Zero-shot Generative Linguistic Steganography

Ke Lin[1] **Yiyang Luo**[2] **Zijian Zhang**[1] **Ping Luo**[1]
[1]Tsinghua University
[2]Nanyang Technological University
{leonard.keilin, zijianzhang510}@gmail.com
lawrenceluoyy@outlook.com
luop@tsinghua.edu.cn

Abstract

Generative linguistic steganography attempts to hide secret messages into covertext. Previous studies have generally focused on the statistical differences between the covertext and stegotext, however, ill-formed stegotext can readily be identified by humans. In this paper, we propose a novel zero-shot approach based on in-context learning for linguistic steganography to achieve better perceptual and statistical imperceptibility. We also design several new metrics and reproducible language evaluations to measure the imperceptibility of the stegotext. Our experimental results indicate that our method produces 1.926× more innocent and intelligible stegotext than any other method[1].

1 Introduction

Data transmission is generally secured using encryption algorithms to create an unrecognizable ciphertext for secure data transmission (Bellare et al., 1997). Nonetheless, messages transmitted as ciphertexts may easily arouse suspicion from attackers, and eavesdropping may place a threat on the messages being sent. Once the communication channel is discovered, the attackers could alter the messages in any way, causing significant damage.

Steganography is the key to ensuring communication privacy by concealing sensitive messages within monitored channels. General steganography involves embedding a secret message into a public multimedia carrier, such as text (Krishnan et al., 2017), image (Subramanian et al., 2021; Tao et al., 2019; Guo et al., 2015), audio (Lin et al., 2018), and video (Liu et al., 2019) to create a stego-carrier that is then transmitted via the public channel. As shown in Fig. 1, the problem of secure and covert communication can be viewed as a message transmission between two parties, Alice and Bob, along with a malicious party, Eve. Eve will monitor the entire transmission and prevent unintelligible messages from being transmitted as Alice attempts to send secret information to Bob. The solution to this problem is to use a steganographic function to embed the secret message into a stego-carrier controlled by the shared secret key. The stego-carrier is then sent to Bob without alarming Eve, and Bob can extract the secret message from the stego-carrier by using the corresponding extraction function. A key objective in this scenario is to ensure that covert communications are imperceptible.

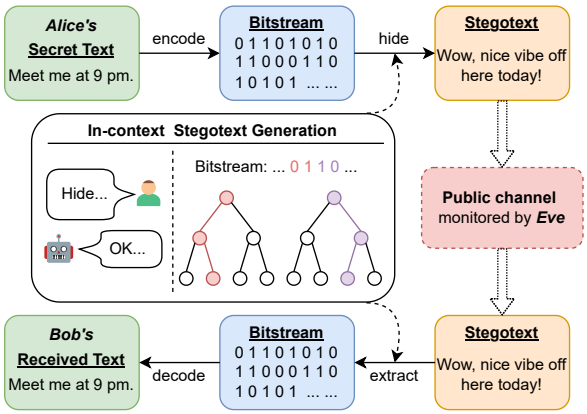

Figure 1: Generative linguistic steganography pipeline.

In public channels, text is commonly used, however, text contains relatively little redundant information (Zipf, 1999) when compared to other media, so embedding large amounts of information can be challenging. Moreover, steganalysis tools can detect stegotext containing secret messages based on machine learning or statistical characteristics (Taleby Ahvanooey et al., 2019). To address this problem, modern steganography utilizes machine learning techniques to enhance its imperceptibility.

Over the last few decades, researchers have devised many techniques to realize generation-based steganography (Xiang et al., 2022), including rule-based (Chapman and Davida, 1997), statistical (Moraldo, 2014; Shniperov and Nikitina, 2016),

[1]The project is available in https://github.com/leonardodalinky/zero-shot-GLS.

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 5168–5182
June 16-21, 2024

and combinations of these methods (Luo et al., 2016). Recent developments (Otter et al., 2021) in machine learning and natural language processing have led to improved performance in linguistic steganography. It involves the use of neural networks to learn the statistical distribution of a large number of sentences, and then generate steganographic sentences from the learned language model (Dai and Cai, 2019; Yang et al., 2019b).

Prior works have typically formulated statistical imperceptibility as the distance between the covertext distribution and stegotext distribution, however, there is no clear and simple method of measuring perceptual imperceptibility. Furthermore, many studies (Ziegler et al., 2019; Yang et al., 2021) have demonstrated that metrics for assessing statistical imperceptibility fail to measure perceptual imperceptibility and produce inaccurate or even incorrect results. The discrepancy between the statistical and perceptual imperceptibility prevents the practical application of linguistic steganography.

Our Contributions. In this work, we propose a novel zero-shot linguistic steganography method with both improved perceptual and statistical imperceptibility. Our new method is built based on the in-context learning of large language models (LLM), which utilize some samples of the covertext to generate more intelligible stegotext using the question-answer paradigm (QA). Initially, the secret text is encoded in binary bitstreams, which are then used to generate the stegotext. At each timestep t, the LM computes the distribution of the t-th token and embeds the bitstream into the generated text by Huffman encoding. To demonstrate the efficacy of our proposed method compared with different supervised and training-free approaches, we develop several new metrics and conduct reproducible experiments of language evaluation.

In summary, our contributions are as follows:

- We present a zero-shot approach for linguistic steganography based on in-context learning using samples of the covertext.
- We improve both the binary coding process and the embedding process by introducing several novel techniques.
- We design several metrics and language evaluations to evaluate both the perceptual and statistical imperceptibility, whereas our method produces more innocent and intelligible stegotext compared to all the previous methods.

2 Background

2.1 Generative Linguistic Steganography

General text generation task aims at creating token sequence $\boldsymbol{x} = [x_1, \ldots, x_n]$ from the joint probability of language model $\mathbf{P}_{\text{LM}}(\boldsymbol{x})$. The generation process can often be viewed as an auto-regression:

$$\begin{aligned}\mathbf{P}_{\text{LM}}(\boldsymbol{x}) &= \prod_{t=1}^{n} \mathbf{P}_{\text{LM}}(x_t \mid x_1, \ldots, x_{t-1}) \\ &= \prod_{t=1}^{n} \mathbf{P}_{\text{LM}}(x_t \mid x_{<t})\end{aligned} \tag{1}$$

The goal of linguistic steganography is to send a secret message $\boldsymbol{m} \in \{0,1\}^l \subset \mathcal{M}$ to the receiver through a monitored public channel. In generative linguistic steganography, the sender and receiver share an embedding algorithm $f_{\text{emb}} : \mathcal{P}_{\text{LM}} \times \mathcal{M} \mapsto \mathcal{P}_{\text{stega}}$, which transforms $\mathbf{P}_{\text{LM}}$ into the steganographic distribution $\mathbf{P}_{\text{stega}}$ controlled by message $\boldsymbol{m}$. It is then used to achieve covert communication by generating *stegotext* $\boldsymbol{y} \sim \mathbf{P}_{\text{stega}}$. Similarly, there exists an extraction process $f_{ext} : \mathcal{P}_{\text{stega}} \mapsto \mathcal{M}$ that extracts the message $\boldsymbol{m}$ from the stegotext $\boldsymbol{y}$.

2.2 Statistical Imperceptibility

To measure the statistical imperceptibility under surveillance by an eavesdropper, Cachin (1998) proposed to use the Kullback-Leibler divergence (KLD) between the distributions of natural text $\mathbf{P}_{\text{true}}$ and stegotext $\mathbf{P}_{\text{stega}}$. Since we approximate the distribution of true text $\mathbf{P}_{\text{true}}$ using a language model $\mathbf{P}_{\text{LM}}$, there is a gap between the distribution predicted by the model and the true distribution of text. Hence, we need to approximate this gap by finding an upper bound. We can formulate the upper bound of imperceptibility by using Jensen–Shannon divergence (JSD) instead:

$$\begin{aligned}\text{JSD}(\mathbf{P}_{\text{true}} \parallel \mathbf{P}_{\text{stega}})^{\frac{1}{2}} \leq \text{JSD}(\mathbf{P}_{\text{true}} \parallel \mathbf{P}_{\text{LM}})^{\frac{1}{2}} \\ + \text{JSD}(\mathbf{P}_{\text{LM}} \parallel \mathbf{P}_{\text{stega}})^{\frac{1}{2}}\end{aligned} \tag{2}$$

Assuming that the language model is successful in modeling the covertext distribution, we can simplify the statistical imperceptibility as $\text{JSD}(\mathbf{P}_{\text{LM}} \parallel \mathbf{P}_{\text{stega}})$. This strong assumption can be considered true due to language models' performance. In other words, it is feasible to measure imperceptibility using only a language model $\mathbf{P}_{\text{LM}}$ and a steganographic model $\mathbf{P}_{\text{stega}}$.

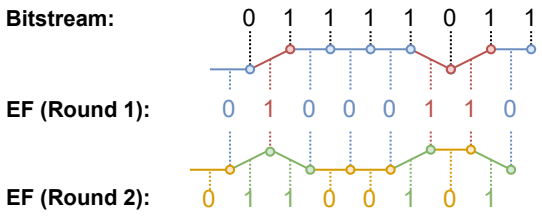

Figure 2: Example of EF coding. The red and green lines represent that the coding in bitstream changed (from 0 to 1, or from 1 to 0) and therefore will be assigned as 1 for the next iteration. The blue and yellow line represents that the coding in bitstream did not change (from 0 to 0, or from 1 to 1) and therefore will be assigned as 0 for the next iteration. Bitstream is processed through several rounds until the fewest 1s exist.

3 Methodology

In this section, we present the overall framework of our approach, consisting of three modules: (a) *Codec* module to (de)compress secret text into (from) bitstreams, (b) *Embedding* module to select candidate words from the distribution of stegotext, (c) *In-Context Stegotext Generation* module to approximate the joint distribution of covertext. The pipeline of our framework is illustrated in Fig. 1. Figure 3 also explains the details of the embedding module and the in-context stegotext generation.

3.1 Codec

The sender in linguistic steganography wishes to send secret text to the receiver, which must first be converted into a binary bitstream. We refer to this process as encoding and decoding (codec).

Variable-length Coding. There are two categories of encoding algorithms: fixed-length coding (FLC) and variable-length coding (VLC). We employ Huffman coding of the VLC family for token-level encoding, as it is a relatively simple algorithm and has a sub-optimal compression ratio. Tokens are represented in binary form using the Huffman code table conditioned on $\mathbf{P}_{\text{LM}}(x_t \mid x_{<t})$.

Edge Flipping Coding. We observe that more frequent occurrences of sequential zeros in the bitstreams can improve the quality of the generated text in the embedding module due to the local ordering of the Huffman tree (i.e., the left branch always has a higher probability than the right branch). Hence, we introduce Edge Flipping coding (EF), inspired by the differential encoding of communication (Weber, 1978), on top of the Huffman process to achieve better text generation performance.

In EF coding, the bitstream is transformed to record the position at which the bits change based on the edge-triggered flip-flops in electronics. We apply multi-round EF coding (e.g. 16 rounds) and the bitstream with the least number of 1s is retained. A simple example of EF is provided in Fig. 2. The analysis of EF coding is presented in Section 4.6.

3.2 Embedding

The goal of the embedding module is to hide and extract bitstreams within regular sentences.

Hide & Extract. During each phase of the generative operation, the LM yields a set of *candidate words*. These words are determined based on the subsequent token's probability and can potentially serve as alternatives for the next possible word to be generated. Then the LM will determine the length of the binary bitstream that can be embedded based on the number of candidate words.

Algorithm 1 shows the brief algorithm of the hiding process. The binary representation of selected words in the conditional word distribution is obtained using the Huffman encoding algorithm. It should be noted that this Huffman process is independent of the aforementioned Huffman coding in the Codec section. Following Dai and Cai (2019), we use the threshold τ to reduce perplexity by pruning words with lower probabilities.

Algorithm 1 Information Hiding Algorithm

Input: Bitstream $\boldsymbol{m}$, threshold τ.
Output: stegotext $\boldsymbol{y} = [y_1, \ldots, y_n]$.

Timestep $t \leftarrow 1$, output sentence $\boldsymbol{y} \leftarrow \emptyset$
while not the end of $\boldsymbol{m}$ **do**
  ▷ `Compute conditional probs`
  $\boldsymbol{p} \leftarrow \mathbf{P}_{\text{stega}}(x_t \mid x_{<t})$
  ▷ `Prune candidate words`
  $\boldsymbol{c} \leftarrow [c_i \mid \boldsymbol{p}(c_i) \geq \tau]$
  ▷ `Huffman encoding`
  $H \leftarrow \text{Huffman}(\boldsymbol{c}, \boldsymbol{p})$
  ▷ `Select candidate`
  $y_t \leftarrow$ Word $c \in H$ whose binary representation matches the prefix of $\boldsymbol{m}$
  $\boldsymbol{y} \leftarrow \boldsymbol{y} \cup \{y_t\}, t \leftarrow t + 1$
end while

As the reverse process of hiding, the information extraction process extracts steganographic information from stegotext. With a few minor differences,

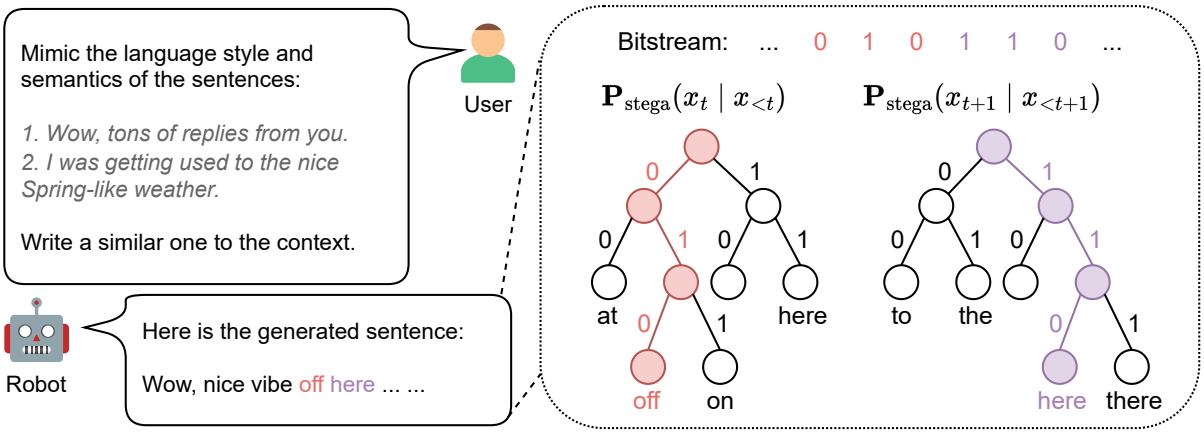

Figure 3: A running example of the embedding module and in-context stegotext generation. The selected context of samples from covertext in gray is used to instruct the stegotext generation. Huffman trees are constructed based on the conditional distribution, with the word matching the prefix of a bistream being selected as the next token. For example, the red token has 5 candidate words after applying probability pruning. Huffman tree is constructed for the 5 candidates and the word "off" matches the prefix of the bitstream and is chosen as the next word.

the extraction of secret bitstream is almost identical to the hiding process. The extraction algorithm is left out in Appendix A.2 for simplicity.

Annealing Selection. Candidate word selection with specific thresholds τ may result in trivial candidates in some cases (e.g., $|\boldsymbol{c}| = 1$). We note that trivial candidates often lead to a chain of trivial cases in subsequent stegotext generation, reducing the embedding rate and bit per word (BPW). Inspired by the stimulated annealing algorithm (van Laarhoven and Aarts, 1987), we propose the annealing search to prevent trivial candidates:

$$\mathbf{P}'_{\text{stega}}(x_t \mid x_{<t}) = \text{Softmax}(\frac{\text{LM}(x_t \mid x_{<t})}{T_t}) \quad (3)$$

Here, the temperature T_t at timestep t varies according to the previous candidates $\boldsymbol{c}_{t-1}$, the initial temperature T_0, and the annealing factor α:

$$T_t \leftarrow \begin{cases} \alpha \cdot T_{t-1} & |\boldsymbol{c}_{t-1}| = 1 \\ T_0 & |\boldsymbol{c}_{t-1}| > 1 \end{cases} \quad (4)$$

Repeat Penalty. Another corner case is that several generated words may repeat themselves after a few turns, but escape the previous annealing cases. The recently generated words are penalized for short-term repetition by applying a moving penalty:

$$\mathbf{P}''_{\text{stega}}(x_t \mid x_{<t}) = \text{Softmax}(\text{LM}(\cdot) - \boldsymbol{\delta}_t) \quad (5)$$

Penalties are based on the decay factor β, initial penalty δ_0, and the previously selected word. For any word $c \in \boldsymbol{c}_{t-1}$, $\boldsymbol{\delta}_t(c)$ has the following value:

$$\boldsymbol{\delta}_t(c) \leftarrow \begin{cases} \max\{0, \boldsymbol{\delta}_{t-1}(c) - \beta\} & c \text{ not selected} \\ \delta_0 & c \text{ selected} \end{cases} \quad (6)$$

3.3 In-Context Stegotext Generation

The core of the embedding module is to calculate the conditional probability $\mathbf{P}_{\text{stega}}(x_t \mid x_{<t})$. We use the question-answer paradigm with pre-defined prompting context C using the LLaMA2-Chat-7B (Touvron et al., 2023) as the stegotext generator. In other words, the distribution of stegotext can be represented as $\mathbf{P}_{\text{stega}}(x_t \mid x_{<t}, C)$, which is conditioned on both the previously generated tokens and the context from the covertext.

Context Selection. To approximate the distribution of covertext, we propose to incorporate several random samples from the covertext dataset into the prompting context. Since the covertext dataset is mostly available to both the sender and receiver, only a little information (e.g., random state for selection) is required to be pre-shared across different parties to reconstruct the prompting context. Therefore, an eavesdropper who suspects stegotext will be unable to decrypt the message.

In-Context Generation. As LLM is capable of comprehending the semantics of context due to its generalization ability, we utilize in-context QA tasks as a way to instruct the model to mimic the style and semantics of the covertext. To be more

specific, we design the following structured QA format: <$\texttt{ctx}_1$,...,$\texttt{ctx}_k$,[MISS]>, where the QA model is instructed to complete the missing sentence [MISS] which is similar to the context. The number of contextual sentences k is set as a hyperparameter during inference. The details of the in-context QA task can be found in Appendix A.3.

4 Experiments

4.1 Experiment Setup

Datasets. Our experiments evaluate the prior works and our method using two widely used large-scale datasets, the IMDB dataset (Maas et al., 2011) and the Twitter dataset (Go et al., 2009). The texts in each dataset are broken down into sentences by the nltk toolkit. Statistics are shown in Table 2.

Baselines. Since no training is needed for our zero-shot approach, we compare the performance of our method with several *training-free* methods: (a) **NLS** (Ziegler et al., 2019): This method embeds a secret bitstream using GPT-2 (Radford et al., 2019) for sentence complement. (b) **SAAC** (Shen et al., 2020): This method improves the embedding process using Self-Adjusting Arithmetic Coding.

Furthermore, we also compare our benchmarks against some *fully-supervised* steganographic techniques: (a) **RNN-Stega** (Yang et al., 2019a): This method utilizes Recurrent Neural Network models trained on covertext to generate stegotext. (b) **VAE-Stega** (Yang et al., 2021): This method adopts a Variational Auto-encoder to generate stegotext. (c) **ADG** (Zhang et al., 2021): This method proposes the Adaptive Dynamic Grouping techniques for better imperceptibility.

Implementation Details. All of the models and benchmarks are implemented with PyTorch. Pre-trained models (e.g., GPT-2 and LLaMA2-Chat) are obtained using Huggingface library. To ensure a fair comparison, we rebuild all baseline methods using the same datasets throughout the entire experiment. Bits per word (BPW) are used to measure the capability of hiding information within stegotext. Only steganographic methods with similar BPW should be compared. Additionally, we adopt a unified BPW by the nltk toolkit to eliminate vocabulary inconsistencies. In each setting, 8K stegotext is generated for evaluation. We adopt the LLaMA2-Chat-7B as the QA model and set the context size of our method to $k = 2$, the threshold $\tau = 0.005$, the factors $\alpha = 1.25, \beta = 0.5$, and the initial values $T_0 = 1.0, \delta_0 = 4.0$. More details of baselines can be found in Appendix A.4.

4.2 Evaluation Metrics

Perplexity. As a measure of the generation quality, the perplexity (PPL) is commonly used:

$$\text{PPL} = 2^{-\frac{1}{n}\sum_{i=1}^{n}\log_2 \mathbf{P}(x_i|x_{<i})} \tag{7}$$

JSD. Following Section 2.2, we measure the statistical imperceptibility of the covertext and stegotext using JSD. Specifically, we finetune two GPT-2 models on covertext $\mathcal{C}_{\text{cover}}$ and stegotext $\mathcal{C}_{\text{stega}}$, respectively, representing $\mathbf{P}_{\text{LM}}$ and $\mathbf{P}_{\text{stega}}$. An evaluation of the JSD is then conducted using randomly sampled text $\mathcal{C}_{\text{sample}}$. Three variants of JSD with respect to $\mathbf{P}_{\text{LM}}$ and $\mathcal{C}_{\text{sample}}$ are designed:

- *Full* JSD: Stego-models are trained on $\mathcal{C}_{\text{cover}}$ if needed[2]. $\mathcal{C}_{\text{sample}}$ are sampled from $\mathcal{C}_{\text{cover}}$. Full JSD measures the statistical imperceptibility between covertext and stegotext.
- *Half* JSD: Covertext dataset is split into two equal parts, $\mathcal{C}^1_{\text{cover}}$ and $\mathcal{C}^2_{\text{cover}}$. While the stego-models are trained on $\mathcal{C}^1_{\text{cover}}$, the evaluation is performed on $\mathcal{C}^2_{\text{cover}}$. It estimates imperceptibility under different but similar distributions.
- *Zero* JSD: Zero JSD is similar to Full JSD except that it uses a vanilla GPT-2 model as the $\mathbf{P}_{\text{LM}}$, without fine-tuning on the covertext. This evaluates the imperceptibility between the stegotext and the normal text.

However, PPL and JSDs are not good metrics for assessing different stegosystems. While PPL and JSD comparisons are suitable for supervised text generation tasks like machine translation, where a direct comparison with a golden text is possible, they face challenges in general text generation due to the absence of such direct relationships between generated output and training data. In steganography, where language distributions may vary significantly across methods, comparing PPLs and JSDs becomes impractical. Additionally, variated settings of language models, such as datasets, hyperparameters, and model architecture, will further complicate such comparisons, even if these models are within the same domain or task.

Though PPL and JSD *cannot* be used to compare generation ability among different methods, it still provides an indication of performance between different settings of the same method.

[2]Only supervised methods need training on covertext.

Methods	Training-free	IMDB					Twitter				
		BPW	PPL	JSD_{full}	JSD_{half}	JSD_{zero}	BPW	PPL	JSD_{full}	JSD_{half}	JSD_{zero}
RNN-Stega (LSTM)		1.978	10.23	30.33	33.12	38.27	2.556	13.04	39.92	38.97	48.10
		2.682	12.80	26.76	29.36	34.87	3.359	15.38	36.20	35.53	44.76
		3.351	17.02	22.66	25.72	30.28	4.139	19.78	32.17	31.75	39.19
VAE-Stega (BERT-LSTM)		1.972	9.68	34.50	36.47	38.53	2.247	10.06	46.07	45.82	46.61
		2.601	12.38	31.31	33.02	34.56	2.861	12.39	43.89	44.02	43.64
		3.199	16.31	30.03	31.49	32.82	3.438	16.13	42.12	42.54	40.87
ADG		4.931	56.22	18.24	21.19	22.86	5.702	63.86	25.92	25.35	27.68
NLS	✔	1.889	10.40	23.63	22.83	17.91	2.059	10.95	37.71	36.17	29.34
		2.531	12.90	22.08	21.28	16.73	2.806	14.01	36.61	35.17	29.45
		3.140	16.70	20.37	19.63	14.44	3.513	18.68	34.42	32.90	30.18
SAAC	✔	4.471	28.74	18.28	16.40	13.17	5.078	36.74	33.75	32.11	23.08
		4.749	37.89	18.04	16.06	11.49	5.299	43.35	33.42	31.82	22.33
		5.111	44.02	17.87	15.98	11.44	5.716	54.35	33.20	31.68	22.04
ours	✔	1.906	8.81	17.90	16.86	13.40	2.550	9.48	30.90	29.34	24.90
		2.420	13.70	18.37	17.37	13.67	3.265	14.44	30.99	29.45	25.32
		3.376	45.04	18.61	17.87	13.91	4.029	47.37	31.74	30.18	25.40

Table 1: Experimental results under different BPWs and datasets. JSDs are multiplied by 10^2. With the *same* BPW, lower JSD ↓ indicates better performance. There is no data available for smaller BPW in ADG or SAAC due to their design of high embedding rates. Our proposed method shows a reduced Psic Effect in comparison to prior methods as our JSDs rise with increasing BPW, whereas other methods see a decrease in JSDs.

Dataset	# sentences	Avg. Words	Avg. Bits
IMDB	494,716	26.70	146.43
Twitter	2,063,307	12.31	114.85

Table 2: Statistics of the preprocessed datasets.

4.3 Experimental Results

Table 1 summarizes the main results on the IMDB and Twitter datasets. Some methods are designed in such a way that it is inapplicable to adjust the BPW during sentence generation, especially for a small BPW, resulting in a lack of experiments with certain settings. This also poses a challenge for comparing metrics between similar BPWs.

Psic Effect An interesting finding in our results is the Perceptual-Statistical Imperceptibility Conflict Effect (**Psic Effect**) mentioned by Yang et al. (2021), a phenomenon in which an increase in BPW and PPL is associated with a decrease in JSD. As shown in Table 1, Figure 4, and Table 3, while techniques like NLS exhibit superior PPL and JSD scores, they produce lower-quality text and are more susceptible to detection by attackers. The existence of this unusual phenomenon implies that it will be more difficult for machines to detect stegotext when the text becomes more *chaotic* and *unintelligible*. This indicates the inconsistency of imperceptibility as perceived by humans and machines, as examples shown in Appendix C. We believe that previous methods may have fallen into some local optimum of covertext distribution, but do not achieve a balance between the quality and distribution of the stegotext.

In conclusion, the state-of-the-art metrics do not match our method in every case, but our stegotext is much more natural and **does not suffer from the Psic Effect**, as is evidenced in the case study (Fig. 4) and Section 4.5.

Furthermore, experiments with three variants of JSD have shown that our zero-shot method can be used to fool experts with specialized backgrounds and ordinary people without prior knowledge. The generalization ability of our methods ensures better performance than supervised methods in both few-shot and large-data scenarios.

4.4 Steganalysis

To assess the statistical imperceptibility of stego-methods, a classifier is usually deployed as a steganalyzer to distinguish between covertext and stegotext. Table 3 presents the results of our steganalysis. Most prior works have primarily focused on syntactic similarity, which we believe is insufficient as text evaluation should encompass both syntactic analysis and semantic analysis. Thus we utilize three distinct methods, namely TS-BiRNN (Yang

Secret Bitstream (Base64): cli2EARMSajA

Fully-supervised

RNN-Stega: Richards are also in the film and the film shows us the most disgusting and violent aspects of the movie.The plot revolves around a group of people who are attacked and murdered.

VAE-Stega: listen. the music and the music, the music, music of all the actors, the music, and the music! this movie is not worth your eyes.

ADG: Richian is condemned to discover the facts that blackmails people through them.

Training-free

NLS: There was so much I wanted to see in this film. I was hoping it was a little more like what I was hoping for from the original film and not what I do not expect.

SAAC: I was looking forward to going as Keanu. I really wanted to do a little red carpet before I die.

Ours: I was genuinely underwhelmed by the film, and I'm afraid the only thing that stood out to me was the final scene. I had been eagerly anticipating it, but unfortunately the rest of the movie fell flat for me.

Figure 4: Examples of stegotext generated by various methods about movie reviews with a similar BPW ≈ 2.5, except ADG and SAAC. Compared with other baseline methods, our method generates more reasonable sentences.

Methods	Syntactic		Semantic
	TS-BiRNN	R-BiLSTM-C	BERT-C
Fully-supervised			
RNN-Stega	94.02	93.88	96.50
VAE-Stega	94.75	95.65	96.17
Training-free			
NLS	84.60	86.21	92.25
ours	**80.29**	**84.34**	**89.61**

Table 3: Steganalysis with BPW ≈ 2.5. Each value is the classification accuracy by steganalyzer. Lower accuracy $\downarrow$ indicates better statistical imperceptibility.

et al., 2019b), R-BiLSTM-C (Niu et al., 2019) and BERT-C, to assess the performance of stego-methods in terms of syntactic similarity and semantic similarity. Among these methods, the BERT-C method, inspired by Wen et al. (2022), utilizes a BERT encoder (Devlin et al., 2019) to encode the input sequence and uses a linear classifier to determine whether the given sequence is a stegotext.

We exclude ADG and SAAC from our analysis since their BPW settings are difficult to adjust, leading to an unfair comparison. Moreover, due to their high BPW settings (BPW > 5), ADG and SAAC cause significant Psic Effect, which makes it difficult to classify their stegotext accurately through syntactic analysis. Even with extremely low text quality, some of these methods can achieve almost 50% detection accuracy, which indicates a perfect but unreasonable statistical imperceptibility.

In conclusion, the results show that our method outperforms most of the previous methods.

4.5 Language Evaluation

While statistical imperceptibility can measure the similarity between text distributions, human perception still plays a significant role in assessing the **perceptual imperceptibility** of the stegotext.

To simulate the human evaluation and ensure reproducibility, we conduct language evaluation on the generated stegotext of each method using large language models. First, we attempt to collect the stegotext generated with similar BPW ≈ 2.5 using the IMDB dataset[3]. We then randomly select two sentences from each set of stegotext to compare, denoted by $\texttt{<sent}_A\texttt{, sent}_B\texttt{>}$. Next, GPT-3.5 is used to determine which sentence in the pair is considered high-quality. To be more specific, we compare our method with each baseline and collect ratios of how much of our stegotext is deemed to be better than the others. We sample 10K sentence pairs between our method and each baseline. Appendix A.5 illustrates the details of the implementation of language evaluation.

Table 4 demonstrates three kinds of quality measurement conducted in our experiments:

(a) **Soundness** measures the extent to which one sentence is more *logical* and *meaningful*.

(b) **Relevance** measures the extent to which one sentence is more *relevant* to the specific topic.

(c) **Engagingness** measures the extent to which one sentence is more *appealing* and *engaging*.

Our three measurements are all close to 1.0 when compared to the ground truth covertext, which indicates a high level of similarity. Most supervised

[3]For ADG and SAAC, the BPW is set to 4.931 and 4.471.

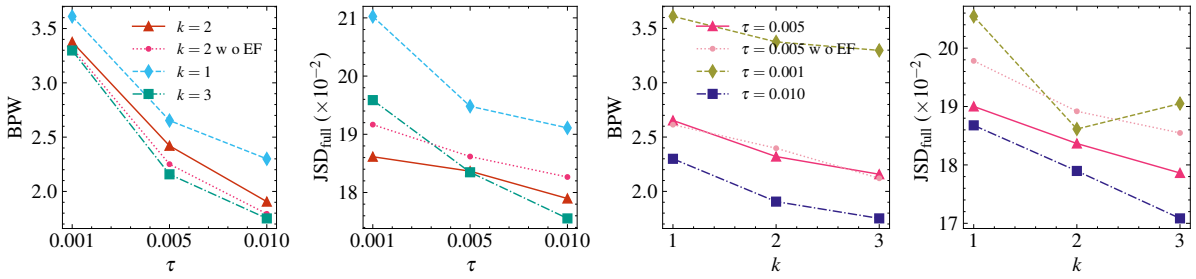

Figure 5: Curves of BPW and JSD$_{\text{full}}$ with respect to: (a) different threshold τ (b) different context size k

Eval.	GT	Fully-supervised			Training-free	
		RNN	VAE	ADG	NLS	SAAC
Sound.	0.788	3.812	8.363	8.042	1.373	1.654
Relev.	1.196	2.397	3.608	5.345	2.479	3.850
Engag.	1.157	5.386	9.267	7.224	1.924	2.380
Avg.	**1.047**	**3.865**	**7.080**	**6.870**	**1.926**	**2.628**

Table 4: Language evaluation results. Each value represents the ratio of how much our stegotext is considered better than the particular method. For example, a ratio of 3.812 means that the number of our stegotext considered better is 3.812× more than the other method.

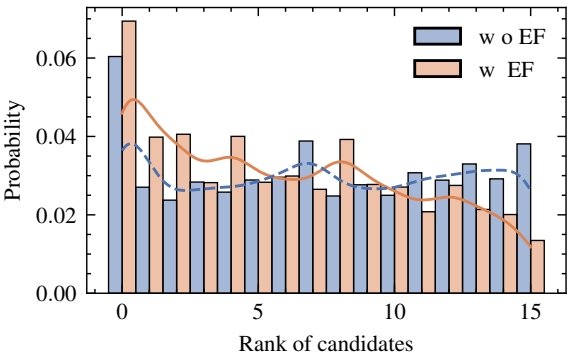

Figure 6: Distribution of the rank of selected candidates.

methods aim to approximate covertext distribution, which results in a large gap between readability and imperceptibility. Training-free methods, however, are more effective at balancing both perceptual and statistical imperceptibility. Furthermore, our method outperforms all other baselines with an average improvement of at least 1.926×.

4.6 Ablation Study

We analyze EF coding in terms of the probability of selecting a candidate from a pool of candidates. We replace every occurrence of the candidate with its rank of likelihood in the descending-sorted candidate pool in Fig. 6. It appears that words with lower ranks are more likely to be selected, which is generally of high word probability. The shift of word distribution results in slight improvements in BPW and JSDs, as shown in Fig. 5.

Figure 5 also shows the performance of our method under different hyper-parameter settings. We can observe from the results that an increase in threshold τ and context size k would result in a decrease in BPW and JSD. According to the results, a higher sentence quality would lead to a lower embedding rate but better statistical imperceptibility. This indicates a tradeoff between embedding rate and statistical imperceptibility exists.

5 Related Work

Linguistic steganography can be divided into three types (Fridrich, 2009), selection-based (Sajedi and Jamzad, 2008; Chen and Chen, 2019), edit-based (Chang and Clark, 2014) and generation-based steganography (Kang et al., 2020; Xiang et al., 2020). However, the low embedding rate of these methods hinders their practical application, giving rise to generation-based steganography.

To encode secret messages into textual data, early generative steganography utilized the Markov model to obtain conditional probability distributions (Moraldo, 2014) for text generation. Subsequently, the development of NLP has significantly improved distribution modeling. It was first proposed by Fang et al. (2017) to use a supervised recurrent neural network (RNN) to learn the statistical language model of natural texts with a fixed-length coding dictionary. Subsequently, Yang et al. (2019a) proposed a generation-based steganography based on RNN and Huffman coding. Yang *et al.* also proposed that different neural models make a difference, such as the generative adversarial network (Yang et al., 2020) and variational autoencoder (Yang et al., 2021). Zhang et al. (2021) and

Ding et al. (2023) further presented provably secure algorithms for better statistical imperceptibility.

While supervised methods attempt to fit models to the source distribution, the restriction of the source distribution leads to a high risk of generating similar but unintelligible text due to the complicated process of embedding secret messages. To address this limitation, models pre-trained on large-scale datasets are used. Ziegler et al. (2019) proposed that arithmetic coding be used with a pre-trained GPT-2 model. Dai and Cai (2019) further proposed an improved Huffman encoding algorithm, which is intended to adjust embedding rates dynamically. Shen et al. (2020) improved the previous works with self-adjusting arithmetic coding for near-imperceptible text. Nevertheless, all of these training-free approaches employ a similar paradigm for generating the context's *next sentence*, making them impractical in actual usage.

6 Conclusion

In this work, we present a zero-shot approach based on in-context learning for linguistic steganography. For the LLM to generate a similar sentence, samples of covertext serve as the context for the LLM in the QA. We also integrate several techniques to improve the text generation process for better readability and imperceptibility of the stegotext. Our experimental results exhibit the high imperceptibility of our approach compared to other methods. In addition, the stegotext is more intelligible and resembles the covertext in many aspects. We hope this work will be a step towards the more practical applications of linguistic steganography.

7 Limitations

Our approach has indeed demonstrated advantages in producing intelligible and innocent stegotext, yet we rely heavily on the generalization capabilities of LLMs. The computation overhead of LLMs is approximately $3\times$ to $5\times$ in comparison with prior works. There may be an additional vulnerability if real-time communication is required.

Further, we have not tested the proposed method in a relatively large embedding rate setting (e.g., BPW > 5). While we believe that stegotext generated under settings like these would be highly unintelligible and impractical in real life, the limits of our method have yet to be determined.

Data integrity is another essential requirement of our steganographic method. The incompleteness of the stegotext may result in invertible damage to the covert bitstream as in most previous works. Fortunately, textual data are less likely to suffer from lossy compression issues than other types of media (e.g., image, audio, and video).

8 Ethical Considerations

The use of pre-trained LLMs may exhibit potential issues, such as insults, political biases, or gender discrimination. It should also be noted that the data underlying our method may introduce additional ethical concerns. While we did not observe such problems during our experiments, users should be aware of these issues in practice.

Acknowledgement

This research is supported by the National Key R&D Program of China under grant (No.2022YFB2703001). Thanks to all the reviewers for their insightful suggestions.

References

Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. 1997. A concrete security treatment of symmetric encryption. In *Proc. 38th Annual Symposium on Foundations of Comput. Sci.*, pages 394–403. IEEE.

Christian Cachin. 1998. An information-theoretic model for steganography. In *Information Hiding*, pages 306–318, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ching-Yun Chang and Stephen Clark. 2014. Practical linguistic steganography using contextual synonym substitution and a novel vertex coding method. *Comput. Linguistics*, 40(2):403–448.

Mark Chapman and George Davida. 1997. Hiding the hidden: A software system for concealing ciphertext as innocuous text. In *Inf. and Commun. Secur.*, pages 335–345, Berlin, Heidelberg. Springer Berlin Heidelberg.

Xianyi Chen and Sheng Chen. 2019. Text coverless information hiding based on compound and selection of words. *Soft Comput.*, 23.

Falcon Dai and Zheng Cai. 2019. Towards near-imperceptible steganographic text. In *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, pages 4303–4308, Florence, Italy. Assoc. Comput. Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. 2019 Conf. North American Assoc.*

Comput. Linguistics, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Tina Fang, Martin Jaggi, and Katerina Argyraki. 2017. Generating steganographic text with LSTMs. In *Proc. ACL 2017, Student Res. Workshop*, pages 100–106, Vancouver, Canada. Assoc. Comput. Linguistics.

Jessica Fridrich. 2009. *Steganography in digital media: principles, algorithms, and applications*. Cambridge University Press.

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

Linjie Guo, Jiangqun Ni, Wenkang Su, Chengpei Tang, and Yun-Qing Shi. 2015. Using statistical image model for jpeg steganography: Uniform embedding revisited. *IEEE Trans. Inf. Forensics Security*, 10(12):2669–2680.

Huixian Kang, Hanzhou Wu, and Xinpeng Zhang. 2020. Generative text steganography based on lstm network and attention mechanism with keywords. *Electron. Imag.*, 2020(4):291–1.

R. Bala Krishnan, Prasanth Kumar Thandra, and M. Sai Baba. 2017. An overview of text steganography. In *2017 4th Int. Conf. Signal Process., Commun. and Netw. (ICSCN)*, pages 1–6.

Zinan Lin, Yongfeng Huang, and Jilong Wang. 2018. Rnn-sm: Fast steganalysis of voip streams using recurrent neural network. *IEEE Trans. Inf. Forensics Security*, 13(7):1854–1868.

Yunxia Liu, Shuyang Liu, Yonghao Wang, Hongguo Zhao, and Si Liu. 2019. Video steganography: A review. *Neurocomputing*, 335:238–250.

Yubo Luo, Yongfeng Huang, Fufang Li, and Chinchen Chang. 2016. Text steganography based on ci-poetry generation using markov chain model. *KSII Trans. Internet and Inf. Syst. (TIIS)*, 10(9):4568–4584.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics*, pages 142–150, Portland, Oregon, USA. Assoc. Comput. Linguistics.

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

Yan Niu, Juan Wen, Ping Zhong, and Yiming Xue. 2019. A hybrid r-bilstm-c neural network based text steganalysis. *IEEE Signal Process. Lett.*, 26(12):1907–1911.

Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. 2021. A survey of the usages of deep learning for natural language processing. *IEEE Trans. Neural Netw. Learn. Syst.*, 32(2):604–624.

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

Hedieh Sajedi and Mansour Jamzad. 2008. Cover selection steganography method based on similarity of image blocks. In *2008 IEEE 8th Int. Conf. Comput. and Inf. Technol. Workshops*, pages 379–384.

Jiaming Shen, Heng Ji, and Jiawei Han. 2020. Near-imperceptible neural linguistic steganography via self-adjusting arithmetic coding. In *Proc. 2020 Conf. Empirical Methods in Natural Lang. Process. (EMNLP)*, pages 303–313, Online. Association for Computational Linguistics.

Alexey Nikolaevich Shniperov and KA Nikitina. 2016. A text steganography method based on markov chains. *Autom. Control and Comput. Sci.*, 50:802–808.

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

Milad Taleby Ahvanooey, Qianmu Li, Jun Hou, Ahmed Raza Rajput, and Yini Chen. 2019. Modern text hiding, text steganalysis, and applications: A comparative analysis. *Entropy*, 21(4).

Jinyuan Tao, Sheng Li, Xinpeng Zhang, and Zichi Wang. 2019. Towards robust image steganography. *IEEE Trans. Circuits Syst. Video Technol.*, 29(2):594–600.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Peter J. M. van Laarhoven and Emile H. L. Aarts. 1987. *Simulated annealing*, pages 7–15. Springer Netherlands, Dordrecht.

W. Weber. 1978. Differential encoding for multiple amplitude and phase shift keying systems. *IEEE Transactions on Communications*, 26(3):385–391.

Juan Wen, Ziwei Zhang, Yu Yang, and Yiming Xue. 2022. Few-shot text steganalysis based on attentional meta-learner. In *Proceedings of the 2022 ACM Workshop on Information Hiding and Multimedia Security*, pages 97–106.

Lingyun Xiang, Rong Wang, Zhongliang Yang, and Yuling Liu. 2022. Generative linguistic steganography: A comprehensive review. *KSII Trans. Internet and Inf. Syst. (TIIS)*, 16(3).

Lingyun Xiang, Shuanghui Yang, Yuhang Liu, Qian Li, and Chengzhang Zhu. 2020. Novel linguistic steganography based on character-level text generation. *Mathematics*, 8(9):1558.

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

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

Zhongliang Yang, Ke Wang, Jian Li, Yongfeng Huang, and Yu-Jin Zhang. 2019b. Ts-rnn: Text steganalysis based on recurrent neural networks. *IEEE Signal Process. Lett.*, 26(12):1743–1747.

Zhongliang Yang, Nan Wei, Qinghe Liu, Yongfeng Huang, and Yujin Zhang. 2020. Gan-tstega: Text steganography based on generative adversarial networks. In *Digital Forensics and Watermarking*, pages 18–31, Cham. Springer International Publishing.

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

Zachary Ziegler, Yuntian Deng, and Alexander Rush. 2019. Neural linguistic steganography. In *Proc. 2019 Conf. Empirical Methods Natural Lang. Process. and 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, pages 1210–1215, Hong Kong, China. Assoc. Comput. Linguistics.

George Kingsley Zipf. 1999. *The psycho-biology of language: An introduction to dynamic philology*, volume 21. Psychology Press.

A Detailed Implementation

A.1 Edge Flipping Coding

The pseudo-code for Edge Flipping Coding is presented in Algorithm 2 for encoding, and Algorithm 3 for decoding.

Algorithm 2 Edge Flipping Encoding

Input: Bitstream $\boldsymbol{m}$.
Output: EF bitstream $\boldsymbol{m}'$.

Output bitstream $\boldsymbol{m}' \leftarrow \emptyset$
Bit state $s \leftarrow 0$
for $b \in \boldsymbol{m}$ **do**
  if $s = b$ **then**
    $\boldsymbol{m}' \leftarrow \boldsymbol{m}' \cup \{0\}$
  else
    $\boldsymbol{m}' \leftarrow \boldsymbol{m}' \cup \{1\}$
  end if
  ▷ Update bit state
  $s \leftarrow b$
end for

Algorithm 3 Edge Flipping Decoding

Input: EF Bitstream $\boldsymbol{m}'$.
Output: Bitstream $\boldsymbol{m}$.

Output bitstream $\boldsymbol{m} \leftarrow \emptyset$
Bit state $s \leftarrow 0$
for $b' \in \boldsymbol{m}'$ **do**
  if $b' = 1$ **then**
    $s \leftarrow \sim s$
  end if
  ▷ Append bit state
  $\boldsymbol{m} \leftarrow \boldsymbol{m} \cup \{s\}$
end for

Since EF coding is a bijection, it can be applied multi-round to the original bitstream. The final bitstream is the one with the fewest 1s.

A.2 Bitstream Extraction

We provide the bitstream extraction algorithm of the embedding module in Algorithm 4.

A.3 In-Context QA

The complete in-context QA templates of our method are presented in Fig 7. We instruct the LLM to follow the XML-style response for parsing results. Covertext context is placed in the `{CONTEXT}` slot. There is also a hint of the corpus placed on the `{CORPUS}` slot.

```
«SYS»
You are an expert at mimicing the language style of others (e.g., the use of
words and phrases).  And you are a helpful and respectful assistant.

Users will input sentences from a given corpus.  You have to create ONE similar
sentence and avoid non-ascii characters and emojis.  This is very important to
the user's career.

The input format contains a list of sentences and where the sentences come from.
For example:
<CORPUS>{CORPUS}</CORPUS>
<CONTEXT>
Example sentence 1.

Example sentence 2.
</CONTEXT>

Your output should be like:

The generated similar sentence in ONE LINE is:

«/SYS»

[INST]<CORPUS>{CORPUS}</CORPUS>
<CONTEXT>
{CONTEXT}
</CONTEXT>[/INST]

The generated similar sentence in ONE LINE is:
```

Figure 7: QA templates for in-context stegotext generation. Slot `{CORPUS}` and `{CONTEXT}` would be replaced by the corpus name and the selected context from covertext in the runtime.

Algorithm 4 Information Extraction Algorithm

Input: stegotext $\boldsymbol{y} = [y_1, \ldots, y_n]$, threshold τ.
Output: Bitstream $\boldsymbol{m}$.

Timestep $t \leftarrow 1$, output bitstream $\boldsymbol{m} \leftarrow \emptyset$
while not the end of $\boldsymbol{m}$ **do**
  ▷ `Compute conditional probs`
  $\boldsymbol{p} \leftarrow \mathbf{P}_{\text{stega}}(x_t \mid x_{<t})$
  ▷ `Prune candidate words`
  $\boldsymbol{c} \leftarrow [c_i \mid \boldsymbol{p}(c_i) \geq \tau]$
  ▷ `Huffman encoding`
  $H \leftarrow \text{Huffman}(\boldsymbol{c}, \boldsymbol{p})$
  ▷ `Select candidate`
  $\boldsymbol{s} \leftarrow$ Bits $\boldsymbol{s}$ whose binary representation matches certain word $c \in H$
  $\boldsymbol{m} \leftarrow \boldsymbol{m} \cup \boldsymbol{s}, t \leftarrow t + 1$
end while

A.4 Baselines

We rebuild the entire benchmark pipeline for a fair comparison between different stego-methods. The following are the implementation details:

- **RNN-Stega**: We refactor the official codebase from `TensorFlow` to `PyTorch`. Hyper-parameters: lr=0.001, dropout=0.5, 30 epochs, 768 LSTM units, and 2 layers of LSTM.
- **VAE-Stega**: We rebuild the VAE pipeline using PyTorch following the official codebase. Hyper-parameter setting: lr=3e-4, 20 epochs, hidden size of 768, $\text{DistilBERT}_{\text{BASE}}$ as encoder, 2 layers of LSTM as decoder.
- **ADG**: We reproduce ADG based on the official codebase. Hyper-parameter setting: lr=0.001, 30 epochs, 768 LSTM units, and 2 layers of LSTM.
- **NLS**: We reproduce NLS based on the official codebase. We use `gpt2-medium` model from Huggingface Library as the base model.
- **SAAC**: We reproduce SAAC based on the official codebase. We use `gpt2-medium`

Eval.	GT	Fully-supervised			Training-free	
		RNN	VAE	ADG	NLS	SAAC
Sound.	.4406	.7922	.8932	.8894	.5786	.6232
Relev.	.5446	.7056	.7830	.8424	.7126	.7938
Engag.	.5364	.8434	.9026	.8784	.6580	.7041

Table 5: Raw results of language evaluations.

Metrics	α			β		
	1.00	1.25	1.50	0.25	0.50	1.00
BPW	2.149	2.420	2.462	2.116	2.420	2.511
JSD_{full}	18.11	18.37	18.66	18.28	18.37	19.02

Table 6: Ablation on the annealing search factor α and repeat penalty factor β.

model from Huggingface Library as the base model. The hyper-parameter δ is set the same as the original paper: 0.01, 0.05 and 0.10.

A.5 Language Evaluation

In this section, we provide the actual prompts for ChatGPT to discriminate the quality of the sentence pair <$\texttt{sent}_A$,$\texttt{sent}_B$>. Figure 8 illustrates the prompting templates. We also randomly swap the sentences of a pair before calling the ChatGPT API.

The raw data of language evaluations is presented in Table 5, where each grid represents the percentage of how much ours is considered better.

A.6 Ablation

We present the ablation study of the factors α and β of annealing search and repeat penalty in Table 6.

B Efficiency

Table 7 shows the computational overhead of baselines and our method. Due to the use of LLMs, our method requires a cost of $3\times$ to $5\times$ more time.

C Examples

Figure 9 provides more examples of the generated stegotext of each method. In comparison with other approaches, ours has better sentence quality.

Method	RNN	VAE	ADG	NLS	SAAC	ours
Avg. Time (s)	1.55	2.16	1.87	2.89	3.38	8.41

Table 7: The average time cost of generating one steganographic sentence using a single RTX3090.

D Scientific Artifacts

The licenses for all the baselines are listed below: RNN-Stega (MIT License), VAE-Stega (MIT License), ADG (MIT License), NLS (MIT License), SAAC (GPL-3.0 License). The licenses for models are listed below: BERT (Apache 2.0 License), GPT-2 (Modified MIT License), LLaMA2-Chat-7B (LLAMA 2 Community License).

```
The user will input 2 sentences.
You have to decide which one is more logically sound and meaningful.  This is
important to the user's career.

The result should be in JSON.
It should contain the key "result" with value being either 0 or 1, indicating
the first or second one.

1.  <sent_A>

2.  <sent_B>
```

(a) Template for evaluating soundness.

```
The user will input 2 sentences.
You have to decide which one is more likely to be part of some "{TOPIC}".  This
is important to the user's career.

The result should be in JSON.
It should contain the key "result" with value being either 0 or 1, indicating
the first or second one.

1.  <sent_A>

2.  <sent_B>
```

(b) Template for evaluating relevance. Here {TOPIC} is used to restrict the evaluation to specific topics. For IMDB, we use "movie reviews" as the topic.

```
The user will input 2 sentences.
You have to decide which one is more engaging and appealing to readers.  This is
important to the user's career.

The result should be in JSON.
It should contain the key "result" with value being either 0 or 1, indicating
the first or second one.

1.  <sent_A>

2.  <sent_B>
```

(c) Template for evaluating engagingness.

Figure 8: Templates for language evaluations.

Secret Bitstream (Base64): xB0CEVgADkP2YoA=

Fully-supervised

RNN-Stega: Sin City is a great film, and the story is told from the viewer.The plot is pretty simplistic, and has no redeemable qualities to it whatsoever.
VAE-Stega: short! i have the feeling that i have a copy on dvd. i would have been watching it, and i'm a huge fan.
ADG: Sinbad?No?So HOAME.horrible movie.Acting wise as it describes is hard.

Training-free

NLS: And while the latter is certainly true, I think it is also fair to say that CBS has done a pretty good thing. The network has been a champion for science and for health.
SAAC: The Center for Public Integrity of West Allis, Wisconsin, and Wythe Law School of Louisville, Kentucky reported this week that cigarette makers have guities.

Ours: All countries have an ugly side, just as ours does. It's unfair to say that only the Philippines has this issue, though. We are all capable of such shortfalls.

(a)

Secret Bitstream (Base64): pTJAjKMijGEMkA==

Fully-supervised

RNN-Stega: Adding insult from this review, it's not as if it is shot on the big screen.The problem was that the film was not intended.
VAE-Stega: four years later he're going to get to the job, and then he becomes a suspect that he was not a guy and he has a very strange and very good
ADG: Adding Wes Anderson is a good story,or a movie that you will get used to.In other words, it's a great

Training-free

NLS: The only thing missing from the film's budget is a big name star, and that will have to be one of the film-makers' two choices.
SAAC: The cleanroom in which the court martial took place is flanked by treetops, allowing for ...

Ours: Cheaper doesn't always have to equal inferior, particularly in the case of indie films, where creators are often pushed to maximize the impact of limited resources.

(b)

Secret Bitstream (Base64): CYmMLJkPqMO4CHaxyEBBXrkDs5Iw

Fully-supervised

RNN-Stega: Extreme to the point where a film that was made was a huge waste.The film's ending is an absolute disaster, but it was very disappointing to see a man that could easily be a good thing.The film had some pretty interesting elements, and the movie was very well done, but that's not enough for it.
VAE-Stega: if you like "the rose", "love" (the "i have you, you're better! ")..... the ending is very funny. this is no exception. it'is so bad!... the movie is bad.. bad!!.
ADG: Extremely directed by Gulna Wong who brings the thought of interest for a free part of her life.Avoid this film as a film for the best of heartless Arts cinema lover.

Training-free

NLS: It was only a matter for him to find a wife to take care in the meantime and then the marriage will be broken up and it can be said he'll lose all his love for his beloved. I mean it makes perfect sense, it seems to make more sense that he has to leave her for the woman he loves.
SAAC: She wants to break things up with him. I've always seen this as perfect because it's a result of her deep love for man in this movie. However, not everybody likes watching two men break up, you know?

Ours: Oh my gawd, stop dting married men! Like seriously, why do we keep falling into this trap over and over? Anglea basset's chracter should totall be more concerned with the fact that her husband abandoned hre for another woma than some mediocore romance flic. Overall, I would give this movie a big fat C- for wasted poasbility and a little more thought.

(c)

Figure 9: Examples of stegotext generated by various methods on the topic of "movie reviews".