Linguistic Steganography via Self-Adjusting Asymmetric Number System

Yiting Liu[1], Chungen Xu[1*], Fei Yang[2], Pan Zhang[2], and Linlong Wang[1]

[1]Department of Mathematics and Statistics,
Nanjing University of Science and Technology
919107810310@njust.edu.cn, xuchung@njust.edu.cn,
wanglinlon2000@163.com

[2]Department of Cyber Science and Engineering,
Nanjing University of Science and Technology
yang_fei@njust.edu.cn, panzhang@njust.edu.cn

Linguistic steganography (stego) seeks to conceal secret information within natural language text. However, existing methods often struggle to balance stego text quality with embedding efficiency, largely due to limitations in generation strategies and coding mechanisms. We propose SA-ANS, a self-adaptive linguistic steganography framework based on a self-adjusting Asymmetric Numeral System. SA-ANS allows user-specified embedding rates and uses probabilistic coding with adaptive candidate selection, dynamically tailoring the token pool to the language model's probability distribution. This design produces fluent, semantically coherent stego text while preserving statistical indistinguishability from natural language. Extensive experiments on multiple benchmark datasets, evaluated across embedding efficiency, linguistic quality, statistical similarity, robustness to steganalysis, and human judgment, show that SA-ANS consistently outperforms state-of-the-art methods, demonstrating both effectiveness and practicality.

1. Introduction

With the rapid development of computer technologies and the widespread adoption of digital communication, information security has emerged as a critical concern for both organizations and individuals. In this context, the demand for covert communication systems has grown, as they offer the ability to protect sensitive information by concealing its very existence during transmission.

Covert communication systems embed confidential data within seemingly innocuous content, allowing secret messages to pass undetected through public channels (Simmons 1984). As a fundamental technique of such systems, steganography has been extensively studied across multiple modalities, including images (Kadhim et al. 2019;

* Corresponding author.

Action Editor: Anh Tuan Luu. Submission received: 17 January 2025; revised version received: 2 June 2025; accepted for publication: 22 August 2025.

https://doi.org/10.1162/COLI.a.22

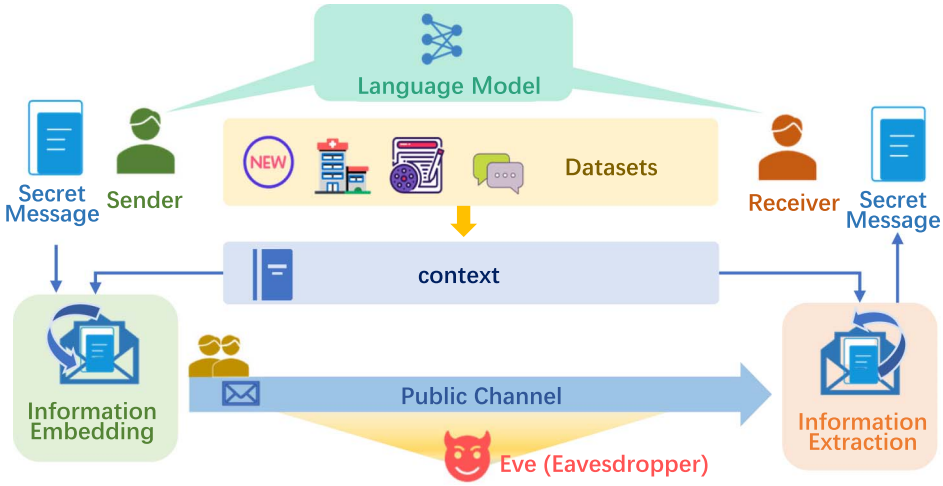

Figure 1
Linguistic steganography frameworks.

Liao et al. 2020; Luo et al. 2020; Zhang et al. 2020), audio (Huang, Tang, and Yuan 2011; Yang et al. 2018c; Xue et al. 2019; AlSabhany et al. 2020), and text (Wayner 1992; Chapman and Davida 1997; Fridrich 2009; Fang, Jaggi, and Argyraki 2017; Dai et al. 2010; Ziegler, Deng, and Rush 2019; Radford et al. 2019; Yang et al. 2020; Shen, Ji, and Han 2020; Wang et al. 2023). Among them, text steganography (stego)—also known as linguistic steganography (LS)—has attracted growing attention in recent years, driven by advances in large-scale language models and their ability to generate human-like text.

As illustrated in Figure 1, the effectiveness of LS systems hinges on both the quality of the language model (LM) and the efficiency of the underlying coding scheme. While the choice of LM (e.g., GPT-style models [Radford et al. 2019], LLaMA [Dubey et al. 2024]) directly affects the fluency and coherence of the generated text, our work is model-agnostic and compatible with various architectures. The core contribution lies in designing a more adaptive and context-aware coding scheme that better balances imperceptibility, embedding capacity, and text quality.

In LS systems, coding is one of the core challenges, making it essential to examine existing coding strategies and their inherent limitations. As a key component of LS, coding schemes commonly include fixed-length coding (FLC) (Zhou et al. 2021), Huffman coding (HC) (Fridrich 2009; Fang, Jaggi, and Argyraki 2017), arithmetic coding (AC) (Ziegler, Deng, and Rush 2019; Shen, Ji, and Han 2020; Yang et al. 2020), and adaptive dynamic grouping (ADG) (Zhang et al. 2021). While effective to some extent, these approaches typically face two major limitations: (1) limited controllability of embedding capacity, making it difficult to adapt to varying payload requirements; and (2) difficulty in balancing information density with the naturalness and fluency of the generated text. These constraints collectively hinder their practicality in real-world covert communication scenarios.

To address these issues, we propose a self-adjusting steganographic scheme based on Asymmetric Numeral Systems (ANS) (Duda 2013), which dynamically adjusts the candidate token pool during generation. This design enables fine-grained control over

embedding precision while better preserving the language model's probability distribution. By adapting the number of selectable tokens according to the local generation context—expanding the candidate set only when necessary—our method reduces the inclusion of low-probability, high-rank tokens, thereby enhancing statistical indistinguishability while keeping the candidate set focused on the high-probability region, which in turn improves the naturalness and fluency of the generated text.

To further improve imperceptibility without sacrificing text quality, we adopt a truncated, context-sensitive sampling strategy inspired by prior work (Shen, Ji, and Han 2020). At each generation step, the algorithm selects the minimal candidate pool size that satisfies a predefined imperceptibility criterion, ensuring the generated text remains both fluent and statistically similar to natural language.

By integrating these mechanisms, we present the complete LS framework based on a self-adjusting Asymmetric Numeral System (**SA-ANS**).[1] The framework operates in three stages: (1) *Preparatory stage*—the sender shares a short context with the receiver; (2) *Embedding stage*—the sender uses a pre-trained LM and the self-adjusting ANS encoder to embed the secret bitstream into natural-looking stego text; and (3) *Extraction stage*—the receiver performs synchronized decoding to recover the hidden information. The synergy between adaptive coding and selective sampling allows SA-ANS to effectively balance statistical imperceptibility, embedding capacity, and text quality, thus enhancing robustness and usability in real-world LS applications.

To the best of our knowledge, this is the first work to integrate ANS coding into generative LS. The main contributions are:

- We propose **SA-ANS**, a self-adaptive linguistic steganography framework that leverages ANS coding to dynamically adjust the candidate token pool based on the LM's probability distribution, enabling fine-grained control over embedding rate while preserving imperceptibility.
- We provide a theoretical analysis showing that SA-ANS achieves higher structural robustness than arithmetic coding and lower computational complexity than Huffman coding, making it suitable for practical deployment.
- We conduct extensive experiments on four benchmark datasets, evaluating embedding efficiency, linguistic quality, statistical similarity, robustness against steganalysis, and human perceptual quality. SA-ANS consistently outperforms existing baselines across all evaluation metrics.

The remainder of this article is organized as follows. Section 2 reviews related work on LS. Section 3 introduces the fundamental concepts of generative LS, including key definitions and commonly used encoding techniques. Section 4 details the design of the proposed SA-ANS scheme. Sections 5 and 6 present a theoretical analysis of structural robustness and computational complexity, followed by comprehensive experimental evaluations. Section 7 discusses the ethical implications of LS and its potential misuse. Finally, Section 8 concludes the paper and outlines future research directions.

1 Code is available at `https://github.com/liuyiting900/SAANS`.

2. Related Work

2.1 Taxonomy of Linguistic Steganography

LS methods are generally categorized into three types: selection-based, modification-based, and generation-based approaches (Fridrich 2009).

Selection-based methods choose existing sentences that convey the intended meaning without altering the original content, such as in coverless steganography (Zhang et al. 2016, 2017). The primary advantage of this approach is that the cover text remains completely natural, since no modifications are introduced. However, a significant drawback is its extremely low payload capacity, which limits its practicality in most real-world scenarios.

Modification-based methods embed secret information by making subtle changes to existing texts, such as synonym substitution (Bolshakov and Gelbukh 2004; Chen et al. 2011), syntactic restructuring (Tang and Chen 2013), or the insertion of linguistic redundancies (Low et al. 1995). However, natural language is inherently information-dense and offers limited redundancy, which significantly restricts the space available for modification. As a result, such methods often suffer from low embedding capacity when applied to text carriers.

In contrast, generation-based methods synthesize text from scratch using an LM, embedding secret information during the generation process (Wayner 1992; Chapman and Davida 1997; Ziegler, Deng, and Rush 2019; Shen, Ji, and Han 2020; Wang et al. 2023). By eliminating the reliance on existing cover texts, these methods offer greater flexibility, scalability, and concealment. Our work falls into this category and aims to enhance both the embedding capacity and security of generation-based steganography by leveraging advanced probabilistic coding and adaptive generation techniques.

2.2 Language Modeling for Generation-Based Linguistic Steganography

Early LS systems used character-level distribution models (Wayner 1992) or hand-crafted grammatical rules (Chapman and Davida 1997; Petitcolas and Katzenbeisser 2000). These methods typically leveraged features such as text spacing, character properties, or encoding characteristics at the binary level to embed secret information. However, they often produced grammatically incorrect or semantically incoherent text, making the generated content highly unnatural. Such artifacts render these methods highly vulnerable to detection by statistical analysis or deep learning-based steganalysis tools (Wen et al. 2019; Yang et al. 2019; Niu et al. 2019).

To improve the naturalness and fluency of generated text, subsequent studies introduced statistical LMs such as Markov chains (Dai et al. 2010; Moraldo 2014; Shniperov and Nikitina 2016; Luo et al. 2016; Yang et al. 2018b). These methods typically construct a state transition matrix based on publicly available text corpora and generate stego text by selecting appropriate tokens according to this matrix. However, such models are constrained by the Markov assumption, which considers only limited contextual information and fails to effectively capture long-range dependencies in natural language.

With the advancement of deep neural networks, recurrent neural networks (RNNs), such as long short-term memory (Kang, Wu, and Zhang 2020), have been widely adopted in steganographic frameworks (e.g., RNN-Stega [Yang et al. 2018a] and VAE-Stega [Yang et al. 2020]). These models enable conditionally controlled text generation and significantly improve both the fluency of the generated text and the embedding capacity. Subsequently, researchers introduced more sophisticated generative models,

such as Generative Adversarial Networks (Zhou et al. 2021) and large-scale pretrained LMs (Ziegler, Deng, and Rush 2019; Shen, Ji, and Han 2020; Wang et al. 2023; Wu et al. 2024; Kuznetsov et al. 2025; Karpov et al. 2025; Perry et al. 2025), which support context-aware generation and offer more accurate modeling of natural language distributions.

In contrast to previous studies that focused on enhancing text quality via increasingly complex language models, our approach emphasizes the integration of a robust and efficient encoding scheme with adaptive generation, all while keeping the base LM architecture intact.

2.3 Coding Methods for Generation-Based Linguistic Steganography

Information coding plays a central role in LS, serving as the mechanism that transforms a binary secret message into a sequence of tokens generated by a LM. The design of the coding scheme determines the trade-off among embedding capacity, computational efficiency, and concealment quality.

These schemes can be broadly classified into two major categories based on their underlying design principles and technical implementations: prefix-free coding methods (Yang et al. 2018a, 2020; Ziegler, Deng, and Rush 2019) and grouping-based methods (Zhang et al. 2021). Prefix-free coding methods—including HC (Huffman 1952), AC (Rissanen and Langdon 1979), and ANS (Duda 2013)—are inherently well-suited for probability alignment, enabling more effective mapping of secret bits to tokens based on the LM's output distribution. In contrast, grouping-based methods—such as ADG (Zhang et al. 2021)—encode information by recursively partitioning the token space into subsets with approximately equal cumulative probabilities, offering greater flexibility and provable security guarantees at the cost of higher complexity. Our work follows the prefix-free coding paradigm, focusing on enhancing its embedding efficiency, robustness, and adaptability for LS.

To better understand the foundation of prefix-free coding in LS, we briefly review its evolution starting from HC-based methods. Early steganographic systems often adopted coding strategies derived from HC (Huffman 1952). HC assigns shorter codes to more probable tokens based on their predicted frequency. Two variants of HC commonly used in practice are FLC and VLC (Yang et al. 2018a). Both rely on Huffman trees for token selection: FLC restricts encoding to fixed-depth paths, simplifying implementation but limiting efficiency, while VLC leverages the full variable-length structure to improve capacity, at the cost of higher decoding complexity.

To better match the output distribution of LMs, AC (Rissanen and Langdon 1979) was subsequently introduced. Unlike HC, which operates on discrete tokens, AC encodes an entire bitstream into a single real-valued interval by recursively partitioning the probability space according to symbol probabilities, thereby achieving near-optimal compression and high embedding efficiency. This capability has been leveraged in steganographic systems such as VAE-Stega (Yang et al. 2020), which integrates HC and AC to balance embedding capacity and fluency, and Neural AC (Ziegler, Deng, and Rush 2019), which embeds secret information along a continuous trajectory within the LM's predictive distribution. Although numerical precision issues in long sequences can be mitigated through renormalization, AC in steganography depends entirely on the probability intervals of candidate tokens. In highly concentrated distributions, high-probability tokens alone may be insufficient to provide the required capacity, making it necessary to include lower-probability, higher-rank tokens. While this expansion increases the theoretical capacity, it also shifts the token distribution, potentially increasing divergence and reducing fluency.

To address these limitations, we propose SA-ANS, a stack-based, bijective prefix-free coding scheme designed for efficient and adaptive mapping between secret bitstreams and token sequences. Unlike AC, SA-ANS operates entirely in the integer domain, eliminating numerical instability while preserving the benefits of probability alignment.

3. Preliminaries

In this section, we first introduce the meanings of the notations used in this article. Then, we introduce generation-based LS in Section 3.2 and its imperceptibility in Section 3.3, and summarize coding methods that can be applied to generation-based LS in Section 3.4.

3.1 Notation

We summarize all notations used throughout the article in Table 1. For clarity, the notations are grouped into the following three categories:

Language modeling and text representation. The secret bitstream to be embedded is denoted by $X = \{0,1\}^z \in \mathcal{X}$, where $\mathcal{X}$ represents the space of binary bitstreams. The LM generates a stego token sequence $S = [s_1, s_2, \ldots, s_n]$ conditioned on a shared context C. At each generation step t, the model computes the conditional probability distribution $\mathbf{P}_{\text{LM}}(s_t \mid s_{<t})$ over the vocabulary. The distribution of the generated stego sequence is denoted by $\mathbf{Q}(S)$.

ANS Encoding and Decoding Process. The ANS coder maintains an internal state χ in the form of an integer, which is continuously updated during both encoding and decoding. The frequency resolution is determined by a precision parameter $M = 2^q$, where q is the number of precision bits. For each token j in the candidate pool, an integer frequency f_j is assigned, and the cumulative distribution is computed as $\mathbf{CDF}[j] = \sum_{i=0}^{j-1} f_i$. Each

Table 1
List of notations.

Category	Symbol	Description
LM	X	Secret bitstream to be embedded, $X = \{0,1\}^z$
	S	Generated stego token sequence
	C	Shared context used to initialize the generation
	$\mathbf{P}_{\text{LM}}(s_t \mid s_{<t})$	Conditional probability from LM for next token
	$\mathbf{Q}(S)$	Distribution of the stego text sequence S
ANS	χ	Current ANS state (integer) during encoding/decoding
	M	Precision parameter: $M = 2^q$, q is coding precision
	f_j	Integer frequency of the j-th token in candidate pool
	$\mathbf{CDF}[j]$	Cumulative distribution: $\sum_{i=0}^{j-1} f_i$
	m	Embedded integer value within frequency range $[0, f_j)$
	q	Coding precision
Sampling	L	Embedding precision
	k	Size of the candidate pool
	s_t	The t-th stego token in S
	y_t	Token selected at time t via ANS decoding

segment of the secret bitstream is embedded as an integer value m within the frequency interval $[0, f_j)$.

Candidate Pool Sampling and Token Selection. At each step, the top-k tokens from the LM output distribution form the candidate pool. A token is selected from this pool according to the ANS embedding rule. The token generated at step t is denoted by s_t, and y_t refers to the corresponding symbol used during decoding. The embedding precision used during token selection is represented by L.

3.2 Linguistic Steganography

Researchers generally believe that the origins of LS can be traced back to the "prisoner's dilemma" (Simmons 1984). In this scenario, two prisoners, Alice and Bob, need to establish an indistinguishable covert communication channel under the supervision of Eve, to discuss their escape plan. Alice and Bob transmit a secret message $X \in \mathcal{X}$. Controlled by a key k in the key space $\mathcal{K}$, a mapping function $\mathsf{Emb}(\cdot)$ is used to map X to S:

$$\mathcal{X} \times \mathcal{K} \to \mathcal{S},\ \mathsf{Emb}(X, k) = S. \tag{1}$$

Under the guidance of the key $k \in \mathcal{K}$, Bob uses an extraction function $\mathsf{Ext}(\cdot)$ to extract the correct secret message X from the stego text space $\mathcal{S}$:

$$\mathcal{S} \times \mathcal{K} \to \mathcal{X},\ \mathsf{Ext}(S, k) = X. \tag{2}$$

To achieve imperceptibility, we aim to make the distribution of the generated stego text statistically indistinguishable from that of natural language. Formally, this requires the probability distribution $\mathbf{P}_{\mathcal{S}}$ over the stego text space $\mathcal{S}$ to be close to the distribution $\mathbf{P}_{\mathcal{N}}$ over natural language space $\mathcal{N}$, i.e.,

$$d\left(\mathbf{P}_{\mathcal{N}}, \mathbf{P}_{\mathcal{S}}\right) \leq \epsilon. \tag{3}$$

In practical scenarios, we approximate these distributions using $\mathbf{Q}$ for stego text and $\mathbf{P}^*_{\mathrm{LM}}$ for natural language, and adopt Total Variation Distance (TVD) and KL divergence to formally quantify imperceptibility.

3.3 The Imperceptibility of Linguistic Steganography

The security of LS primarily depends on statistical imperceptibility. To evaluate the imperceptibility performance of a steganography model, we assume the presence of a passive eavesdropper Eve. When the stego text is transmitted through a public channel, Eve observes the probability distribution $\mathbf{Q}$ of the stego text. If she can't distinguish between the arbitrarily generated stego text distribution $\mathbf{Q}$ and the real natural language distribution $\mathbf{P}^*_{\mathrm{LM}}$, then the model is considered to have good performance.

Following Dai and Cai (2019), We use the TVD to denote "imperceptibility" as follows:

$$\mathrm{TVD}(\mathbf{P}^*_{\mathrm{LM}}, \mathbf{Q}) = \frac{1}{2}\|\mathbf{Q} - \mathbf{P}^*_{\mathrm{LM}}\|_1. \tag{4}$$

Here, $||\cdot||_1$ is $\mathcal{L}_1$ norm. We approximate $\mathbf{P}^*_{\mathrm{LM}}$ using the probability distribution of sentences in the training samples $\mathbf{P}_{\mathrm{LM}}$. Further analysis yields:

$$\mathrm{TVD}(\mathbf{P}^*_{\mathrm{LM}}, \mathbf{Q}) \leq \frac{1}{2}\|\mathbf{P}^*_{\mathrm{LM}} - \mathbf{P}_{\mathrm{LM}}\|_1 + \frac{1}{2}\|\mathbf{P}_{\mathrm{LM}} - \mathbf{Q}\|_1. \tag{5}$$

In Equation (5), the first term measures the quality of the LM. We assume that the distributions of the LM and the real text are indistinguishable, so the value of the first term can be ignored. The second term represents the difference between the distribution of the generated stego text $\mathbf{Q}$ and the probability distribution of the sentences in the training sample $\mathbf{P}_{\mathrm{LM}}$.

By the Pinsker's inequality (Reid and Williamson 2009), we use the KL divergence (D_{KL}) to measure the upper bound of the TVD:

$$\frac{1}{2}\|\mathbf{P}_{\mathrm{LM}} - \mathbf{Q}\|_1 \leq \sqrt{\frac{1\mathrm{n}2}{2} D_{\mathrm{KL}}(\mathbf{Q}\|\mathbf{P}_{\mathrm{LM}})}, \tag{6}$$

where D_{KL} measures the similarity between the statistical distributional features of the stego text and the normal text, which is defined as:

$$D_{\mathrm{KL}}(\mathbf{Q}\|\mathbf{P}_{\mathrm{LM}}) = \sum_{t=1}^{n} \mathbf{Q}(s_t) \log\left(\frac{\mathbf{Q}(s_t)}{\mathbf{P}_{\mathrm{LM}}(s_t)}\right). \tag{7}$$

A smaller D_{KL} indicates a greater similarity between the distributions. When $D_{\mathrm{KL}} = 0$, the steganographic scheme achieves perfect security in terms of statistical indistinguishability, as the stego text is distributionally identical to natural language.

3.4 Coding Methods in Linguistic Steganography

Our work primarily focuses on the second term in Equation (5), which reflects how closely the stego text distribution aligns with that of the target LM. Ensuring this alignment requires a carefully designed encoding algorithm that preserves the statistical properties of the LM output distribution.

In LS, coding techniques play a crucial role in embedding secret messages into fluent and semantically plausible text. In this section, we review four representative coding methods—Huffman Coding (HC) (Huffman 1952), Arithmetic Coding (AC) (Rissanen and Langdon 1979), Adaptive Dynamic Grouping (ADG) (Zhang et al. 2021), and Asymmetric Numeral Systems (ANS) (Duda 2013)—and discuss the advantages of using ANS for steganographic applications.

Huffman Coding. HC is a classical memoryless entropy encoder that assigns variable-length bit strings to symbols based on their probability, with more frequent symbols receiving shorter codes. Although conceptually simple, HC often incurs high overhead in practical steganographic embedding due to its rigid binary tree structure and limited adaptability.

Arithmetic Coding. AC encodes an entire message into a single real-valued interval by recursively partitioning it according to symbol probabilities, achieving near-optimal compression and high embedding efficiency. While numerical precision issues in long

sequences can be mitigated with renormalization (Witten, Neal, and Cleary 1987), AC in steganography relies entirely on the probability intervals of candidate tokens. When the distribution is highly concentrated, high-probability tokens alone may not provide enough capacity, forcing the inclusion of lower-probability, higher-rank tokens. This expands capacity but shifts the distribution, increasing divergence and potentially reducing fluency.

Adaptive Dynamic Grouping. ADG encodes information by partitioning the vocabulary into groups, with each group corresponding to a specific secret message. At decoding time, the receiver determines the group to which each token belongs and extracts the associated bit sequence. While ADG provides provable security guarantees, it suffers from considerable time overhead due to repeated group construction and token mapping, which limits its scalability in high-throughput scenarios.

Asymmetric Number System. ANS is an efficient entropy coding method that compresses data by mapping symbol sequences to an integer state. Compared to AC, ANS can, in steganographic tasks, prioritize high-probability tokens while meeting embedding capacity requirements, thereby keeping the generated distribution closer to the original LM, reducing divergence, and preserving text fluency.

The variant of ANS used in this article is the range Asymmetric Numeral System (rANS) (Duda 2013). Let $W = \{w_0, w_1, \cdots, w_{K-1}\}$ be the symbol alphabet, where K denotes the number of distinct symbols. Each symbol w_j is assigned a positive integer frequency f_j, reflecting its relative probability in the data. The sum of all frequencies, $\sum_{j=0}^{K-1} f_j$, is typically set to 2^q, where q denotes the coder's precision, thereby defining the total number of internal states used for encoding.

To facilitate encoding and decoding, we define the cumulative distribution function (CDF) for each symbol w_j as:

$$\mathbf{CDF}[j] = \sum_{i=0}^{j-1} f_i, \quad \text{with } \mathbf{CDF}[0] = 0. \tag{8}$$

Let $A = [a_1, a_2, \ldots, a_n]$ be the sequence of symbols to be encoded. At time step t, suppose the current rANS state is χ, and the symbol to be encoded is a_t, whose index in the alphabet is j. Based on the frequency and cumulative frequency of the symbol, the rANS encoding updates the state as follows:

$$\chi' = \left\lfloor \frac{\chi}{f_j} \right\rfloor \cdot 2^q + \mathbf{CDF}[j] + (\chi \bmod f_j). \tag{9}$$

Decoding an rANS state χ' involves two steps: identifying the symbol to decode, and recovering the previous state.

The first step is to compute an auxiliary value u from the current state:

$$u = \chi' \bmod 2^q. \tag{10}$$

Next, the symbol index j is identified as the unique value satisfying the condition $\mathbf{CDF}[j] \leq u < \mathbf{CDF}[j+1]$. The decoded symbol a_t is the one whose index in the alphabet is j.

A = ["0","1","1"], W= ["0", "1"], f("0") = 1, f("1") = 2, $\chi_0 = 1$

x	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
m_1 = "0"	0			**1**			2			3
m_2 = "1"		0	1		2	**3**		4	**5**	

Result of ANS encoding: $\chi_3 = 8$

Figure 2
An illustration of rANS encoding with a binary alphabet {"0", "1"}, where the symbol frequencies are $f_{"0"} = 1$ and $f_{"1"} = 2$, yielding a CDF of $[0, 1, 3]$. Encoding the symbol sequence {"0", "1", "1"} proceeds as follows: $\chi_1 = \lfloor 1/1 \rfloor \cdot 3 + 0 + 0 = 3$, $\chi_2 = \lfloor 3/2 \rfloor \cdot 3 + 1 + 1 = 5$, and $\chi_3 = \lfloor 5/2 \rfloor \cdot 3 + 1 + 1 = 8$. The final state $\chi_3 = 8$ compactly encodes the original input sequence.

Once the symbol index j is known, the previous rANS state χ can be recovered using the following decoding step:

$$\chi = f_j \cdot \left\lfloor \frac{\chi'}{2^q} \right\rfloor + (u - \mathbf{CDF}[j]). \tag{11}$$

This formulation ensures that the rANS coder offers an efficient and reversible representation of discrete symbol sequences by maintaining a single integer-valued state. This state serves as a compact buffer that dynamically grows during encoding and shrinks during decoding, enabling streaming-friendly operation. A concrete example of the rANS process is illustrated in Figure 2.

4. Linguistic Steganography via Self-Adjusting Asymmetric Number System

In this section, we first present an overview of the SA-ANS scheme (Section 4.1), then describe the specific tasks in the preparation phase (Section 4.2), and finally discuss the embedding and extraction algorithms in detail in Sections 4.3 and 4.4, respectively.

4.1 SA-ANS Overview

The SA-ANS framework enables covert communication by leveraging a shared LM to embed a secret bitstream into a sequence of natural language tokens. Both the sender and the receiver are assumed to possess access to a common LM and share a predefined context sequence C, which acts as the semantic seed for text generation.

To transmit a secret message X, the sender uses a SA-ANS encoder, embedding bits incrementally into token selection. The generation process is guided by the LM's next-token distribution, which is truncated via nucleus sampling. The resulting distribution is then transformed into an integer frequency table compatible with ANS encoding. The resulting stego text S is fluent and semantically coherent, rendering it indistinguishable from naturally generated text.

Upon receiving the stego sequence S, the receiver reconstructs the candidate distributions and decoding tables using the same LM and context C. The embedded bitstream X is then recovered using the SA-ANS decoding algorithm, which reverses the encoding process by tracking ANS state transitions and recovering embedded bits from each token.

This scheme ensures strong imperceptibility, precise encoder–decoder synchronization, and full reversibility of the embedded message. A detailed description of the SA-ANS protocol are presented in the following sections.

4.2 Preparatory Stage

In the preparatory stage, the sender and receiver agree to use a publicly available LM. They also share a predefined context C to establish a common semantic starting point. By using the same LM and context, both parties ensure synchronization, enabling the sender to embed and the receiver to extract the secret message in a reliable and covert manner.

To initialize the context C, the sender and receiver may agree on a fixed prefix text, such as a standard prompt or sentence. The selected context should preserve linguistic naturalness while providing sufficient entropy—that is, it should lead the LM to generate a diverse distribution over plausible next tokens. A higher-entropy context increases the number of viable token choices at each step, enabling more bits of information to be embedded during probabilistic token selection without compromising the fluency or coherence of the generated text.

4.3 Embedding Stage

The embedding procedure consists of five core components: *(1) message preprocessing*, *(2) candidate pool construction*, *(3) ANS decoding*, *(4) state update*, and *(5) token generation*. We detail each step below.

Message Preprocessing. The embedding process begins by interpreting the first L bits of the secret bitstream $X = \{x_1, x_2, \ldots, x_L, \ldots\}$ as a binary integer. The initial ANS state χ is defined as:

$$\chi = \sum_{i=1}^{L} x_i \cdot 2^{L-i}, \tag{12}$$

here, L is the embedding precision, indicating how many secret bits are injected per step. Separately, the ANS coding precision is defined by q, which sets the coding range $M \leftarrow 2^q$ and determines the resolution of the frequency table used in rANS decoding.

Candidate Pool Construction. At each embedding step t, the current context C (i.e., the previously generated tokens) is fed into the LM to compute the next-token probability distribution:

$$\mathbf{P}_{\mathrm{LM}} \leftarrow \mathsf{softmax}(\mathsf{LM}(C)/T), \tag{13}$$

where T is a temperature parameter controlling the distribution's sharpness.

To construct a compact candidate pool, we apply nucleus sampling by selecting the smallest integer k such that:

$$\sum_{i=0}^{k-1} \mathbf{P}_{\mathrm{LM}}(m_i) \geq 1 - \nu. \tag{14}$$

To promote convergence, we apply a decaying schedule to ν, defined as:

$$\nu_\mu = \frac{\nu_0}{\mu^2}, \tag{15}$$

where μ denotes the index of the current message segment.

Borrowing Equation (6), this strategy ensures that the ℓ_1 divergence between the original distribution $\mathbf{P}_{\text{LM}}$ and its truncated version $\mathbf{Q}$ satisfies:

$$\frac{1}{2}\|\mathbf{P}_{\text{LM}} - \mathbf{Q}\|_1 \leq \sqrt{\frac{\pi^2 \ln 2}{12}\nu_0}. \tag{16}$$

Let $\mathcal{M}_t = \{m_0, m_1, \ldots, m_{k-1}\}$ be the candidate token pool. The probabilities $p_i = \mathbf{Q}(m_i)$ are scaled into integer frequencies $f_i = M \cdot p_i$ such that $\sum_i f_i = M$. A CDF is then constructed:

$$\mathbf{CDF}[0] = 0, \quad \mathbf{CDF}[j+1] = \sum_{i=0}^{j} f_i. \tag{17}$$

ANS Decoding. Given the current ANS state χ, compute the modulo residue:

$$u = \chi \bmod M, \tag{18}$$

then find the index j such that:

$$\mathbf{CDF}[j] \leq u < \mathbf{CDF}[j+1]. \tag{19}$$

This index determines the next token m_j to be emitted.

State Update. The ANS state is updated using the reverse rANS transform:

$$\chi' = \left\lfloor \frac{\chi}{M} \right\rfloor \cdot f_j + (u - \mathbf{CDF}[j]). \tag{20}$$

Then, exactly L bits $b_1, b_2, \ldots, b_L \in \{0, 1\}$ are appended to χ', where each b_i is either the next bit from the bitstream or zero (if the stream is exhausted). The updated state becomes:

$$\chi'' = 2^L \cdot \chi' + \sum_{i=1}^{L} b_i \cdot 2^{L-i}, \tag{21}$$

where b_i denotes the next bit from the bitstream if available, or 0 otherwise.

Token Generation. The selected token $y_t = m_j$ is appended to the output sequence. Let $S = [y_1, \ldots, y_{t-1}]$ and C denote the current decoding context. Then,

$$S' = S \cup \{y_t\}, \quad C' = C \circ y_t, \tag{22}$$

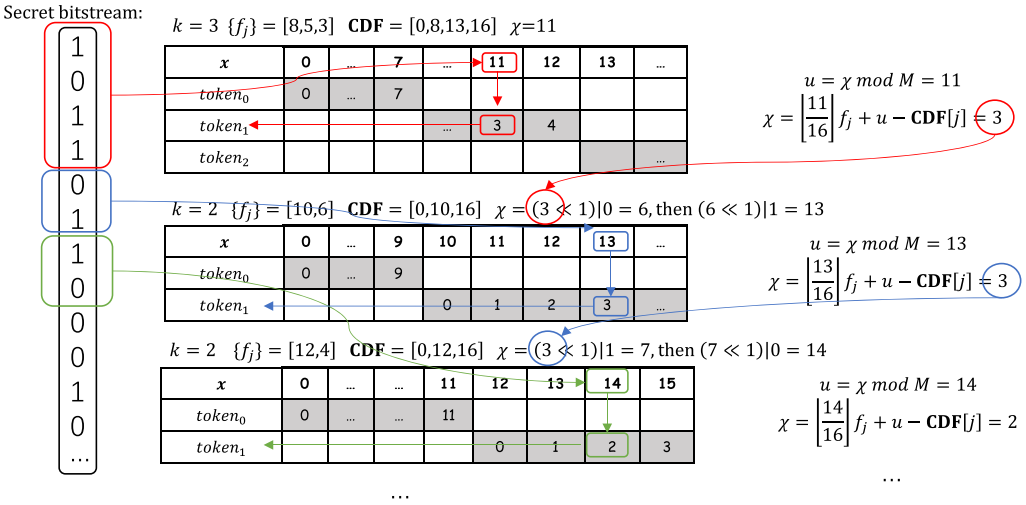

Figure 3
Step-by-step illustration of the SA-ANS embedding process using a concrete example. The encoder starts with a secret bitstream and initial state $\chi = 11$. At each step, it computes $u = \chi \bmod M$ to locate a token from the CDF intervals, emits the corresponding token, and updates the state via reverse rANS transformation. For instance, $u = 11$ falls into interval $[8, 13)$, emitting the token at index $j = 1$ and updating χ to 3. After appending new bits, the updated state is used in the next step. This process continues until all bits are embedded.

where $\cup$ denotes sequence extension and $\circ$ represents context concatenation. The decoding process proceeds to the next step, and this procedure is repeated until all message bits have been consumed.

Figure 3[2] and Appendix A provide a step-by-step illustration of the SA-ANS embedding procedure.

Degenerate Case: $\nu = 0$. As a special case of the above, when the nucleus threshold is set to $\nu = 0$, no adaptive truncation is performed. Instead, a fixed top-k strategy—as used in AC (Ziegler, Deng, and Rush 2019)—is employed to construct the candidate pool. Consequently, SA-ANS degenerates into a non-adaptive variant of *ANS steganography* with a static vocabulary constraint, losing the dynamic flexibility of nucleus sampling.

4.4 Extraction Stage

The extraction stage in SA-ANS is responsible for reconstructing the original secret bitstream from the stego token sequence. With access to the same pretrained LM and shared context, the decoder deterministically regenerates the candidate token distribution at each decoding step.

2 In this figure and extraction example, L is not fixed, demonstrating that consistent L across encoding and decoding rounds ensures correct recovery. In the article, a fixed L is used for analytical clarity and reproducibility.

Reconstructing the CDF. At decoding step t, the decoder receives the previously generated context C and computes the corresponding logits. These logits are scaled by a temperature parameter T and passed through a softmax function to yield the next-token distribution $\mathbf{P}_{\text{LM}}$.

Consistent with the encoding process, a nucleus sampling strategy with a decaying threshold $\nu_\mu = \nu_0/\mu^2$ is applied to select the top-k tokens forming the candidate set. The probabilities of these tokens are then normalized and quantized into integer frequencies $\{f_j\}$ such that $\sum_j f_j = 2^q$, where q denotes the ANS coding precision. The CDF is subsequently constructed based on these frequencies.

Reconstructing the ANS State. Let γ_t denote the ANS state carried from the previous step. The decoder identifies the index j such that the current stego token y_t matches the j-th token in the reconstructed candidate pool $\mathcal{M}_t$. Using the associated frequency f_j and base value $\mathbf{CDF}[j]$, the current ANS state χ is recovered via the inverse ANS update:

$$\chi = \left\lfloor \frac{\gamma_t}{f_j} \right\rfloor \cdot M + \mathbf{CDF}[j] + (\gamma_t \bmod f_j). \tag{23}$$

This step effectively traces back the internal encoding state corresponding to the generated token.

Bit Extraction via Fixed-Length Shifting. From the reconstructed state χ, the embedded bits are extracted by reading the L least significant bits (LSBs) in reverse order. Let $\gamma_1, \gamma_2, \ldots, \gamma_L \in \{0, 1\}$ denote the extracted bits. Then,

$$\gamma_i = \left\lfloor \frac{\chi}{2^{i-1}} \right\rfloor \bmod 2, \quad \text{for } i = 1, 2, \ldots, L. \tag{24}$$

This procedure ensures a constant embedding rate of L bits per token, regardless of the magnitude of the ANS state. The extracted bits are collected in reverse order and are flipped at the end of decoding to restore the original sequence. The updated state χ is stored as γ_{t+1} and passed to the next decoding step.

By repeating this process for each token in the stego sequence, the complete secret bitstream $X = [x_1, x_2, \ldots, x_z]$ is successfully recovered.

Figure 4 and Appendix B provide a step-by-step illustration of the SA-ANS extraction procedure, in which the original bitstream is progressively recovered from the stego token sequence by reversing the ANS encoding process and applying parity-based bit retrieval from reconstructed ANS states.

5. Analysis of the SA-ANS

We begin by evaluating the robustness of the proposed SA-ANS scheme in Section 5.1. Its state-driven and self-synchronizing design enables reliable message recovery even under partial generation or truncation.

Next, in Section 5.2, we analyze the computational efficiency of SA-ANS. We show that its runtime grows linearly with the length of the embedded bitstream and remains comparable to existing steganographic approaches.

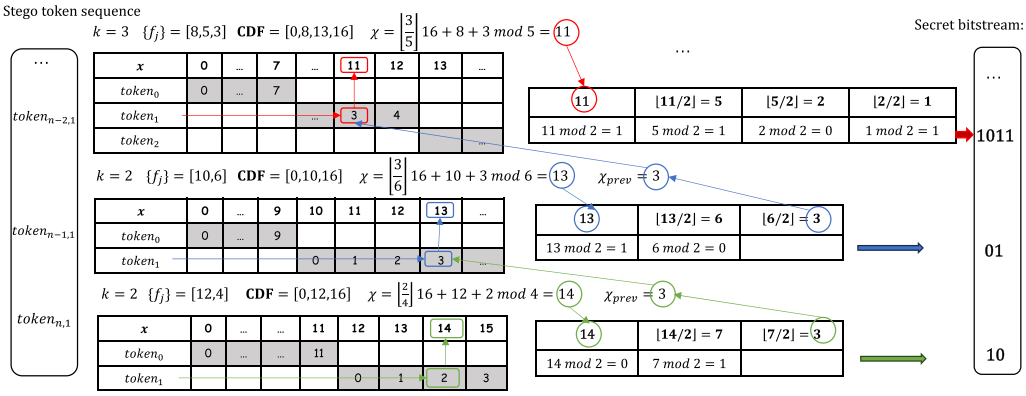

Figure 4
Reverse bit extraction in SA-ANS steganography. Starting from the final ANS state χ, each bit is sequentially recovered by checking the parity of χ: even values yield 0, odd values yield 1. After extracting each bit, the state is right-shifted via integer division by 2. For example, beginning with $\chi = 14$, the first bit is 0 (even), and the next state is 7; since 7 is odd, the next bit is 1, resulting in the bit sequence "10". Dividing 7 by 2 yields the previous ANS state $\chi_{\text{prev}} = 3$. This parity-based approach ensures reversible decoding and accurate recovery of the original bitstream.

5.1 Robustness Analysis

Structural robustness refers to the ability to recover the embedded secret message (or its prefix) even when the stego text is truncated or partially lost. This property is vital for real-world applications involving streaming, interruption, or lossy transmission.

Robustness of SA-ANS. SA-ANS demonstrates structural robustness through its explicit, integer-based state mechanism. Unlike AC–based approaches (e.g., SAAC [Shen, Ji, and Han 2020]), which encode the entire message into a single compressed interval, SA-ANS maintains a discrete state χ that evolves deterministically at each token generation step, guided by the model-derived frequency table.

This alignment between token sequence and state transitions enables token-by-token reversibility: Given the final state χ, decoding can proceed in reverse to reconstruct the embedded bitstream prefix up to the current token. No external metadata or access to future tokens is required. This per-token invertibility allows SA-ANS to support partial decoding and enables robustness in streaming or interrupted generation scenarios. As illustrated in Figure 4, decoding terminates gracefully with the entire decoded prefix preserved.

Comparison with SAAC. By contrast, SAAC encodes the bitstream into a progressively narrowing floating-point interval $[l, u)$, embedding the message in the position of a final value $z \in [l, u)$. In the event of interruption, only the most significant bits shared by l and u are guaranteed to be recoverable. Any remaining bits embedded in finer subintervals are lost. As shown in Figure 5, this restricts recovery to coarse binary prefixes and prevents full reconstruction.

The global nature of AC further limits its robustness: Decoding cannot resume unless the full token sequence is available. This lack of self-synchronization makes

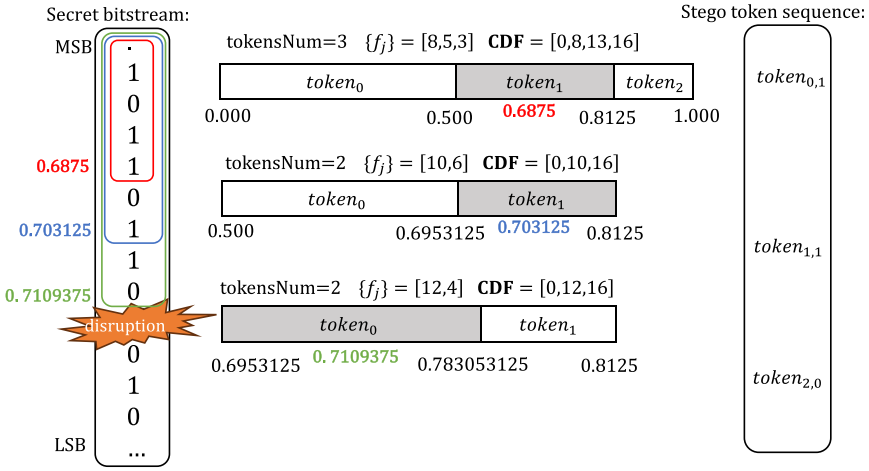

Figure 5
Illustration of AC-based steganographic embedding. The secret bitstream is progressively compressed into a narrowing interval by selecting tokens according to the predicted probability distribution (quantized into discrete CDFs). If the encoding process is interrupted (e.g., due to transmission failure), only the tokens generated up to that point (e.g., $j = 0$ in the final row) are retained. Given the final known interval $(0.6953125, 0.783053125)$, only the common binary prefix "101101" can be reliably recovered. The remaining bits ("10"), embedded in the unresolved subinterval, are lost due to insufficient floating-point precision.

SAAC inherently fragile under partial generation and incompatible with streaming settings.

In contrast, SA-ANS embeds message bits into a discrete, locally invertible integer state χ at each step. This state evolves incrementally with each token and can be reversed independently. As a result, the decoder can recover the complete bitstream prefix by retracing state transitions, without requiring access to future tokens or the full stego sequence.

5.2 Complexity Analysis

We analyze the computational complexity of the SA-ANS embedding procedure by decomposing it into its constituent steps. Let z denote the length of the secret bitstream, L the embedding precision (i.e., the number of bits embedded per token), V the vocabulary size of the LM, and k the size of the truncated candidate set after top-k filtering.

State Initialization. The initial ANS state χ is computed from the first L bits of the secret bitstream in $\mathcal{O}(L)$ time.

Per-Step Embedding. Each embedding step corresponds to the generation of one token and embeds exactly L secret bits. The operations involved in one step are as follows:

- Language model inference: $\mathcal{O}(V)$,
- Top-k filtering and normalization: $\mathcal{O}(k)$,
- CDF construction via cumulative sum: $\mathcal{O}(k)$,

- Token selection via binary search: $\mathcal{O}(\log k)$,
- Integer-based state update: $\mathcal{O}(1)$,
- Bit-packing via left-shift and accumulation: $\mathcal{O}(L)$.

The total complexity per embedding step is therefore:

$$\mathcal{T}_{\text{step}} = \mathcal{O}(V + k + \log k + L). \tag{25}$$

Given that $V \gg k, L$ in practical settings, the step complexity simplifies to $\mathcal{T}_{\text{step}} = \mathcal{O}(V)$, dominated by the LM's forward pass.

Total Embedding Complexity. Since each token encodes exactly L bits, the total number of tokens required to embed a bitstream of length z is:

$$N = \left\lceil \frac{z}{L} \right\rceil = \mathcal{O}(z). \tag{26}$$

The total time complexity of the entire embedding process is then:

$$\mathcal{T}_{\text{total}} = N \cdot \mathcal{T}_{\text{step}} = \mathcal{O}(z \cdot V). \tag{27}$$

This yields a constant-rate embedding scheme with linear complexity in the bitstream length and vocabulary size, allowing predictable runtime and capacity scaling.

To summarize the computational efficiency of different steganographic encoding schemes, we provide a comparative analysis in Table 2. The table reports the per-token computational cost, the number of tokens required to embed a bitstream of length z, and the resulting total time complexity for each method.

SA-ANS achieves linear overall complexity, with each token generated via a constant-time update of an integer state and a language model forward pass. Thanks to its incremental and discrete encoding structure, SA-ANS supports efficient per-token processing without global recomputation. In contrast, SAAC relies on global interval compression using AC. Although its theoretical per-token cost is similar to SA-ANS, it lacks token-level reversibility and incurs structural fragility under interruption. Finally, HC methods exhibit higher per-token cost due to the need to rebuild the tree at each generation step, resulting in $\mathcal{O}(z \cdot V \log V)$ overall complexity.

This comparison highlights the scalability advantages of SA-ANS: It supports efficient, robust, and per-token reversible embedding, with linear complexity in the bitstream length. By avoiding global re-encoding and enabling constant-rate local updates,

Table 2
Computational complexity comparison of different steganographic encoding methods.

Method	Per Token Cost	Total Tokens	Total Complexity
Huffman (Yang et al. 2018a)	$\mathcal{O}(V \log V)$	$\mathcal{O}(z)$	$\mathcal{O}(z \cdot V \log V)$
SAAC (Shen, Ji, and Han 2020)	$\mathcal{O}(V)$	$\mathcal{O}(z)$	$\mathcal{O}(z \cdot V)$
SA-ANS (Ours)	$\mathcal{O}(V)$	$\mathcal{O}(z)$	$\mathcal{O}(z \cdot V)$

SA-ANS is particularly well-suited for large-scale deployment, real-time inference, and streaming steganographic applications.

6. Experimental Results and Analysis

This section begins by discussing the selection of key parameters and analyzing potential failure modes of the proposed SA-ANS scheme. These design considerations underpin the subsequent experiments and are essential for ensuring reliable encoding and decoding.

We then evaluate the performance of SA-ANS through comprehensive experiments across multiple dimensions, including embedding efficiency, text quality, statistical plausibility, discourse coherence, anti-steganalysis robustness, and human evaluation.

Additionally, we examine the influence of different LMs on the overall performance of the steganographic system.

6.1 Parameter Selection and Failure Modes

SA-ANS introduces two key precision parameters: the coding precision q, which determines the resolution of the frequency table as M, and the embedding precision L, which controls the number of bits injected into the ANS state per token.

When q is small, the integer frequency f_i assigned to each candidate token is computed by quantizing the softmax probabilities:

$$f_i = \left\lfloor p_i \cdot M + \frac{1}{2} \right\rfloor, \tag{28}$$

where $p_i \in [0, 1]$ is the normalized probability of token i, and $M = 2^q$ denotes the frequency resolution. At low coding precision (i.e., small M), the frequency table becomes too coarse to capture fine-grained distinctions between tokens. In this case, even nonzero probabilities may be quantized to zero. Specifically, when:

$$p_i < \frac{0.5}{M} \quad \Rightarrow \quad f_i = 0, \tag{29}$$

the corresponding token is effectively discarded during decoding. This truncation reduces the effective vocabulary size and increases the risk of invalid state-to-token mappings, often resulting in fallback behavior or decoding failure.

Furthermore, the decoding process relies on locating a token index j such that $\mathbf{CDF}[j] \leq u < \mathbf{CDF}[j+1]$, where $u = \chi \bmod M$. An excessively large M (high q) leads to a very large and sparse frequency table. This significantly increases memory consumption and computational overhead.

Figure 6 empirically illustrates this behavior: As the coding precision q increases, the self-perplexity (SPPL) drops sharply. When $q < 15$, the system frequently selects low-probability tokens or falls back, resulting in incoherent outputs and high perplexity. Once $q \geq 30$, the token distribution is sufficiently resolved and decoding stabilizes. Based on this observation, we set $q = 30$ in all experiments to ensure accurate state transitions and robust decoding.

For the embedding precision, we evaluate $L \in \{4, 8, 16\}$, which defines the number of bits embedded per token and enables a trade-off between capacity and output quality. We constrain L to be a multiple of 4 to match the 8-bit ASCII encoding of characters,

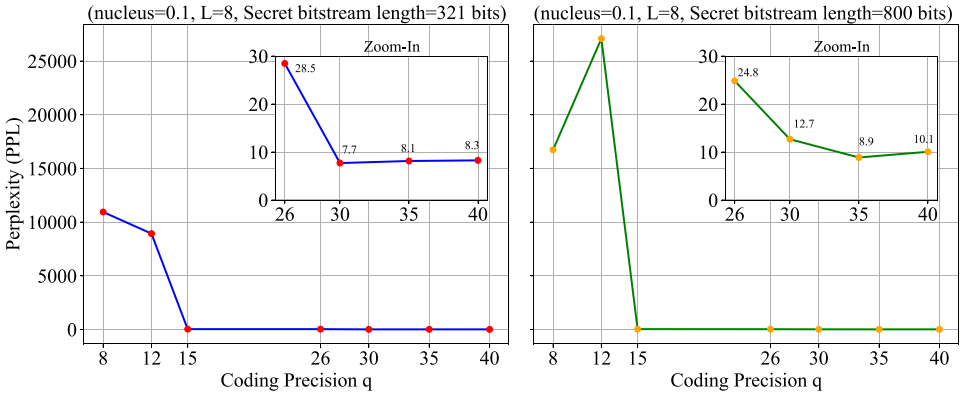

Figure 6
Influence of coding precision q on SPPL under SA-ANS embedding. The figure compares performance for a 321-bit message (left) and an 800-bit message (right). A critical degradation is observed when $q < 15$, with decoding stabilizing as SPPL converges sharply for $q \geq 30$.

ensuring alignment with character boundaries and avoiding unnecessary zero-padding in the bitstream.

6.2 Experimental Setup

Synchronization Assumptions. Most LM-based steganographic methods implicitly assume that the sender and receiver share an identical LM, including the architecture, parameters, vocabulary, and tokenization behavior. Such synchronization is essential to ensure consistent decoding results. However, in real-world deployment, this assumption may be challenged by various factors such as hardware differences (e.g., CPU vs. GPU computation), floating-point precision errors, library version mismatches, or nondeterministic operations in deep learning frameworks. Even minor numerical discrepancies can lead to divergence in decoding, especially when sampling-based generation is employed, thereby affecting the accurate extraction of hidden information.

To mitigate these potential issues, all experiments in this article are conducted in a controlled environment, using the same model implementation and fixed decoding parameters. We further adopt greedy decoding to minimize randomness during generation, ensuring stable and reproducible evaluation across different systems.

Datasets. We use four benchmark datasets to demonstrate the effectiveness of SA-ANS: **Twitter** (Go, Bhayani, and Huang 2009), **IMDB**[3] (Maas et al. 2011), **News**[4] (Hermann et al. 2015), and **COVID-19** (Wang et al. 2020).

- Twitter: The sentiment 140 dataset consists of 1.6 million tweets from the Twitter API.[5] Twitter texts are brief, highly informal, and lack strict grammar.

3 https://www.imdb.com/.
4 https://www.kaggle.com/snapcrack/all-the-news.
5 https://twitter.com/.

Table 3
Statistics of datasets.

Datasets	Twitter	IMDB	News	COVID-19
Num. of Sentences	2.64	1.28	1.96	2,000
Avg. Num. of Words	9.68	19.94	22.24	24.21
Avg. Num. of Bits	90.02	183.45	211.08	308.65

- IMDB: This dataset includes longer reviews (around 15 words) on various topics.
- News: Contains 143,000 articles from 15 American publications, mainly covering political topics.
- COVID-19: A medical dataset focused on COVID-19 research.

These diverse datasets provide a comprehensive evaluation of our model's ability to mimic human writing styles. Table 3 summarizes the dataset statistics.

Compared Methods. We compare SA-ANS with several state-of-the-art linguistic steganography methods, including RNN-Stega (Yang et al. 2018a), AC (Ziegler, Deng, and Rush 2019), SAAC (Shen, Ji, and Han 2020), and ADG (Zhang et al. 2021). Additionally, we include ANS steganography as a baseline, which is a non-adaptive ANS-based method equivalent to SA-ANS with $\nu = 0$.

Parameter Setting. All primary experiments were conducted using the OpenAI GPT-2 medium LM (345M parameters), implemented with PyTorch 1.4.0 and HuggingFace Transformers. No additional fine-tuning was applied. To assess the generalizability of our method across different architectures, we extended the evaluation in our ablation studies to include more advanced models, specifically, LLaMA-3.1-8B (Huang et al. 2024) and Qwen-3-8B (Qwen Team 2024).

For RNN-Stega, we evaluated Huffman tree depths $H \in \{3, 5, 7\}$. For AC baselines, candidate pool sizes were set to {300, 600, 900}, consistently using 26-bit fixed-point precision. For SAAC, we applied imperceptibility thresholds $\nu \in \{0.1, 0.05, 0.01\}$ and used AC with a fixed-point precision of 26 bits.

For ANS, based on the analysis in Section 6.1, we set the coding precision to $q = 30$ and evaluated embedding precisions $L \in \{4, 8, 16\}$, where each token deterministically encodes L bits per step, using a fixed candidate pool size of $k = 900$. SA-ANS adopts the same configuration but further incorporates adaptive control via nucleus sampling with thresholds $\nu \in \{0.01, 0.05, 0.10\}$, allowing a flexible trade-off between embedding capacity and text fluency during generation.

Evaluation Metrics. We comprehensively evaluate the performance of our method across five dimensions: *embedding efficiency*, *text quality*, *statistical analysis*, *anti-steganalysis*, and *human evaluation*.

Embedding Efficiency. This metric measures the amount of secret information carried by the generated text. We adopt the **Embedding Rate (ER)**, defined as the average number of secret bits embedded per word (bits per word, bpw):

$$\mathrm{ER} = \frac{1}{N}\sum_{i=1}^{N}\frac{b_i}{l_i}, \tag{30}$$

where N is the total number of stego sentences, b_i is the number of embedded bits in the i-th sentence, and l_i is the number of tokens in that sentence.

Text Quality. This metric evaluates the fluency of stego text using automatic measures such as perplexity (PPL). PPL primarily reflects the statistical naturalness of text as perceived by a LM and can indirectly capture certain aspects of grammatical correctness and semantic coherence. We adopt two metrics: **Perplexity (PPL)** (Mikolov et al. 2010), which quantifies how well a LM predicts a sequence:

$$\mathrm{PPL} = 2^{-\frac{1}{n}\log \mathbf{P}_{\mathrm{LM}}(s_1,s_2,\ldots,s_n)}, \tag{31}$$

where n is the sentence length. Lower PPL values indicate greater fluency and closer alignment with the statistical properties of natural text. We report both self-perplexity (SPPL, computed by the generation model itself) and GPT-Neo-1.3B PPL. ΔPcs (Yang et al. 2020), the absolute difference in SPPL between stego text and its surrounding context, is computed as:

$$\Delta\mathrm{Pcs} = |\mathrm{SPPLstego} - \mathrm{SPPL}_{\mathrm{context}}|, \tag{32}$$

which reflects the degree of distortion introduced by embedding.

Statistical Analysis. This metric evaluates statistical imperceptibility, defined as the similarity between stego text generated by a LM and natural text generated without hidden information. The similarity is quantified by comparing their token distributions. We adopt two distributional divergence measures: **KL Divergence (D_{KL})** (Yang et al. 2018a), which measures the discrepancy between the empirical distribution $\mathbf{Q}$ and the language model distribution $\mathbf{P}_{\mathrm{LM}}$; and **Jensen–Shannon Divergence (D_{JS})** (Li et al. 2023), a symmetric variant of D_{KL} that captures bidirectional similarity between two distributions:

$$D_{\mathrm{JS}} = \frac{1}{2}D_{\mathrm{KL}}\left(\mathbf{Q}(v_x)\middle|\frac{\mathbf{Q}(v_x)+\mathbf{P}_{\mathrm{LM}}(v_y)}{2}\right) + \frac{1}{2}D_{\mathrm{KL}}\left(\mathbf{P}_{\mathrm{LM}}(v_y)\middle|\frac{\mathbf{Q}(v_x)+\mathbf{P}_{\mathrm{LM}}(v_y)}{2}\right). \tag{33}$$

Anti-Steganalysis. This metric assesses imperceptibility from the detection perspective, defined as the difficulty for automated steganalysis models to distinguish stego text from natural text. We measure detection resistance using three strong deep learning-based LS detectors: FastText (Joulin et al. 2016), TextRNN (Liu, Qiu, and Huang 2016), and DPCNN (Johnson and Zhang 2017). The evaluation metrics are: **Accuracy (Acc)**, the proportion of correctly predicted labels:

$$\mathrm{Acc} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}, \tag{34}$$

and **F1-score (F1)**, the harmonic mean of precision and recall:

$$\mathrm{F1} = \frac{2 \times \mathrm{TP}}{2 \times \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}, \tag{35}$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively.

Human Evaluation. This metric measures text quality from the human perspective, serving as a complement to automatic measures such as PPL. We adopt the Human Judgment Score, defined as:

$$\mathrm{Score} = \frac{N_{\mathrm{acc}}}{N_{\mathrm{total}}}, \tag{36}$$

where N_{acc} denotes the number of stego sentences judged acceptable by human annotators, and N_{total} is the total number of evaluated stego samples.

6.3 Embedding Efficiency

A key advantage of the ANS-based steganographic framework is its ability to precisely control the embedding rate via the embedding precision parameter L. Since each generated token deterministically injects L bits into the ANS state, the theoretical embedding rate is simply given by $\mathrm{ER} = L$.

This deterministic capacity contrasts with entropy-based methods such as HC and AC, where the number of bits embedded per token varies with the output distribution of the LM. Flatter, high-entropy distributions allow for more bits per token, while peaked, low-entropy distributions limit the available capacity. As a result, their effective embedding rate fluctuates and cannot be precisely controlled.

A fixed embedding rate brings several practical benefits. First, it enables accurate estimation of the number of tokens required to encode a message of length z, which is particularly useful in applications with strict length constraints or interactive generation scenarios. Second, it simplifies system design and debugging, as both encoder and decoder operate under a shared, deterministic embedding budget.

However, this rigidity may also introduce limitations. In low-entropy contexts—where the LM output distribution is highly peaked—the fixed rate may necessitate selecting low-probability tokens to satisfy bit constraints, potentially degrading fluency or increasing perplexity. In contrast, adaptive methods such as AC dynamically adjust the embedding rate to match distributional entropy, often resulting in more natural outputs.

6.4 Text Quality

To evaluate the quality of the generated stego texts, we conduct experiments across four datasets, with all models embedding secret messages of 480 bits. The evaluation considers three key metrics: SPPL, GPT-Neo-1.3B perplexity, and ΔPcs. Lower values across these metrics indicate better performance.

As shown in Table 4 and Figure 7, SA-ANS outperforms most baseline methods in text quality, consistently achieving lower SPPL and GPT-Neo-1.3B PPL while maintaining a competitive embedding rate. For instance, on the *News* dataset, SA-ANS achieves

Table 4
Text quality comparison of stegos generated by SA-ANS, ANS, and baselines from secret messages of length 480-bit. "↑" is that the higher the value, the better the result, and "↓" is that the lower the value, the better the performance. **Bold** is the best result.

Datasets		Twitter			IMDB			News			COVID-19		
Methods	**Param**	ER ↑	SPPL ↓	ΔPcs ↓	ER ↑	SPPL ↓	ΔPcs ↓	ER ↑	SPPL ↓	ΔPcs ↓	ER ↑	SPPL ↓	ΔPcs ↓
RNN-Stega	$H = 3$	2.35	11.70	8.49	2.30	11.44	8.23	2.47	21.10	17.89	2.46	11.52	8.31
	$H = 5$	2.79	10.21	7.00	3.34	16.48	13.27	3.82	23.68	20.47	3.14	12.97	9.76
	$H = 7$	5.01	58.39	55.18	4.51	34.92	31.71	4.48	31.22	28.01	5.11	46.37	43.16
ADG	–	2.31	34.44	31.23	1.65	11.01	7.80	2.12	41.68	38.47	1.82	12.07	8.86
AC	$k = 300$	4.37	23.97	20.76	4.02	19.92	16.71	4.66	29.49	26.28	2.31	5.49	2.28
	$k = 600$	4.11	19.06	15.85	4.28	21.52	18.31	2.82	7.91	4.70	4.26	21.31	18.10
	$k = 900$	3.62	13.17	9.96	3.00	8.39	5.18	4.23	20.13	18.37	5.51	49.24	46.03
SAAC	$\nu = 0.01$	5.07	36.16	32.95	5.23	42.86	39.65	5.39	44.74	41.53	5.96	69.49	66.28
	$\nu = 0.05$	4.88	25.52	22.31	4.50	25.37	22.16	5.27	40.64	37.43	5.17	39.01	35.80
	$\nu = 0.10$	4.49	24.96	21.75	4.42	25.89	22.68	5.15	38.23	35.02	4.57	26.57	23.36
SA-ANS	$\nu = 0.00$ (ANS)	4	8.49	5.28	4	17.72	14.51	4	9.44	6.21	4	19.06	15.85
		8	18.12	14.91	8	34.17	30.96	8	13.52	10.31	8	55.80	52.59
		16	38.11	33.90	16	50.82	47.61	16	48.96	45.74	16	75.54	72.33
	$\nu = 0.01$	4	13.61	10.40	4	18.88	15.67	4	19.67	16.46	4	15.02	11.81
		8	17.61	14.40	8	20.58	17.37	8	21.02	17.81	8	63.25	60.04
		16	38.09	34.88	16	26.38	23.17	16	21.96	18.75	16	47.12	43.91
	$\nu = 0.05$	4	**5.12**	**1.91**	4	16.76	13.55	4	14.83	11.62	4	**4.79**	**1.58**
		8	24.13	20.92	8	27.29	24.08	8	24.56	21.35	8	32.18	28.97
		16	13.04	9.83	16	25.18	21.97	16	26.60	23.39	16	50.47	47.26
	$\nu = 0.10$	4	7.64	4.43	4	**5.03**	**1.82**	4	**3.35**	**0.14**	4	17.13	13.92
		8	9.39	6.18	8	14.28	11.07	8	15.96	12.75	8	17.23	14.02
		16	7.73	4.52	16	15.59	12.38	16	18.86	15.65	16	37.69	34.48

the lowest SPPL (3.35) under the configuration of $\nu = 0.10$ and $L = 4$, demonstrating its ability to preserve high text quality even at relatively high embedding rates.

Moreover, as illustrated in the right panel of Figure 7, all methods exhibit a clear trade-off between embedding rate and perplexity—higher embedding rates tend to result in increased PPL. This trend is particularly evident among traditional methods: RNN-Stega performs well only at low embedding rates but suffers from instability as

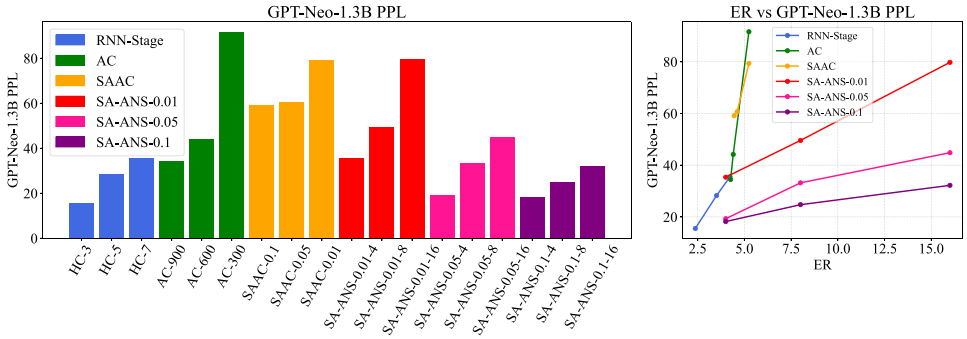

Figure 7
Left: Bar chart of GPT-Neo-1.3B PPL for different methods and parameter settings (method-parameter). Each color indicates a different method: RNN-Stega, AC, SAAC, and SA-ANS. Right: Line chart showing the relationship between ER and GPT-Neo-1.3B PPL.

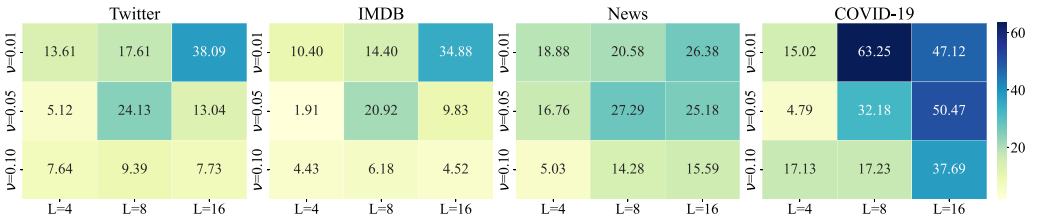

Figure 8
SPPL across datasets under different L and ν.

ER increases; and while AC and SAAC support higher ERs, they also lead to substantial increases in both PPL and ΔPcs, indicating a notable decline in overall text quality.

To further understand the behavior of SA-ANS, Figure 8 illustrates the impact of different configurations of L and ν on SPPL. When L is fixed, increasing ν lowers the threshold $1 - \nu$ (as defined in Equation (14)), thereby reducing the size of the candidate pool during sampling. Experimental results suggest that larger values of ν are associated with lower SPPL—a trend also observed in the SAAC method. This may be attributed to the fact that a higher ν biases the model toward selecting high-probability tokens while filtering out low-probability, less fluent alternatives, which may help enhance text fluency despite a narrower sampling space.

This effect becomes particularly pronounced when L is small (e.g., $L = 4$), where more decoding steps are required to embed the same amount of information. In such cases, the model inherently tends to favor high-probability tokens at each step to ensure encoding efficiency. As a result, the fluency gains introduced by moderate values of ν are further amplified, allowing SA-ANS to achieve lower perplexity.

To further investigate this interaction, Figure 9 visualizes the relationship between the selected candidate index k and token probability across different settings. It shows that, across all embedding precisions, larger ν values consistently lead to the selection of tokens with higher probabilities and lower k. This trend is especially evident when L is small, where the increased number of encoding steps magnifies the effect. The observed log-linear pattern confirms that SA-ANS tends to prioritize high-probability tokens under stricter nucleus constraints, thereby contributing to more fluent text generation.

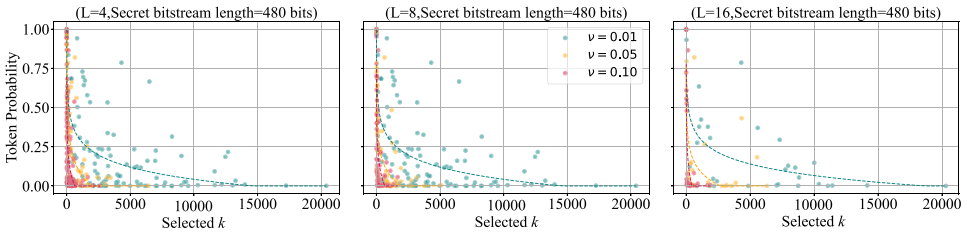

Figure 9
Scatter plots with log-linear fits showing the relationship between selected index k and token probability during encoding, under different nucleus thresholds $\nu \in 0.01, 0.05, 0.10$ and embedding precisions $L \in 4, 8, 16$. All settings use a fixed 480-bit secret message.

6.5 Statistical Analysis

Table 5 presents a detailed comparison of various steganographic methods across four datasets using two key metrics: D_{KL} and D_{JS}. Lower values of these metrics indicate that the generated text distribution is closer to that of the original LM, implying better statistical imperceptibility.

Overall, SA-ANS achieves the best performance in most configurations. In particular, when $\nu = 0.01$, it consistently yields the lowest divergence scores across nearly all datasets. For example, on the *Twitter* dataset, it attains $D_{\mathrm{KL}} = 0.012$ and $D_{\mathrm{JS}} = 0.00001$, outperforming all baselines. These results suggest that adopting adaptive sampling with a smaller ν allows SA-ANS to better preserve the statistical properties of the original LM during encoding.

As illustrated in Figure 10, decreasing ν substantially reduces D_{KL}, since a smaller ν enlarges the candidate pool and makes the generated distribution more closely match the original LM. However, this gain in imperceptibility comes at the cost of text quality: As ν decreases, PPL tends to increase. A larger candidate set increases the likelihood of selecting low-probability tokens, potentially reducing fluency. This reflects an inherent trade-off in steganographic text generation—lower divergence improves statistical imperceptibility, while lower PPL enhances linguistic quality—necessitating a balance based on application requirements.

To better understand this trade-off and the source of SA-ANS's advantage, we compare the changes in top-k values selected by SA-ANS and SAAC at each step under identical settings. As shown in Figure 11, SA-ANS's adaptive mechanism frequently reduces the top-k value during encoding—but only when the current coding range is much larger than the required capacity to embed the remaining bits (i.e., $M \gg L$). In

Table 5
Statistical analysis comparison of stegos generated by SA-ANS, ANS, and baselines. "↓" is that the lower the value, the better the performance. **Bold** is the best result.

Datasets		Twitter			IMDB			News			COVID-19		
Methods	**Param**	ER ↑	D_{KL} ↓	D_{JS} ↓	ER ↑	D_{KL} ↓	D_{JS} ↓	ER ↑	D_{KL} ↓	D_{JS} ↓	ER ↑	D_{KL} ↓	D_{KL} ↓
RNN-Stega	H = 3	2.35	1.105	0.01520	2.30	1.184	0.01126	2.47	1.131	0.01024	2.46	1.101	0.01439
	H = 5	2.79	0.589	0.00124	3.34	0.771	0.00216	3.82	0.658	0.00721	3.14	0.542	0.00348
	H = 7	5.01	0.688	0.00125	5.51	0.507	0.00185	4.48	0.455	0.04200	5.11	0.601	0.00342
ADG	–	2.31	2.021	0.03195	1.65	1.709	0.00132	2.12	2.381	0.01033	1.82	1.718	0.01032
AC	$k = 300$	4.37	0.197	0.02137	4.02	0.229	0.00217	4.66	0.216	0.00748	2.31	0.146	0.00591
	$k = 600$	4.11	0.133	0.02001	4.28	0.143	0.00631	2.82	0.156	0.00492	4.26	0.149	0.00124
	$k = 900$	3.62	0.095	0.01127	3.00	0.067	0.00213	4.23	0.097	0.00731	5.51	0.108	0.00203
SAAC	$\nu = 0.01$	5.07	0.096	0.01244	5.23	0.156	0.00722	5.39	0.072	0.00401	5.96	0.111	0.01082
	$\nu = 0.05$	4.88	0.107	0.00416	4.50	0.177	0.00849	5.27	0.089	0.00527	5.17	0.147	0.00684
	$\nu = 0.10$	4.49	0.144	0.00317	4.42	0.266	0.00993	5.15	0.095	0.00926	4.57	0.151	0.00410
SA-ANS	$\nu = 0.00$ (ANS)	4	0.051	0.00016	4	0.106	0.00191	4	0.047	0.01130	4	0.073	0.00721
		8	0.117	0.00027	8	0.167	0.00388	8	0.038	0.00130	8	0.093	0.00210
		16	0.185	0.00378	16	0.152	0.00481	16	0.184	0.01031	16	0.102	0.01001
	$\nu = 0.01$	4	**0.012**	**0.00001**	4	0.014	**0.00015**	4	0.013	**0.00002**	4	**0.014**	**0.00020**
		8	0.014	0.00002	8	0.014	0.00018	8	**0.012**	**0.00002**	8	0.015	0.00170
		16	0.014	0.00002	16	**0.013**	0.00726	16	0.014	**0.00002**	16	0.015	0.00181
	$\nu = 0.05$	4	0.049	0.00036	4	0.068	0.00043	4	0.066	0.00041	4	0.044	0.00041
		8	0.073	0.00044	8	0.071	0.00043	8	0.064	0.00036	8	0.067	0.00039
		16	0.071	0.00043	16	0.070	0.00043	16	0.067	0.00041	16	0.067	0.00039
	$\nu = 0.10$	4	0.115	0.00130	4	0.130	0.00016	4	0.134	0.00167	4	0.149	0.00110
		8	0.139	0.00182	8	0.102	0.00018	8	0.126	0.00156	8	0.129	0.00170
		16	0.133	0.00179	16	0.144	0.00016	16	0.130	0.00198	16	0.145	0.00181

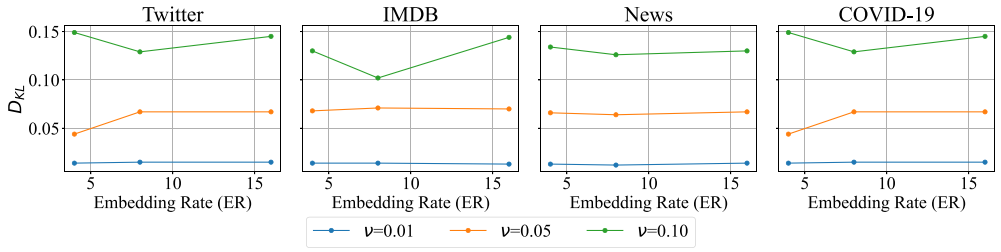

Figure 10
D_{KL} under different ER and ν values across four datasets. We observe that D_{KL} generally increases with larger ν and tends to remain stable or slightly increase with higher ER. A smaller ν leads to lower D_{KL}, indicating better imperceptibility.

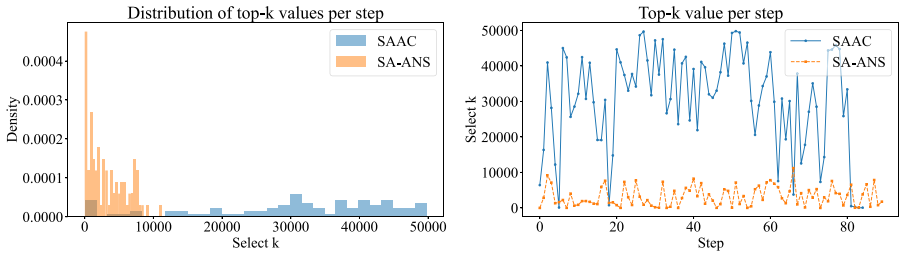

Figure 11
Top-k selection of SA-ANS vs. SAAC. SA-ANS often reduces k when capacity is sufficient, keeping candidates in the high-probability region, which lowers D_{KL} and PPL. SAAC relies solely on interval count, frequently expanding k to meet capacity, adding low-probability tokens and increasing divergence.

such cases, reducing k keeps the candidate set concentrated in the highest-probability region of the LM distribution, preserving relative probabilities and making the generated distribution closer to the original LM, thereby lowering D_{KL}. A narrower candidate set also increases the likelihood of selecting high-probability (low-rank) tokens, which improves fluency and reduces PPL. This synergy between selective k reduction and ranking preference allows SA-ANS to balance statistical imperceptibility with text quality. In contrast, SAAC derives its encoding capacity solely from the number of probability intervals defined by the candidate token distribution at each step. When the distribution is highly concentrated, high-probability tokens cover only a small number of intervals, insufficient to encode all the required bits. To meet capacity demands, SAAC must expand the candidate set to include low-probability, high-rank tokens. These tokens rarely appear in the LM's natural distribution, so their frequent inclusion shifts probability mass to the tail, distorting the distribution and increasing D_{KL}.

Taken together, both the quantitative results in Table 5 and the qualitative analysis in Figure 11 confirm that SA-ANS's lower and more stable token ranks are a key factor in its superior statistical imperceptibility.

6.6 Anti-steganalysis

Figure 12 reports the detection accuracies of stego texts generated by four baseline schemes and our proposed SA-ANS method under three steganalysis models: FastText (Joulin et al. 2016), TextRNN (Liu, Qiu, and Huang 2016), and DPCNN (Johnson and

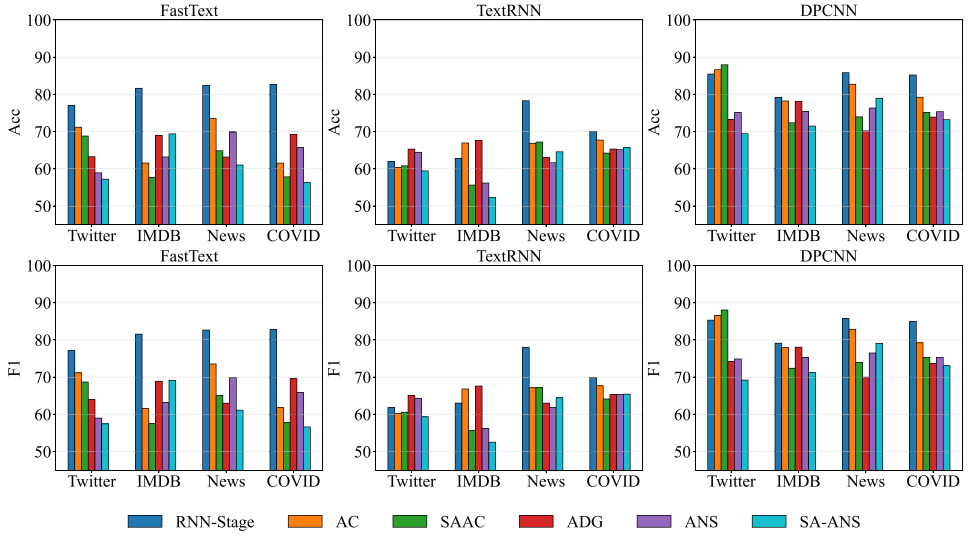

Figure 12
Anti-steganalysis comparison of stegos generated by SA-ANS, ANS, and baselines.

Zhang 2017). Across most settings, SA-ANS achieves lower detection accuracies than the baselines, indicating improved robustness against diverse steganalysis techniques. These results demonstrate that SA-ANS can generate stego texts that preserve a high level of security while offering better resistance to detection.

6.7 Human Evaluation

To ensure a fair comparison, we adopt a standardized human evaluation protocol. Specifically, 100 news articles are randomly sampled from the CNN/DailyMail dataset, and the first three sentences of each article are used as context. For each method under evaluation, one stego sentence is generated by embedding ciphertext into this context, while the original human-written fourth sentence is retained as a reference.

We construct a mixed evaluation set containing both machine-generated stego sentences and genuine human-written continuations. Each Human Intelligence Task (HIT) on Amazon Mechanical Turk presents crowd workers with the three-sentence context followed by a single candidate fourth sentence. Workers are instructed to judge whether the continuation is natural, considering grammaticality, contextual coherence, and factual plausibility.

Table 6 illustrates an example questionnaire containing the shared context and six candidate continuations from different methods. Participants mark the sentences they consider natural and may optionally provide justifications. Table 7 shows one participant's annotated responses, where SA-ANS outputs are frequently accepted as coherent and contextually appropriate, while certain baselines (such as HC) are often rejected due to vagueness, incoherence, or grammatical errors.

For quantitative evaluation, we define the *human judgment score* as the proportion of stego sentences rated natural by evaluators. The task itself is non-trivial—participants

Table 6
Human evaluation task: Continuation judgment questionnaire.

Instructions
You will read the first three sentences of a news article. Your task is to decide whether each candidate fourth sentence sounds like a natural continuation. Please consider: 1) Is the grammar natural? 2) Is the context coherent? 3) Are there any factual or commonsense issues?
Context (First 3 Sentences)
The US economy saw record growth in Q3. Analysts believe this was due to strong consumer spending. However, some sectors remained below pre-pandemic levels.
Candidate 4th Sentences
Please read each sentence below and tick the box if you think it is a good continuation. ☐ *1. Manufacturing jobs have not yet returned to previous levels.* ☐ *2. Companies have seen the strongest growth in their margins and dividend payouts, and sales and investment activity is more active.* ☐ *3. The most important sectors for growth were manufacturing.* ☐ *4. Manufacturing saw the biggest sharp growth in over the past several months as a sign that the housing market is feeling the price squeeze.* ☐ *5. The US stock market fell for the first time in more than two years after falling nearly 7% over the first three months.* ☐ *6. In general construction is down virtually, which fuelled growth.*
Your Judgment
For each sentence above, please mark: 1) **Yes**, if you think it is natural and plausible. 2) **No**, if it does not sound realistic (because of grammar, context mismatch, or factual error). You may also leave a brief comment explaining your judgment (optional).

Table 7
Participant judgments and concise justifications.

ID	Candidate Sentence		Justification	Algorithm
1	*Manufacturing jobs have not yet returned to previous levels.*	[✓]	*Coherent with the previous sentence and grammatically correct.*	original
2	*Companies have seen the strongest growth...*	[✓]	*Related to economic recovery and well-formed.*	SA-ANS
3	*The most important sectors for growth were manufacturing.*	[×]	*Too vague and incomplete as a continuation.*	HC
4	*Manufacturing saw the biggest sharp growth...*	[×]	*Unclear and illogical link to housing market.*	SAAC
5	*The US stock market fell...*	[×]	*Topic shift breaks context coherence.*	AC
6	*In general construction is down virtually...*	[×]	*Ungrammatical and semantically unclear.*	ADG

correctly identify the genuine continuation only 74% of the time. As shown in Figure 13, SA-ANS consistently surpasses traditional baselines and approaches the naturalness level of human-written text. Rejections of SA-ANS outputs are primarily due to factual inaccuracies or subtle contextual mismatches, suggesting that enhancing factual grounding remains an important direction for future research in LS.

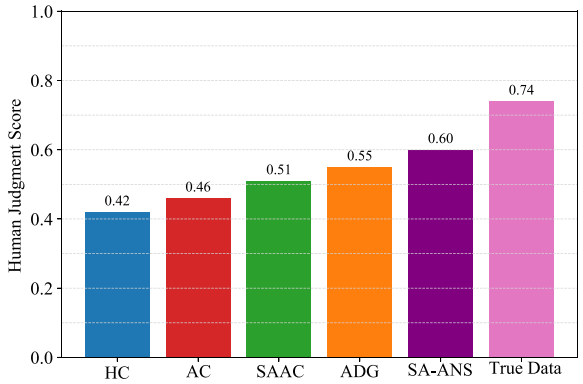

Figure 13
Comparison of human evaluation results for different linguistic steganography methods.

6.8 Ablation Study

To better understand the impact of key design choices in SA-ANS, we conduct a series of ablation experiments focusing on five aspects: *adaptive vs. non-adaptive variants*, *the adaptive control mechanism* (via the nucleus threshold ν), *embedding precision* L, *coding precision* q, and *model robustness across architectures*. Unless otherwise stated, all experiments use the GPT-2 LM, except for the final robustness analysis.

Adaptive vs. Non-adaptive (SA-ANS vs. ANS). We first compare SA-ANS with its non-adaptive counterpart ($\nu = 0.00$). As shown in Table 4, SA-ANS achieves lower SPPL and ΔPcs across most datasets and embedding configurations, with the gap more pronounced at higher embedding precisions (e.g., $L = 16$). This suggests that adaptive control improves fluency and reduces semantic distortion.

Effect of the Threshold ν. We vary $\nu \in 0.01, 0.05, 0.10$ under fixed coding precision to examine adaptive filtering strength. Larger ν values enforce stricter candidate filtering, yielding better text quality (lower SPPL) as shown in Figure 8. Conversely, smaller ν values enlarge the candidate pool, increasing embedding capacity but slightly reducing fluency. Figure 10 shows that this also improves imperceptibility by lowering D_{KL}, confirming ν as a key knob for balancing capacity and quality.

Effect of Embedding Precision L. Keeping other parameters fixed, we vary $L \in 4, 8, 16$. As shown in Figure 8, larger L generally boosts ER but can degrade text quality (higher SPPL). Figure 10 indicates that moderate values (e.g., $L = 8$) often achieve a better trade-off, providing relatively low D_{KL} while maintaining reasonable capacity.

Effect of Coding Precision q. We analyze q in Section 6.1. As Figure 6 shows, SPPL drops sharply as q increases. Low q forces frequent selection of low-probability tokens or fallback operations, leading to incoherence and high perplexity. Higher q offers finer-grained distributions and stable decoding, but at higher memory cost. We set $q = 30$ in all experiments to balance stability and efficiency.

Table 8
Statistical comparison of stego texts generated by SA-ANS and ANS with LLaMA-3.1-8B and Qwen-3-8B across four datasets. Metrics include ER, SPPL, and D_{KL}.

Datasets		Twitter			IMDB			News			COVID-19		
Model	**Param**	ER ↑	SPPL ↓	D_{KL} ↓	ER ↑	SPPL ↓	D_{KL} ↓	ER ↑	SPPL ↓	D_{KL} ↓	ER ↑	SPPL ↓	D_{KL} ↓
Llama-3.1-8B	$\nu = 0.00$ (ANS)	4	10.50	0.049	4	8.57	0.041	4	18.37	0.014	4	48.44	0.036
		8	8.88	0.036	8	23.39	0.031	8	24.37	0.012	8	45.63	0.033
		16	42.98	0.029	16	8.72	0.049	16	41.58	0.021	16	68.79	0.012
	$\nu = 0.01$	4	7.81	0.014	4	15.61	0.016	4	4.53	0.024	4	6.23	0.015
		8	6.17	0.013	8	5.27	0.014	8	3.70	0.019	8	4.45	0.001
		16	8.82	0.014	16	40.01	0.014	16	3.08	0.021	16	12.36	0.014
	$\nu = 0.05$	4	3.16	0.122	4	2.32	0.097	4	2.64	0.084	4	3.77	0.101
		8	7.14	0.082	8	2.48	0.104	8	3.72	0.078	8	5.30	0.085
		16	5.39	0.083	16	1.67	0.081	16	3.61	0.083	16	5.54	0.093
	$\nu = 0.10$	4	3.17	0.172	4	2.21	0.175	4	2.61	0.192	4	2.42	0.133
		8	3.35	0.162	8	1.86	0.146	8	2.51	0.206	8	2.85	0.138
		16	4.48	0.159	16	1.75	0.111	16	2.55	0.192	16	3.21	0.139
Qwen3-8B	$\nu = 0.00$ (ANS)	4	25.65	0.031	4	51.35	0.042	4	21.26	0.038	4	33.61	0.024
		8	43.60	0.021	8	11.95	0.039	8	15.28	0.018	8	32.02	0.031
		16	5.19	0.014	16	15.25	0.018	16	21.12	0.035	16	23.63	0.015
	$\nu = 0.01$	4	4.28	0.049	4	2.87	0.037	4	2.49	0.033	4	4.97	0.016
		8	3.75	0.014	8	5.19	0.031	8	6.81	0.018	8	3.40	0.016
		16	6.48	0.014	16	7.29	0.014	16	4.77	0.013	16	8.77	0.014
	$\nu = 0.05$	4	1.13	0.024	4	1.92	0.123	4	1.89	0.066	4	1.90	0.087
		8	1.28	0.048	8	2.06	0.126	8	2.50	0.064	8	1.58	0.071
		16	1.62	0.087	16	2.20	0.144	16	2.47	0.074	16	1.86	0.094
	$\nu = 0.10$	4	1.05	0.022	4	1.63	0.144	4	1.17	0.040	4	1.28	0.036
		8	1.11	0.043	8	2.06	0.169	8	1.35	0.069	8	1.24	0.030
		16	1.22	0.085	16	2.90	0.183	16	1.69	0.084	16	1.54	0.058

Model Robustness Across Architectures. We test SA-ANS on three representative LMs—GPT-2, LLaMA-3.1-8B (Huang et al. 2024), and Qwen-3-8B (Qwen Team 2024)—using SPPL for fluency and D_{KL} for distributional alignment (Table 8).

As shown in Figure 14, LLaMA-3.1-8B and Qwen-3-8B consistently outperform GPT-2 in SPPL, especially at higher ν, indicating that larger, more advanced models better maintain fluency under steganographic constraints.

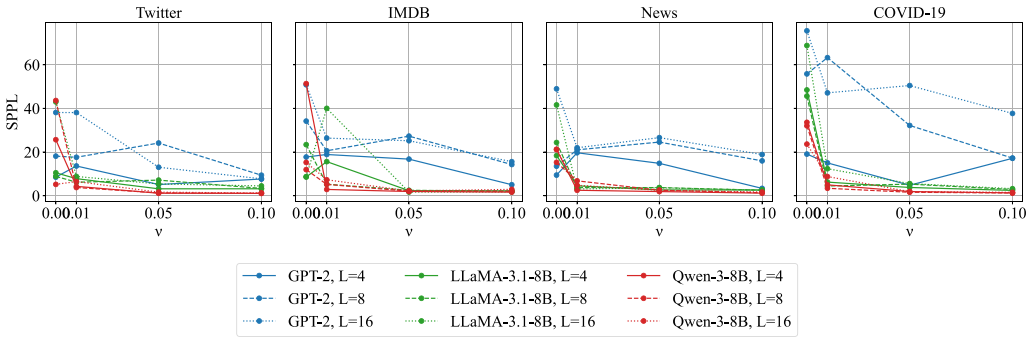

Figure 14
SPPL performance of SA-ANS across four datasets with varying imperceptibility thresholds ν and embedding precisions L for GPT-2, LLaMA-3.1-8B, and Qwen-3-8B.

However, as shown in Table 8, the differences in D_{KL} across architectures are relatively small and vary with the dataset, L, and ν. This indicates that while model architecture has a clear impact on fluency, its effect on distributional similarity is less consistent, and no single model consistently outperforms the others in imperceptibility.

7. Ethical Considerations

Steganography, by design, facilitates covert communication. While our proposed SA-ANS-based LS system is intended to advance research in natural language generation and information security, it inevitably raises important ethical concerns.

From a technical perspective, the proposed method enhances both embedding capacity and imperceptibility. These improvements can serve legitimate and socially beneficial applications, such as protecting whistleblower identities, securing communications for journalists, or bypassing censorship in authoritarian environments. Such use cases demonstrate the value of steganography as a tool for preserving privacy and promoting information freedom.

However, from an ethical and societal standpoint, the same technology may be misused for malicious purposes, including covert coordination of illicit activities, unauthorized data exfiltration, or disseminating disinformation while evading detection. This dual-use nature necessitates a careful examination of potential deployment contexts.

To reduce the risk of misuse, we recommend that future implementations of linguistic steganography systems incorporate:

- Usage transparency (e.g., via watermarking or traceable decoding mechanisms).
- Access control for model deployment and inference.

Additionally, we advocate for collaboration with ethicists and policymakers to develop clear guidelines for the responsible use of such technologies.

We explicitly prohibit any illegal or unethical use of our codebase and models. We strongly support responsible research practices and encourage the broader NLP community to engage in open dialogue regarding the governance and ethical oversight of powerful generative technologies.

8. Conclusion

In this article, we present SA-ANS, a linguistic steganography framework that integrates probabilistic encoding with self-adaptive language generation. Leveraging the structural advantages of Asymmetric Numeral Systems (ANS), SA-ANS dynamically adjusts the candidate token set based on the language model's output, enabling flexible control of the embedding rate and achieving an effective balance between linguistic quality and statistical imperceptibility. This adaptability makes SA-ANS suitable for diverse real-world scenarios with varying requirements for payload and security.

Theoretical analysis demonstrates that SA-ANS offers superior decoding stability over arithmetic coding and lower computational complexity than Huffman coding and the ADG scheme. Extensive experiments across multiple datasets show that SA-ANS consistently outperforms baselines in perplexity, KL divergence, and human evaluation,

while exhibiting strong resistance to steganalysis and maintaining high fluency and semantic coherence.

Human studies further confirm that SA-ANS stego texts are frequently indistinguishable from genuine human-written content. Future work will explore adversarial robustness, lightweight decoding, privacy-enhancing techniques such as differential privacy, and applications to controllable watermarking for large language models.

Appendix A. The Information Embedding Algorithm

The following pseudocode describes the information embedding process of the SA-ANS algorithm:

Algorithm 1 The Information Embedding Algorithm.

Input: Secret bitstream $X = [0, 1, 1, 0, \dots]$, Context C, Embedding precision L, Coding precision q, Temperature T, Nucleus ν

Output: Stego token sequence S, Final status s_n

$M \leftarrow 2^q$, $\chi \leftarrow \mathsf{bitsToInt}(X[0:L])$, $pos \leftarrow L$, $S \leftarrow [\,]$, $t \leftarrow 0$
while $pos < |X|$ **or** true **do**
  $\mathbf{P}_{\mathrm{LM}} \leftarrow \mathsf{softmax}(\mathrm{LM}(C)/T)$
  Select top-k from $\mathbf{P}_{\mathrm{LM}}$ with cumulative mass $\geq 1 - \nu$
  Convert top-k to integer frequencies f_i, compute CDF
  $u \leftarrow \chi \bmod M$, find j such that $\mathbf{CDF}[j] \leq u < \mathbf{CDF}[j+1]$
  $y \leftarrow$ token with index j, append to S, update C
  $\chi \leftarrow \lfloor \chi / M \rfloor \cdot f_j + (u - \mathbf{CDF}[j])$
  for $i = 1$ **to** L **do**
    if $pos < |X|$ **then**
      $\chi \leftarrow (\chi \ll 1) | X[pos]$
      $pos \leftarrow pos + 1$
    else
      $\chi \leftarrow (\chi \ll 1)$ *// pad with 0*
  end for
  $\gamma_n \leftarrow \chi$
end while

Appendix B. The Information Extraction Algorithm

The following pseudocode describes the information extraction process of the SA-ANS algorithm:

Algorithm 2 The Information Extraction Algorithm.

Input: Stego token sequence S, Final status γ_n, Context C, Embedding precision L, Coding precision q, Temperature T, Nucleus ν

Output: Recovered bitstream X

$M \leftarrow 2^q$, $X \leftarrow [\,]$
for $t = 0$ **to** $|S| - 1$ **do**
  $\mathbf{P}_{\text{LM}} \leftarrow \text{softmax}(\text{LM}(C)/T)$
  Select top-k tokens with cumulative prob $\geq 1 - \nu$
  Convert top-k to integer frequencies $\{f_j\}$ and compute CDF
  Locate index j of token s_t in top-k
  $f \leftarrow f_j$, $base \leftarrow \mathbf{CDF}[j]$
  Compute ANS state: $\chi \leftarrow \lfloor \gamma_t / f \rfloor \cdot M + base + (\gamma_t \bmod f)$
  for $i = 1$ **to** L **do**
    Append $(\chi \bmod 2)$ to X
    $\chi \leftarrow \lfloor \chi / 2 \rfloor$
  end for
  $\gamma_{t+1} \leftarrow \chi$
  Update context $C \leftarrow C \| s_t$
end for
Return Reverse(X)

References

AlSabhany, Ahmed A., Ahmed Hussain Ali, Farida Ridzuan, A. H. Azni, and Mohd Rosmadi Mokhtar. 2020. Digital audio steganography: Systematic review, classification, and analysis of the current state of the art. *Computer Science Review*, 38:100316. https://doi.org/10.1016/j.cosrev.2020.100316

Bolshakov, Igor A. and Alexander Gelbukh. 2004. Synonymous paraphrasing using WordNet and internet. In *International Conference on Application of Natural Language to Information Systems*, pages 312–323. https://doi.org/10.1007/978-3-540-27779-8_27

Chapman, Mark and George Davida. 1997. Hiding the hidden: A software system for concealing ciphertext as innocuous text. In *International Conference on Information and Communications Security*, pages 335–345. https://doi.org/10.1007/BFb0028489

Chen, Zhili, Liusheng Huang, Peng Meng, Wei Yang, and Haibo Miao. 2011. Blind linguistic steganalysis against translation based steganography. In *Digital Watermarking: 9th International Workshop, IWDW 2010*, pages 251–265. https://doi.org/10.1007/978-3-642-18405-5_21

Dai, Falcon Z. and Zheng Cai. 2019. Towards near-imperceptible steganographic text. *arXiv preprint arXiv:1907.06679*. https://doi.org/10.18653/v1/P19-1422

Dai, Weihui, Yue Yu, Yonghui Dai, and Bin Deng. 2010. Text steganography system using Markov chain source model and DES algorithm. *Journal of Software*, 5(7):785–792. https://doi.org/10.4304/jsw.5.7.785-792

Dubey, Abhimanyu, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 herd of models. *arXiv e-prints arXiv:2407.21783*.

Duda, Jarek. 2013. Asymmetric numeral systems: Entropy coding combining speed of Huffman coding with compression rate of arithmetic coding. *arXiv preprint arXiv:1311.2540*.

Fang, Tina, Martin Jaggi, and Katerina Argyraki. 2017. Generating steganographic text with LSTMS. *arXiv preprint arXiv:1705.10742*. https://doi.org/10.18653/v1/P17-3017

Fridrich, Jessica. 2009. *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press. https://doi.org/10.1017/CBO9781139192903

Go, Alec, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009.

Hermann, Karl Moritz, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in Neural Information Processing Systems*, 28.

Huang, Siyuan, Xavier Garcia, Thomas Scialom, et al. 2024. LlaMa 3: Open foundation and instruction-tuned language models. Meta AI. https://llama.meta.com/llama3

Huang, Yong Feng, Shanyu Tang, and Jian Yuan. 2011. Steganography in inactive frames of VoIP streams encoded by source Codec. *IEEE Transactions on Information Forensics and Security*, 6(2):296–306. https://doi.org/10.1109/TIFS.2011.2108649

Huffman, David A. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101. https://doi.org/10.1109/JRPROC.1952.273898

Johnson, Rie and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 562–570. https://doi.org/10.18653/v1/P17-1052

Joulin, Armand, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*. https://doi.org/10.18653/v1/E17-2068

Kadhim, Inas Jawad, Prashan Premaratne, Peter James Vial, and Brendan Halloran. 2019. Comprehensive survey of image steganography: Techniques, evaluations, and trends in future research. *Neurocomputing*, 335:299–326. https://doi.org/10.1016/j.neucom.2018.06.075

Kang, Huixian, Hanzhou Wu, and Xinpeng Zhang. 2020. Generative text steganography based on LSTM network and attention mechanism with keywords. *Electronic Imaging*, 32:1–8. https://doi.org/10.2352/ISSN.2470-1173.2020.4.MWSF-291

Karpov, Artem, Tinuade Adeleke, Seong Hah Cho, and Natalia Perez-Campanero. 2025. The steganographic potentials of language models. *arXiv preprint arXiv:2505.03439*.

Kuznetsov, Oleksandr, Kyrylo Chernov, Aigul Shaikhanova, Kainizhamal Iklassova, and Dinara Kozhakhmetova. 2025. DeepStego: Privacy-preserving natural language steganography using large language models and advanced neural architectures. *Computers*, 14(5):165. https://doi.org/10.3390/computers14050165

Li, Jingzhi, Fengling Li, Lei Zhu, Hui Cui, and Jingjing Li. 2023. Prototype-guided knowledge transfer for federated unsupervised cross-modal hashing. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 1013–1022. https://doi.org/10.1145/3581783.3613837

Liao, Xin, Jiaojiao Yin, Mingliang Chen, and Zheng Qin. 2020. Adaptive payload distribution in multiple images steganography based on image texture features. *IEEE Transactions on Dependable and Secure Computing*, 19(2):897–911.

Liu, Pengfei, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Low, Steven H., Nicholas F. Maxemchuk, Jack T. Brassil, and Lawrence O'Gorman. 1995. Document marking and identification using both line and word shifting. In *Proceedings of INFOCOM'95*, volume 2, pages 853–860. https://doi.org/10.1109/INFCOM.1995.515956

Luo, Yubo, Yongfeng Huang, Fufang Li, and Chinchen Chang. 2016. Text steganography based on Ci-poetry generation using Markov chain model. *KSII Transactions on Internet and Information Systems (TIIS)*, 10(9):4568–4584. https://doi.org/10.3837/tiis.2016.09.029

Luo, Yuanjing, Jiaohua Qin, Xuyu Xiang, and Yun Tan. 2020. Coverless image steganography based on multi-object recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(7):2779–2791. https://doi.org/10.1109/TCSVT.2020.3033945

Maas, Andrew, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150.

Mikolov, Tomas, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. https://doi.org/10.21437/Interspeech.2010-343

Moraldo, H. Hernan. 2014. An approach for text steganography based on Markov chains. *arXiv preprint arXiv:1409.0915*.

Niu, Yan, Juan Wen, Ping Zhong, and Yiming Xue. 2019. A hybrid R-BILSTM-C neural network based text steganalysis. *IEEE Signal Processing Letters*, 26(12):1907–1911. https://doi.org/10.1109/LSP.2019.2953953

Perry, Neil, Sanket Gupte, Nishant Pitta, and Lior Rotem. 2025. Robust steganography from large language models. *arXiv preprint arXiv:2504.08977*.

Petitcolas, Fabien A. P. and Stefan Katzenbeisser. 2000. *Information Hiding Techniques for Steganography and Digital Watermarking (Artech House Computer Security Series)*. Artech House.

Qwen Team. 2024. Qwen3: A new generation of large language models from Alibaba Cloud. https://github.com/QwenLM/Qwen. Alibaba Cloud.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Reid, Mark D. and Robert C. Williamson. 2009. Generalised Pinsker inequalities. *arXiv preprint arXiv:0906.1244*.

Rissanen, Jorma and Glen G. Langdon. 1979. Arithmetic coding. *IBM Journal of Research and Development*, 23(2):149–162. https://doi.org/10.1147/rd.232.0149

Shen, Jiaming, Heng Ji, and Jiawei Han. 2020. Near-imperceptible neural linguistic steganography via self-adjusting arithmetic coding. *arXiv preprint arXiv:2010.00677*. https://doi.org/10.18653/v1/2020.emnlp-main.22

Shniperov, Alexey Nikolaevich and K. A. Nikitina. 2016. A text steganography method based on Markov chains. *Automatic Control and Computer Sciences*, 50:802–808. https://doi.org/10.3103/S0146411616080174

Simmons, Gustavus J. 1984. The prisoners' problem and the subliminal channel. In

Advances in Cryptology: Proceedings of Crypto 83, pages 51–67. https://doi.org/10.1007/978-1-4684-4730-9_5

Tang, Xing and Mingsong Chen. 2013. Design and implementation of information hiding system based on RGB. In *2013 3rd International Conference on Consumer Electronics, Communications and Networks*, pages 217–220. https://doi.org/10.1109/CECNet.2013.6703310

Wang, Huili, Zhongliang Yang, Jinshuai Yang, Yue Gao, and Yongfeng Huang. 2023. Hi-Stega: A hierarchical linguistic steganography framework combining retrieval and generation. In *International Conference on Neural Information Processing*, pages 41–54. https://doi.org/10.1007/978-981-99-8073-4_4

Wang, Lucy Lu, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Douglas Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Kinney, et al. 2020. CORD-19: The COVID-19 open research dataset. *ArXiv*.

Wayner, Peter. 1992. Mimic functions. *Cryptologia*, 16(3):193–214. https://doi.org/10.1080/0161-119291866883

Wen, Juan, Xuejing Zhou, Ping Zhong, and Yiming Xue. 2019. Convolutional neural network based text steganalysis. *IEEE Signal Processing Letters*, 26(3):460–464. https://doi.org/10.1109/LSP.2019.2895286

Witten, Ian H., Radford M. Neal, and John G. Cleary. 1987. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540. https://doi.org/10.1145/214762.214771

Wu, Jiaxuan, 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*, pages 10345–10353. https://doi.org/10.1145/3664647.3680562

Xue, Yiming, Jie Zhou, Hao Zeng, Ping Zhong, and Juan Wen. 2019. An adaptive steganographic scheme for H. 264/AVC video with distortion optimization. *Signal Processing: Image Communication*, 76:22–30. https://doi.org/10.1016/j.image.2019.04.012

Yang, Zhong Liang, Xiao-Qing Guo, Zi-Ming Chen, Yong-Feng Huang, and Yu-Jin Zhang. 2018a. RNN-Stega: Linguistic steganography based on recurrent neural networks. *IEEE Transactions on Information Forensics and Security*, 14(5):1280–1295. https://doi.org/10.1109/TIFS.2018.2871746

Yang, Zhongliang, Shuyu Jin, Yongfeng Huang, Yujin Zhang, and Hui Li. 2018b. Automatically generate steganographic text based on Markov model and Huffman coding. *arXiv preprint arXiv:1811.04720*.

Yang, Zhongliang, Xueshun Peng, Yongfeng Huang, and Chinchen Chang. 2018c. A novel method of speech information hiding based on 3D-magic matrix. *arXiv preprint arXiv:1809.03010*.

Yang, Zhongliang, Ke Wang, Jian Li, Yongfeng Huang, and Yu-Jin Zhang. 2019. TS-RNN: Text steganalysis based on recurrent neural networks. *IEEE Signal Processing Letters*, 26(12):1743–1747. https://doi.org/10.18280/ts.360405

Yang, Zhong Liang, Si-Yu Zhang, Yu-Ting Hu, Zhi-Wen Hu, and Yong-Feng Huang. 2020. VAE-Stega: Linguistic steganography based on variational auto-encoder. *IEEE Transactions on Information Forensics and Security*, 16:880–895. https://doi.org/10.1109/TIFS.2020.3023279

Zhang, Dengyong, Xiao Chen, Feng Li, Arun Kumar Sangaiah, and Xiangling Ding. 2020. Seam-carved image tampering detection based on the cooccurrence of adjacent LBPS. *Security and Communication Networks*, 2020(1):8830310. https://doi.org/10.1155/2020/8830310

Zhang, Jianjun, Jun Shen, Lucai Wang, and Haijun Lin. 2016. Coverless text information hiding method based on the word rank map. In *Cloud Computing and Security: Second International Conference, ICCCS 2016*, pages 145–155. https://doi.org/10.1007/978-3-319-48671-0_14

Zhang, Jianjun, Yicheng Xie, Lucai Wang, and Haijun Lin. 2017. Coverless text information hiding method using the frequent words distance. In *Cloud Computing and Security: Third International Conference, ICCCS 2017*, pages 121–132. https://doi.org/10.1007/978-3-319-68505-2_11

Zhang, Siyu, Zhongliang Yang, Jinshuai Yang, and Yongfeng Huang. 2021. Provably secure generative linguistic steganography. *arXiv preprint arXiv:2106.02011*. https://doi.org/10.18653/v1/2021.findings-acl.268

Zhou, Xuejing, Wanli Peng, Boya Yang, Juan Wen, Yiming Xue, and Ping Zhong. 2021. Linguistic steganography based on adaptive probability distribution. *IEEE Transactions on Dependable and Secure Computing*, 19(5):2982–2997. https://doi.org/10.1109/TDSC.2021.3079957

Ziegler, Zachary M., Yuntian Deng, and Alexander M. Rush. 2019. Neural linguistic steganography. *arXiv preprint arXiv:1909.01496*.