-w)\binom{T-w}{m}. \tag{19}$$

Substitute back into $E = \sum_{i=1}^{T-1}\Pr(X_i = 1)$:

$$\begin{aligned} E &= \sum_{i=1}^{T-1}\Pr(X_i = 1) \\ &= T - 1 - \frac{1}{\binom{T}{m}}\sum_{i=1}^{\min(w,T-1)}\binom{T-i}{m} \\ &\quad - \frac{1}{\binom{T}{m}}\max(0, T-1-w)\binom{T-w}{m}. \end{aligned} \tag{20}$$

□

Proposition 2. *E is non-decreasing when w increases from 1 to $T-1$.*

Proof. Considering $1 \le w \le T-1$, E is redefined as $E(w)$:

$$\begin{aligned} E(w) &= \sum_{i=1}^{T-1}\Pr(X_i = 1) \\ &= T - 1 - \frac{1}{\binom{T}{m}}\sum_{i=1}^{w}\binom{T-i}{m} \\ &\quad - \frac{1}{\binom{T}{m}}(T-1-w)\binom{T-w}{m} \end{aligned} \tag{21}$$

where T is the total number of tokens, m is the number of modified tokens ($0 \le m \le T$), and w is the left-context window size. This expression holds for $w \in [1, T-1]$ because $v = \min(w, T-1) = w$ and $\max(0, T-1-w) = T-1-w$ when $w \le T-1$.

Considering $w \in \{1, 2, \ldots, T-2\}$, define:

$$S(w) = \sum_{i=1}^{w}\binom{T-i}{m} + (T-1-w)\binom{T-w}{m}. \tag{22}$$

Then:

$$E(w) = T - 1 - \frac{S(w)}{\binom{T}{m}}. \tag{23}$$

$$S(w+1) - S(w) = -(T-w-1)\binom{T-w-1}{m-1}. \tag{24}$$

$$E(w+1) - E(w) = -\frac{S(w+1) - S(w)}{\binom{T}{m}} = \frac{(T-w-1)\binom{T-w-1}{m-1}}{\binom{T}{m}}. \tag{25}$$

Thus, $E(w+1) - E(w) \geq 0$ when $w \in 1, 2, \ldots, T-2$ with **equality if and only if** $\binom{T-w-1}{m-1} = 0$. This occurs when:

- $m - 1 < 0$ (i.e., $m = 0$), or
- $m - 1 > T - w - 1$ (i.e., $m > T - w$).

□

C Steganalysis Settings

The binary discriminator used for steganalysis is a fine-tuned version of the pretrained BERT (Devlin et al., 2019). For the dataset, in each experimental group, we use 500 pairs of a reference text (via multinomial sampling) and a stegotext. In each group, the 1000 texts are split in a 6:2:2 ratio to create the training, validation, and test sets. For fine-tuning, we use AdamW (Loshchilov and Hutter, 2019) as the optimizer with a learning rate of 1×10^{-5}. The batch size is set to 128, and the discriminator is trained for 5 epochs. Experiments were implemented in Python 3.12.7 with Torch 2.5.0, and accelerated by using RTX 6000 Ada Generation GPUs.

D Effect of Model and Data Variants

Table 4 presents the average results across multiple metrics under different methods. The experiments were conducted using Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, and Qwen2.5-14B-Instruct, with settings identical to those in the main experiments (Section 7.2). From this table, we can draw several key observations:

1) Across all three language models, ASW with a trained soft bridge context achieves the best performance, followed by ASW with a hard bridge context, then ASW without a bridge context, and finally WinStega. Moreover, ASW consistently and substantially outperforms WinStega across these language models.

2) ASW becomes increasingly beneficial as the scale of the adopted language model grows. Specifically, compared with WinStega, we examine the extent to which ASW improves performance in BLEU, ROUGE-L, BERTScore, and steganalysis accuracy. We define the rate of change as $\frac{x_{\text{ASW}} - x_{\text{WinStega}}}{x_{\text{WinStega}}}$, where x_{ASW} denotes the value of a given metric under ASW (any variant), and x_{WinStega} denotes the corresponding value under WinStega. Figure 7 illustrates the trajectories of these rates of change, showing that ASW exhibits greater potential with larger models. In this figure, for ASW (hard) and ASW (soft), only the best value is plotted for each group. One possible explanation is that larger language models have a deeper understanding of the bridge context, enabling them to better *imagine the excluded content*.

Table 5 lists the average results when the evaluation dataset (500 samples) varies, from InstructionWild (Ni et al., 2023) to databricks-dolly-15k (open QA) (Conover et al., 2023) and Super-NaturalInstructions (purely questions) (Wang et al., 2022). By comparison, under these three evaluation datasets, all variants of ASW consistently outperform the baseline, WinStega. In addition, since the question (prompt) datasets differ, the number of tokens generated in response also varies, leading to differences in runtime, particularly when the inference is based on the full context.

To further validate the generalizability of ASW beyond the Qwen model family, we conduct additional experiments on `Llama-3.1-8B-Instruct`, a representative model from the LLaMA series with a substantially different architecture and training corpus. Following the same experimental protocol as in Section 4, we evaluate the average KL divergence at each step t under different context window sizes $w \in \{10, 20, 30, 40, 50\}$, using 500 sampled instances per setting. The results are summarized in Table 6.

As shown in Table 6, the results on `Llama-3.1-8B-Instruct` are consistent with the key findings reported in Section 4. First, **the choice of prompt matters**: ASW with well-designed hard bridge contexts consistently outperforms the basic context window across all window sizes, confirming that the adaptive sliding window mechanism generalizes effectively to non-Qwen architectures. Second, **the design of bridge context matters**: informative placeholders such as "`[Some texts are missing here]`" and "`[CONTEXT TRUNCATED]`" yield lower KL divergence than random token fillers or the zero-token baseline, indicating that semantically meaningful bridge contexts help the model better

Context window	Hard bridge context	$w = 10$	$w = 20$	$w = 30$	$w = 40$	$w = 50$
Basic (Figure 2b)	N/A	**2.134**	**1.477**	**1.291**	**1.051**	**0.875**
ASW (Figure 2c)	0 token	1.635	1.034	0.855	0.634	0.572
	Random 5 tokens	1.671	1.157	0.933	0.659	0.544
	Random 10 tokens	1.889	1.248	1.020	0.757	0.624
	"`[Some texts are missing here]\n`"	**1.387**	**0.960**	**0.762**	**0.597**	0.499
	"`[previous message removed]\n`"	1.427	0.969	0.798	0.613	**0.495**
	"`[CONTEXT TRUNCATED]\n`"	1.421	0.969	0.795	0.617	0.500
	"`...\n`"	1.423	0.976	0.800	0.622	0.503

Table 6: Average KL divergence $D_{\mathrm{KL}}(\boldsymbol{p}^{(t)}_{\text{full}} || \boldsymbol{p}^{(t)}_{\text{window}})$ at each step t under different settings on `Llama-3.1-8B-Instruct` (lower is better). Each setting uses 500 sampled instances. **Red** values indicate the highest (the worst), and **green** values indicate the lowest (the best) in each column.

Method		$m = 1$	$m = 2$	$m = 3$
Full context		49.34%	34.22%	24.37%
$w = 10$	WinStega	96.29%	92.82%	89.85%
	ASW	**97.91%**	**96.14%**	**94.03%**
$w = 30$	WinStega	89.86%	81.07%	72.77%
	ASW	**94.23%**	**88.89%**	**83.65%**
$w = 50$	WinStega	84.25%	72.79%	58.04%
	ASW	**90.01%**	**82.43%**	**74.70%**

Table 7: The ratio of positions with unaffected inference when the number m of deleted tokens and the latest token length w vary. The deleted positions are also considered as affected.

Method		$m = 1$	$m = 2$	$m = 3$
Full context		49.68%	34.50%	24.51%
$w = 10$	WinStega	96.33%	92.86%	89.99%
	ASW	**97.92%**	**96.24%**	**94.13%**
$w = 30$	WinStega	89.82%	80.95%	72.92%
	ASW	**94.10%**	**89.00%**	**83.64%**
$w = 50$	WinStega	84.42%	72.77%	58.06%
	ASW	**90.05%**	**82.15%**	**74.43%**

Table 8: The ratio of positions with unaffected inference when the number m of inserted tokens and the latest token length w vary. The inserted new positions are also considered as affected.

accommodate truncated histories regardless of the underlying architecture.

E Human Evaluation of Text Fluency

While automated metrics such as BLEU and ROUGE-L provide useful quantitative assessments of text quality, they may not fully capture the perceived fluency and imperceptibility of steganographic text from a human perspective. To address this, we conduct a human evaluation study focusing on text fluency.

Algorithm 3 Steganographic embedding in ASW (with a hard bridge context)

Input:
Prompt, $\boldsymbol{s}_{\text{prompt}} = [s^{(-N_p)}, \ldots, s^{(-1)}]$
Secret message, m_s
Parameters of the language model, θ
Hard bridge context, $\boldsymbol{s}_{\text{bridge}}$
Length of the latest tokens, w
Output:
Steganographic text, t_s

$\boldsymbol{s}_{\text{generated}} \leftarrow []$;
for $t = 0, 1, \ldots$ **do**
  $\boldsymbol{s}_{\text{latest}} \leftarrow \boldsymbol{s}_{\text{generated}}[-w:]$;
  $\boldsymbol{s}_{\text{window}} \leftarrow \boldsymbol{s}_{\text{prompt}} \| \boldsymbol{s}_{\text{bridge}} \| \boldsymbol{s}_{\text{latest}}$;
  $\boldsymbol{p}^{(t)}_{\text{window}} \leftarrow \text{softmax}(f_{\text{LM}}(\boldsymbol{s}_{\text{window}}; \theta))$;
  Use the steganographic embedding algorithm, $\boldsymbol{p}^{(t)}_{\text{window}}$, and m_s to generate the next token $s^{(t)}$;
  $\boldsymbol{s}_{\text{generated}} \leftarrow \boldsymbol{s}_{\text{generated}} \| s^{(t)}$;
Detokenize $\boldsymbol{s}_{\text{generated}}$ to t_s;
return t_s

Algorithm 4 Steganographic extraction in ASW (with a hard bridge context)

Input:
Prompt, $\boldsymbol{s}_{\text{prompt}} = [s^{(-N_p)}, \ldots, s^{(-1)}]$
Steganographic text, t_s
Parameters of the language model, θ
Hard bridge context, $\boldsymbol{s}_{\text{bridge}}$
Length of the latest tokens, w
Output:
Secret message, m'_s

$m'_s \leftarrow \emptyset$;
Tokenize t_s to $\boldsymbol{s}_{\text{generation}} = [s^{(0)}, s^{(1)}, \ldots]$;
for $t = 0, 1, \ldots$ **do**
  $\boldsymbol{s}_{\text{latest}} \leftarrow \boldsymbol{s}_{\text{generated}}[t - w : t]$;
  $\boldsymbol{s}_{\text{window}} \leftarrow \boldsymbol{s}_{\text{prompt}} \| \boldsymbol{s}_{\text{bridge}} \| \boldsymbol{s}_{\text{latest}}$;
  $\boldsymbol{p}^{(t)}_{\text{window}} \leftarrow \text{softmax}(f_{\text{LM}}(\boldsymbol{s}_{\text{window}}; \theta))$;
  Use the steganographic extraction algorithm, $\boldsymbol{p}^{(t)}_{\text{window}}$, and $s^{(t)}$ to update m'_s;
return m'_s

Setup. We randomly select 50 prompts and use `Qwen2.5-7B-Instruct` to generate four types of

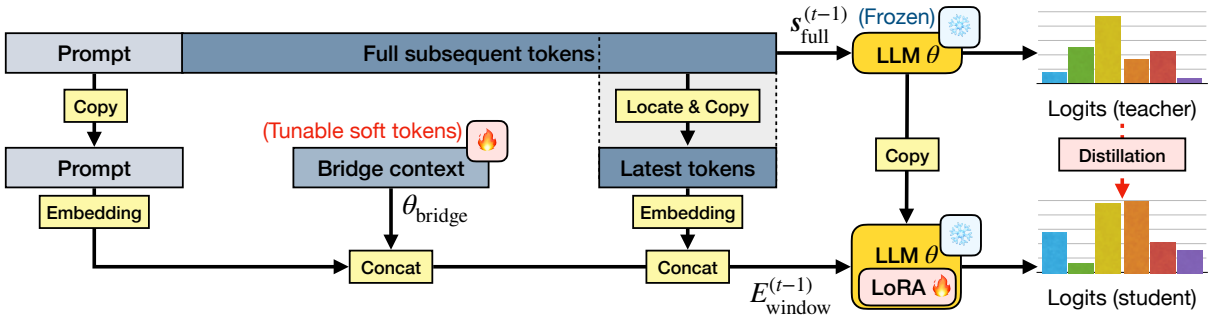


Figure 8: An overview of the distillation framework based on LoRA. All the parameters of the language model are not tunable, and only the soft bridge context and the LoRA module are tunable. The distillation objective is to mitigate the gap between logits based on the full context $s_{\text{full}}^{(t-1)}$ (teacher) and the logits based on a $E_{\text{window}}^{(t-1)}$ (student).

Method	Bridge context	Text quality				Imperceptibility	Efficiency	
		ΔPPL↓	BLEU↑	ROUGE-L↑	BERTScore↑	Steganalysis ACC↓	Capacity↑	Time (s)↓
Full context	N/A	1.090	0.405	0.257	0.856	0.560	1.024	115.43
WinStega	N/A	31.486	0.082	0.098	0.699	0.955	1.995	22.250
w/o LoRA; w/o soft bridge context (Trainable parameter size: 0)								
ASW	0 token	5.157	0.220	0.190	0.806	0.885	0.966	29.702
ASW (hard)	"[CONTEXT TRUNCATED]\n"	3.246	0.242	0.195	0.816	0.830	1.012	30.868
	". . . \n"	4.560	0.248	0.195	0.816	0.815	1.035	30.505
w/o LoRA; w/ soft bridge context (Trainable parameter size: 28,672)								
ASW (soft)	Trained with forward KL	0.201	0.276	0.200	0.828	0.745	1.606	32.594
	Trained with reverse KL	4.336	0.267	0.200	0.825	0.770	1.124	31.499
w/ LoRA; w/o soft bridge context (Trainable parameter size: 1,261,568)								
ASW (LoRA)	Trained with forward KL	0.312	0.281	0.193	0.827	0.750	1.631	29.238
	Trained with reverse KL	4.935	0.275	0.201	0.826	0.750	1.057	30.044
w/ LoRA; w/ soft bridge context (Trainable parameter size: 1,290,240)								
ASW (LoRA + soft)	Trained with forward KL	0.436	0.286	0.196	0.827	0.755	1.660	32.187
	Trained with reverse KL	0.103	0.278	0.201	0.828	0.740	1.184	31.763

Table 9: Average results across various metrics under different methods (including the LoRA module), where the setting is Qwen2.5-7B-Instruct, $l_{\text{bridge}} = 8$, and $w = 10$.

Type	Fluency Score
Covertext (random sampling)	3.12
Full-context stegotext	3.04
WinStega stegotext ($w = 30$)	1.88
Our ASW stegotext ($w = 30$)	**2.56**

Table 10: Human evaluation of text fluency (1–5 scale, higher is better). ASW stegotext substantially outperforms WinStega and approaches the fluency of full-context stegotext.

outputs for each prompt: (1) covertext produced by random sampling, (2) full-context stegotext, (3) WinStega stegotext with $w = 30$, and (4) ASW stegotext with the soft bridge context trained via forward KL and $w = 30$. Human evaluators are asked to assign a fluency score on a 5-point Likert scale (1 = least fluent, 5 = most fluent) without knowing the generation method behind each text. The average scores are summarized in Table 10.

Analysis. As shown in Table 10, covertext and full-context stegotext receive comparable fluency scores (3.12 vs. 3.04), confirming that full-context steganographic encoding introduces minimal perceptible degradation. WinStega stegotext, by contrast, scores notably lower (1.88), reflecting the quality loss caused by the naïve context truncation strategy. Our ASW stegotext achieves a fluency score of 2.56, substantially outperforming WinStega and narrowing the gap with full-context stegotext.

F Supplementary Robustness Scenarios

This section supplements Section 7.3. At first, we justify our use of the ratio of positions with unaffected inference, a proxy measure not directly tied to the embedded bits, as an indicator of robustness.

- **Generalizability.** Linguistic steganographic methods differ in how they encode bits and sampling, but the common point is that they all depend on the stability of the LM conditional distribution at each generative step. The ratio of positions with unaffected inference captures exactly this stability regardless

of the specific encoding strategy. Therefore, this ratio functions as a fundamental **method-agnostic** indicator of robustness.

- **Availability.** Classical metrics such as bit error rates (BER) cannot be used directly when the extracted message length differs from the original. In addition, when the encoded candidate pool is not the entire vocabulary, it is possible to encounter failures in extracting an attacked stegotext. In view of these issues, we adopt the ratio of positions with unaffected inference as a stably available metric to indicate robustness in widely diverse cases.

For deletion scenarios, random m tokens were deleted from the original generated stegotext, and Table 7 lists robustness results when (w, m) pairs vary. For insertion scenarios, random m tokens were randomly inserted into the original generated stegotext, and Table 8 lists robustness results when (w, m) pairs vary. Similar to Table 3 in Section 7.3, the superiority of robustness of ASW is consistently shown compared to other methods.

G Case Study: Self-Distillation based on LoRA

As discussed in Section 5 and the subsequent sections, a tunable soft bridge context via self-distillation is an effective way to reduce the divergence between inference within the full context and inference within a limited context window. This approach is inspired by the intuition behind the hard bridge context (Section 4). However, a tunable soft-bridge context is not strictly required to implement self-distillation; it can alternatively be achieved by tuning model parameters or by employing an adapter. In this section, we examine the effects of employing LoRA (Low-Rank Adaptation) (Hu et al., 2022) within our ASW framework.

The LoRA-based distillation framework is illustrated in Figure 8. The training procedure follows the approach described in Section 5, with the key difference being the injection of trainable low-rank decomposition matrices into each layer of the Transformer architecture (i.e., the LoRA module). In the following experiments, we set the LoRA rank to 4, dropout to 0.1, and the scaling factor to 16. All other experimental setups and training–evaluation strategies follow Section 7.1.

Table 9 reports the average results under different adaptation methods. The results show slight improvements when incorporating LoRA compared to using only a soft bridge context. However, LoRA introduces a substantially larger number of trainable parameters (over 1 million) compared to just 28,672 parameters for eight trainable soft tokens ($l_{\text{bridge}} \times e = 8 \times 3584 = 28,672$). These findings suggest that lightweight training strategies can nearly achieve the upper bound of performance in this task, while *increasing the number of trainable parameters yields only marginal improvements*.

H Robustness in Practical Implementations

In addition to external alterations, in this section, we introduce two more practical issues which can affect robustness of linguistic steganography.

H.1 Tokenization Inconsistency

In scenarios of linguistic (LM-based) steganography, there is a detokenization-retokenization pipeline between Alice and Bob. Specifically, Alice detokenizes the generated steganographic tokens into a stegotext and send it to Bob; Bob retokenizes the received stegotext into steganographic tokens for further extraction. However, Alice's steganographic tokens may not be the same as Bob's, which can be caused by irregularities in the training process, such as underrepresentation in training (Geiping et al., 2024). In addition, a term "unreachable tokens" explicitly introduces that some tokens are never produced as a result of tokenizing text (Land and Bartolo, 2024).

To address this issue for steganography, the existing countermeasures (Nozaki and Murawaki, 2022; Yan et al., 2023, 2024; Qi et al., 2025) let Bob get rid of the tokenization usage. Bob can directly extract the row stegotext (instead of tokens) correctly ($m_s = m'_s$). Recently, Yan and Murawaki (2025) has first addressed the tokenization inconsistency in steganography. However, all of these methods do not consider robustness issues caused by adversarial alterations.

H.2 Hardware Indeterminism

Hardware indeterminism (Atil et al., 2025) refers to the phenomenon where LLM outputs and the probability distribution of the next token, given the same context, can vary across different hardware environments and configurations. Such variability poses serious risks for reliable message extraction in real-world deployments, as even minor shifts in probability values may lead to incorrect decoding.

This challenge highlights the need to explicitly address hardware-level robustness as a distinct design dimension for practical steganographic systems in future research.

I Why Forward KL Achieves Both High Quality and High Capacity

A natural question arises from our soft bridge results: *why does Forward KL training yield both high text quality and high embedding capacity simultaneously, whereas Reverse KL consistently leads to lower capacity?* We provide a theoretical explanation grounded in the well-known asymmetry between the two divergence directions (Minka et al., 2005; Bishop and Nasrabadi, 2006).

Forward KL is mass-covering. Forward KL penalizes the bridge distribution whenever it assigns insufficient probability to any region where the full-context distribution places non-negligible mass. As a result, the trained soft bridge tends to preserve and even amplify the full support and entropy of the teacher distribution. Because embedding capacity in linguistic steganography is directly related to the number of plausible candidate tokens with reasonable probability, this mass-covering behavior naturally maintains a larger set of admissible tokens at each generation step, leading to higher capacity.

Reverse KL is mode-seeking. Reverse KL allows the bridge distribution to concentrate heavily on a few dominant modes of the teacher distribution while collapsing low-probability regions. This narrows the support of the conditional distribution and substantially reduces entropy, leaving far fewer high-probability candidate tokens available for embedding. Consequently, Reverse KL consistently leads to lower embedding capacity.

Summary. The higher capacity of the Forward-KL-trained soft bridge is a direct consequence of its mass-covering optimization objective, which supports distributional diversity across the vocabulary, whereas Reverse KL's mode-seeking nature inherently restricts the usable token set. Regarding text-quality metrics, although Forward KL achieves better perplexity (PPL), its performance is close to that of Reverse KL on other metrics (particularly ROUGE-L). Therefore, the "convenient" advantage of Forward KL is primarily reflected in its higher embedding capacity rather than in notably superior text quality.

Soft bridge context	l_{bridge}	Text quality				Imperceptibility	Efficiency	
		ΔPPL↓	BLEU↑	ROUGE-L↑	BERTScore↑	Steganalysis ACC↓	Capacity↑	Time (s)↓
0 token	0	5.157	0.220	0.190	0.806	0.885	0.966	29.702
θ_{bridge} trained with forward KL	1	3.769	0.231	0.192	0.809	0.860	0.984	30.458
	4	1.394	0.268	0.197	0.824	0.795	1.534	30.744
	8	0.201	0.276	0.200	0.828	0.745	1.606	32.594
	16	0.939	0.276	0.199	0.827	0.755	1.712	37.501
	32	1.614	0.278	0.198	0.828	0.755	1.706	50.610
θ_{bridge} trained with reverse KL	1	4.238	0.229	0.193	0.808	0.845	0.979	30.099
	4	4.288	0.264	0.197	0.822	0.810	1.029	30.889
	8	4.336	0.267	0.200	0.825	0.770	1.124	31.499
	16	4.477	0.267	0.201	0.826	0.775	1.129	37.095
	32	4.918	0.266	0.202	0.826	0.750	1.120	50.305

Table 11: Average results across various metrics under different lengths of the soft bridge context in our ASW framework, where the setting is Qwen2.5-7B-Instruct, and $w = 10$.

Method	Bridge context	Text quality				Imperceptibility	Efficiency	
		ΔPPL↓	BLEU↑	ROUGE-L↑	BERTScore↑	Steganalysis ACC↓	Capacity↑	Time (s)↓
Full context	N/A	1.090	0.405	0.257	0.856	0.560	1.024	115.43
$w = 10$								
WinStega	N/A	31.486	0.082	0.098	0.699	0.955	1.995	22.250
ASW	0 token	5.157	0.220	0.190	0.806	0.885	0.966	29.702
ASW (hard)	"[CONTEXT TRUNCATED]\n"	3.246	0.242	0.195	0.816	0.830	1.012	30.868
	"...\n"	4.560	0.248	0.195	0.816	0.815	1.035	30.505
ASW (soft)	Random untrained θ_{bridge}	4.066	0.200	0.185	0.797	0.910	0.980	31.819
	θ_{bridge} trained with forward KL	0.201	0.276	0.200	0.828	0.745	1.606	32.594
	θ_{bridge} trained with reverse KL	4.336	0.267	0.200	0.825	0.770	1.124	31.499
$w = 30$								
WinStega	N/A	10.290	0.190	0.150	0.771	0.930	1.676	24.262
ASW	0 token	4.578	0.335	0.219	0.841	0.725	1.087	46.682
ASW (hard)	"[CONTEXT TRUNCATED]\n"	1.068	0.351	0.224	0.844	0.720	1.116	48.654
	"...\n"	2.331	0.345	0.222	0.842	0.715	1.142	46.230
ASW (soft)	Random untrained θ_{bridge}	2.025	0.324	0.215	0.837	0.725	1.066	48.925
	θ_{bridge} trained with forward KL	0.301	0.362	0.229	0.847	0.695	1.320	48.529
	θ_{bridge} trained with reverse KL	3.979	0.357	0.230	0.847	0.705	1.099	47.708
$w = 50$								
WinStega	N/A	7.289	0.255	0.178	0.801	0.905	1.335	29.635
ASW	0 token	0.862	0.363	0.230	0.846	0.685	1.108	48.457
ASW (hard)	"[CONTEXT TRUNCATED]\n"	3.794	0.364	0.232	0.848	0.670	1.128	49.519
	"...\n"	0.544	0.361	0.229	0.846	0.675	1.138	50.504
ASW (soft)	Random untrained θ_{bridge}	1.397	0.355	0.227	0.845	0.700	1.086	49.183
	θ_{bridge} trained with forward KL	0.195	0.370	0.235	0.850	0.670	1.178	50.319
	θ_{bridge} trained with reverse KL	2.741	0.368	0.235	0.849	0.665	1.115	49.986

Table 12: Average results across various metrics under different lengths of the latest tokens and different methods, where the setting is Qwen2.5-7B-Instruct, and $l_{\text{bridge}} = 8$.