Semantic Steganography: A Framework for Robust and High-Capacity Information Hiding using Large Language Models

Anonymous ACL submission

Abstract

In the era of Large Language Models (LLMs), generative linguistic steganography has become a prevalent technique for hiding information within model-generated texts. However, traditional steganography methods struggle to effectively align steganographic texts with original model-generated texts due to the lower entropy of the predicted probability distribution of LLMs. This results in a decrease in embedding capacity and poses challenges for decoding stegos in real-world communication channels.

To address these challenges, we propose a semantic steganography framework based on LLMs, which constructs a semantic space and maps secret messages onto this space using ontology-entity trees. This framework offers robustness and reliability for transmission in complex channels, as well as resistance to text rendering and word blocking. Additionally, the stegos generated by our framework are indistinguishable from the covers and achieve a higher embedding capacity compared to state-of-the-art steganography methods, while producing higher quality stegos.

1 Introduction

With the rapid updates and iterations of Large Language Models (LLMs)(Touvron et al., 2023; Du et al., 2022), texts generated by LLMs flood cyberspace, providing a thriving environment for generative linguistic steganography(Yang et al., 2021, 2019; Dai and Cai, 2019; Ziegler et al., 2019; Shen et al., 2020; Kaptchuk et al., 2021; de Witt et al., 2023; Ding et al., 2023). As a technique for hiding information in model-generated texts, mainstream steganography methods(Kaptchuk et al., 2021; de Witt et al., 2023; Ding et al., 2023) focus on aligning steganographic texts (stegos for short) with original model-generated texts (covers for short).

However, current steganography techniques have two major weak points.

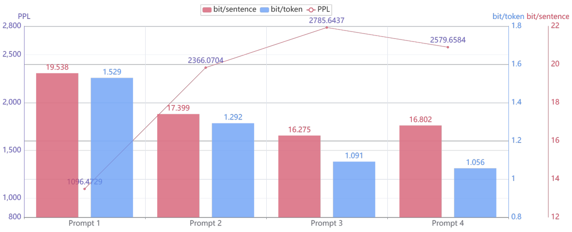
Low Entropy. Given the same text prefix, the entropy of the predicted probability distribution of LLMs is likely to be lower than that of GPT-2(Alec Radford, 2019) or BERT(Devlin et al., 2019). The main reason for this is that LLMs have learned more data and are able to make more accurate predictions. But for SOTA provably secure steganography algorithms(Kaptchuk et al., 2021; de Witt et al., 2023; Ding et al., 2023), entropy is an upper bound on the embedding capacity. A large decrease in entropy leads to a dramatic decrease in embedding capacity. The entropy gap between LLMs of different performances is also quite pronounced. As figure 1 shows, with the same steganography method Arithmetic Coding(AC)(Ziegler et al., 2019), the embedding rate of ChatGLM-2-6B is about 1/4 $\sim$ 1/5 lower than that of ChatGLM-2-6B-int4. With stricter top-k truncation and more detailed prompting, the embedding rate may decrease further.

Not Robust. When applying these steganography methods to real-world communication channels, particularly in social networks, we have found that most received stegos cannot be decoded. This problem is caused by three main reasons:

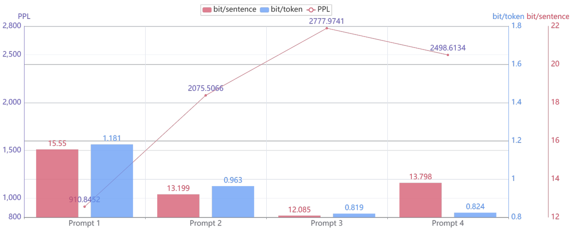
(1) **Text Rendering.** It involves the transcoding and merging of format control characters like spaces, tabs, and newlines, which may be stripped when at the beginning or end of a sentence. It also includes deceptive practices with line breaks and tabs, which can confuse the decoding system. While the transcoding and stripping process may be reversible, the merging is not, leading to inevitable decoding errors.

(2) **Word Blocking.** Social communication channels censor specific words or phrases deemed inappropriate, offensive, or undesirable. It is a common feature in online platforms and messaging apps. However, if words in the stegos are removed, decoding will fail.

(3) **Ambiguous Tokenizing.** It occurs when the



(a) AC + ChatGLM2-6B-int4



(b) AC + ChatGLM2-6B

Figure 1: Perplexity(PPL) and embedding rate of stegos generated by Arithmetic Coding(AC)(Ziegler et al., 2019) with ChatGLM2-6B-int4/ChatGLM2-6B. The left axis represents PPL, while the right two axes represent embedding rate, estimated in bits per sentence and bits per token. Prompt 1 is null, while prompts 2-4 require the model to generate a single given word, namely "the", "like", or "Washington", respectively.

tokenizer used in this process results in a single sentence having two equivalent lists of token IDs. This frequently occurs in LLMs.

The challenges of applying LLMs to state-of-the-art steganography methods highlight the lack of robustness of symbolic token-level embedding in cyberspace. It is necessary to search for a steganography method that can generate robust stegos. Therefore, we have proposed a semantic steganography framework based on LLMs. This framework primarily constructs a semantic space and uses LLMs to generate responses that belong to a subset of that space. To ensure the rationality of LLMs' output and their embedding capacity, we mapped the secret messages onto a semantic space using ontology-entity trees. During the decoding phase, the semantic information in stegos will be retrieved and converted back into secret messages.

Compared to the symbol-based steganography techniques, our framework has the following advantages:

- Our framework is more reliable and robust for transmission in network environments. The stegos generated by our method are able to resist ambiguous tokenizing and text rendering. As for word blocking, our steganography techniques can be decoded correctly with a high probability.

- The stegos in our framework are directly generated by LLMs, making them completely indistinguishable from the covers. Compared to symbolic imperceptibility, we should focus on semantic imperceptibility. We can ensure semantic imperceptibility by using provably secure sampling methods(de Witt et al., 2023; Kaptchuk et al., 2021; Ding et al., 2023).

- Our framework achieves a higher embedding capacity than state-of-the-art steganography methods under the same prompt and generation configurations while producing higher-quality stegos.

2 Methods

2.1 Construct the Semantic Space

Semantic space is a set in which sentences are represented based on their meanings and relationships. And the first encoding step of steganography is to draw a sample from the semantic space.

Various methods exist for constructing a semantic space.

(1) **Classifiers.** In previous work(Zhang et al., 2021), classifiers were used to control the semantic information at the sentence level, but such classifiers need training and are not easy to share with the receiver. To ensure objectivity, avoid using emotions or main themes as they are not realistic due to limited semantic space and restricted embedding capacity. Additionally, the meanings inside the sentence are mostly unused.

(2) **Embeddings.** The embedding output of language models can be used to construct a semantic space, but this method seems to be too sensitive and difficult to design. While this does not affect the encoding method, it can confuse the decoding process. We believe that using the embedding output of language models is feasible and requires further exploration.

(3) **Entities.** The ontology-entity tree is a simple and effective way to construct semantic space. The paths in this tree represent entities and offer a useful and robust decoding method. The entire encoding and decoding process is clear and easy to understand. The capacity of steganography is associated with the number of entities in this tree,

so adding more entities is a viable approach to expanding capacity.

Therefore, we use the ontology-entity tree to organize entities, which are the leaf nodes of the tree and contain the most information in a sentence, to construct the semantic space.

For any entity $e_i \in \mathcal{E}$, we construct an extraction method Ext_{e_i}. Using the extraction method Ext_{e_i} we can extract the number of entity e_i that appears in a sentence, denoted as $Ext_{e_i}(S) = n_i$. We define the **type** of a sentence as follows:

The **type** of sentence S is $e_1^{n_1} ... e_{|\mathcal{E}|}^{n_{|\mathcal{E}|}}$, where $e_i \in \mathcal{E}$ is an entity, $n_i = Ext_{e_i}(S)$ is the times that e_i appears in sentence S.

For instance, consider the sentence "An apple a day, keeps the doctor away" with the entities "apple" and "doctor". From this, we can determine that the **type** of this sentence is apple^{1}doctor^1.

We define the length of a **type** $|T|$ as the number of entities inside the sentence.

$$|T| = \sum_{i=1}^{|\mathcal{E}|} n_i \tag{1}$$

For the sake of clarity, we provide a definition of the partial order relation between types: **type** $T^{(1)} = e_1^{n_1^{(1)}} \cdots e_{|\mathcal{E}|}^{n_{|\mathcal{E}|}^{(1)}}$ is not greater than **type** $T^{(2)} = e_1^{n_1^{(2)}} \cdots e_{|\mathcal{E}|}^{n_{|\mathcal{E}|}^{(2)}}$ if and only if

$$\forall i \in \{1, 2, \cdots, |\mathcal{E}|\}, n_i^{(1)} \leq n_i^{(2)} \tag{2}$$

We can also define an add operation on the **type**, which represents combining 2 sentences into one.

$$T^{(1)} + T^{(2)} = e_1^{n_1^{(1)} + n_1^{(2)}} \cdots e_{|\mathcal{E}|}^{n_{|\mathcal{E}|}^{(1)} + n_{|\mathcal{E}|}^{(2)}} \tag{3}$$

Sentences with the same type are highly correlated as they are likely referring to the same entities and have a relationship. We define **class** $\mathcal{C}$ to denote the set of possible sentences that share the same **type**.

$$\mathcal{C}(e_1^{n_1} \cdots e_{|\mathcal{E}|}^{n_{|\mathcal{E}|}}) = \{S | \textbf{type}(S) = e_1^{n_1} \cdots e_M^{n_M}\} \tag{4}$$

In the end, the semantic space is defined as the set of all possible classes.

$$\mathcal{S} = \{\mathcal{C}(e_1^{n_1} \cdots e_{|\mathcal{E}|}^{n_{|\mathcal{E}|}}) | e_i \in \mathcal{E}, n_i \in \mathbb{N}_+\} \tag{5}$$

Instead of generating a sentence with specific attributes, we prefer to determine and arrange the entities that should appear in the output.

Algorithm 1 Assign($\mathcal{T}_o, T_{pre}, p(\cdot)$) probabilities to nodes

```
Input: Ontology-entity tree $\mathcal{T}_o$, type prefix $T_{pre}$,
    empirical distribution $p(\mathcal{C}(T))$
Output: Probabilities of nodes $P(\cdot)$
··· Initial probabilities of nodes in tree
for $node \in \mathcal{T}_o$ do
    $P(node) \leftarrow 0$
end for
··· Assign probabilities to entities
$sum \leftarrow 0$
for $e \in \mathcal{E}$ do
    for $\mathcal{C}(T) \in \mathcal{S}$ and $p(\mathcal{C}) \neq 0$ do
        if $T_{pre} + e \leq T$ then
            $P(e) \leftarrow P(e) + p(\mathcal{C}(T))$
        end if
    end for
    $sum \leftarrow sum + P(e)$
end for
··· Assign probabilities to stop sampling
$P(stop) \leftarrow p(\mathcal{C}(T_{pre}))$
$sum \leftarrow sum + p(\mathcal{C}(T_{pre}))$
··· Normalization
for $e \in \mathcal{T}_o \cup \{stop\}$ do
    $P(e) \leftarrow P(e)/sum$
end for
··· Accumulate probabilities to nodes
for $Entity \quad e \in \mathcal{T}_o$ do
    $parent \leftarrow e.parent$
    while $parent \neq T.ROOT$ do
        $P(parent) \leftarrow P(parent) + P(e)$
    end while
end for
return $P(\cdot)$
```

2.2 Sample from the Semantic Distribution

This section discusses a secure method of sampling from the semantic space.

For provably secure symbolic steganography methods such as METEOR(Kaptchuk et al., 2021), MEC(de Witt et al., 2023), and DISCOP(Ding et al., 2023), it is expected that the model-generated stegos are indistinguishable from the model-generated covers. That means $D_{KL}(p(cover)||p(stego)) = 0$ (Cachin, 1998). To ensure the KL divergence is 0, secure sampling methods are often designed. As our method does not alter the sampling strategy of LLM, the stegos remain the same as the covers.

Although there is no difference between stegos

Algorithm 2 Sample($\mathcal{T}_o, C, p(\cdot), F_{key}$) a type from semantic space

```
Input: Ontology-entity tree $\mathcal{T}_o$, empirical distribution $p(\mathcal{C}(T))$, cipher bits $C$, PRF $F_{key}$
Output: Target type $T_t$
$\cdots$ Randomize cipher bits
$B \leftarrow F_{key}(C)$
$T_t \leftarrow null$
$\cdots$ Select entities one by one
while $pointer \neq stop$ do
    $\cdots$ Get probabilities of nodes in tree
    $P(\cdot) \leftarrow$ Assign($\mathcal{T}_o, T_t, p(\mathcal{C}(T))$)
    $sum \leftarrow 0$
    $\cdots$ Select nodes layer by layer
    $pointer \leftarrow ROOT$
    while $pointer.child \neq null$ do
        for $node \in pointer.child$ do
            if $sum + P(node) \geq \sum_{i=1}^{n} b_i * 2^{-i}$ then
                $pointer \leftarrow node$
                $B \leftarrow B[n:]$
                break
            end if
            $sum \leftarrow sum + P(node)$
        end for
        if $sum < b_i/2$ then
            $pointer \leftarrow stop$
        end if
    end while
    $T_t \leftarrow T_t + pointer$
end while
```

and covers from a symbolic perspective, there is still a sampling issue from a semantic perspective. To begin with, we need to consider the empirical semantic distribution.

As the semantic space is made up of classes that represent different types, we must extract sentence types from a large corpus and assign the frequency of each class to its probability. In cases where the entities within a sentence cannot be obtained, a prediction model is used. An empirical semantic distribution can then be constructed by either counting sentences or training a model.

To sample from this distribution, randomized methods are necessary to ensure secure sampling. Pseudo-random functions (PRFs) are commonly used to convert a secret bit stream into a pseudo-random bit stream that follows a uniform distribution. The definition of PRFs is as follows.

$F_{key}: \{0,1\}^s \rightarrow \{0,1\}^s$ is PRF if for all P.P.T. classifiers C and key,

$$|Pr(C^{F_{key}}(1^s) = 1) - Pr(C^{\mathcal{O}}(1^s) = 1)| \leq \frac{1}{poly(s)} \tag{6}$$

where $\mathcal{O}$ is an oracle that randomly generates bits and $poly(\cdot)$ denotes all polynomial functions.

The first step in sampling is to use a key and a PRF to invert the ciphertext into a uniformly distributed bitstream. Then the problem is to map the bit stream into entities.

Since a uniformly distributed bit stream $B = [b_1, b_2, \cdots, b_{|B|}]$ can be mapped to a decimal $\hat{B} = \sum_{i=1}^{|B|} 2^{-i} b_i \sim Unif[0,1]$. Then AC can be used to map the decimal $\hat{B}$ to an interval which represents a **class** in semantic space. In our practice, the algorithms used for sampling are referenced in Alg. 1 and 2. For layers of an ontology entity tree, secret bits can be embedded in a path in the tree representing an entity (from ROOT to that entity). It is possible to sample entities one by one, and these entities are finally combined to form a **type**, then let LLM generate a stego belonging to the **class** that relates to this **type** in semantic space. In this way, secret bits can be sequentially embedded in nodes of the ontology-entity tree.

2.3 Feedback CoT for Stego Generation

A **class** is chosen for LLMs to generate after sampling from the semantic distribution. However, making LLMs generate sentences that belong to the **class** is not always successful. A rejection sampling method must be used for LLMs to generate correct sentences.

We proposed a method called **feedback Chain of Thought(CoT)** to increase the success rate of generation.

Since each entity corresponds to a path in the ontology entity tree, additional information describing the entities will be part of the prompt. In the generation process, the LLM for generating stegos is called Generation Agent (GA). To check whether the generated stego satisfies the sampled **class**, another LLM called Check Agent (CA) is used. For each generation loop, CA will return a hint for regeneration or it will consider the generated sentence compliant and return the approval. With the feedback from CA, GA is able to efficiently adjust the generated sentence and quickly converge to a correct version.

Feedback CoT reduces the number of iterations and saves a lot of time in the experiment. A result

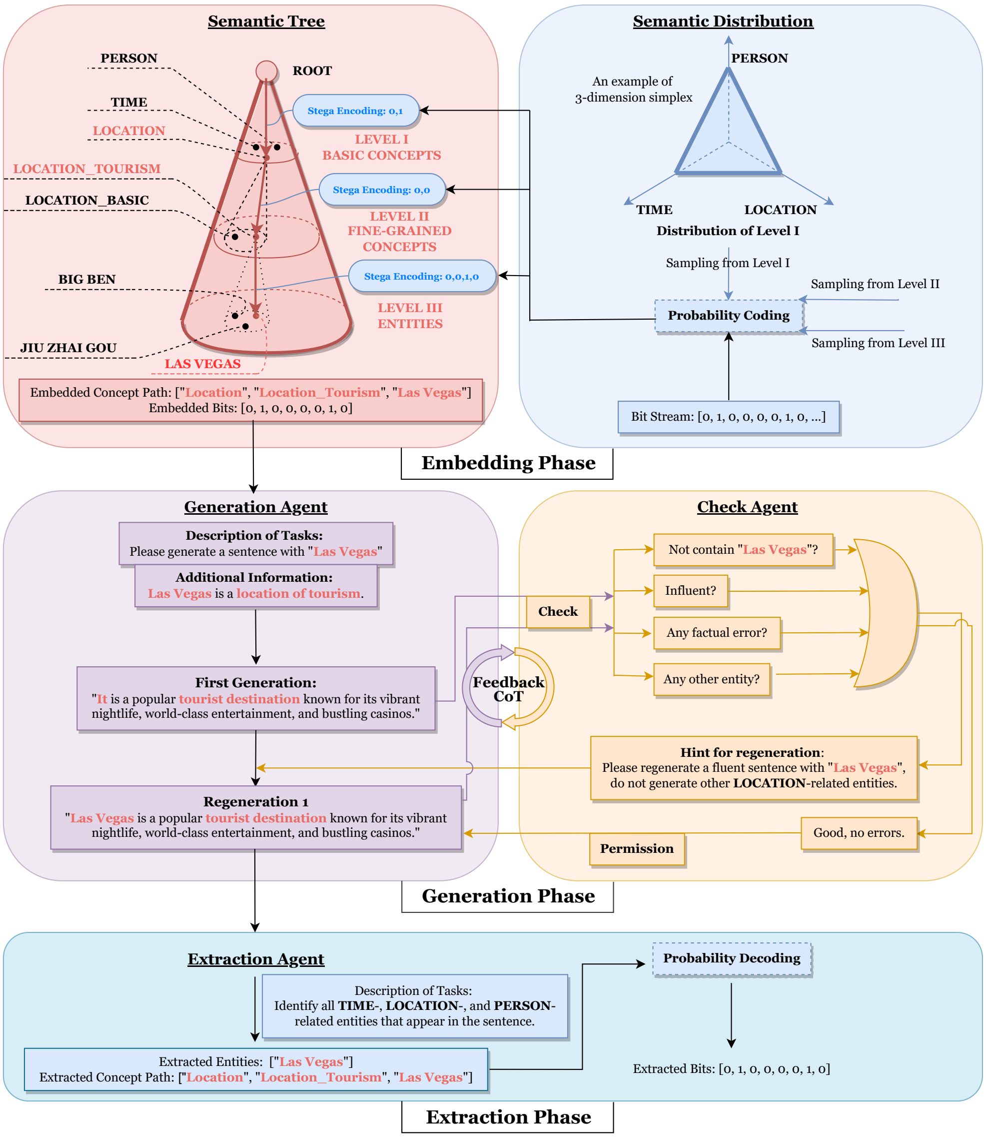


Figure 2: Workflow of our framework, with a simple example. A **type** "**Las Vegas**[1]" is selected according to the secret bit stream. The LLMs follow the instructions and generate a sentence "**Las Vegas** is a popular **tourist destination** known for its vibrant nightlife, world-class entertainment, and bustling casinos. ", which belongs to the **class** of **type** "**Las Vegas**[1]".

about feedback CoT in section 3.1 shows that it is able to decrease the perplexity of generated stegos and reduce the times of regeneration.

2.4 Workflow of Our Framework

As Fig.2 shown, our framework works in 3 phases.

(1) **Embedding Phase**: With a secret bit stream and a provably secure probability coding method, we use Alg.2 to sample a **type** from the semantic distribution. During the sampling process, the paths of entities that selected are preserved for the next phases.

(2) **Generation Phase**: We use the paths of entities and a description of task for GA to generate a primitive stego. Then the feedback loop starts running. CA generates a hint for regeneration and GA is instructed by CA to correct the stego. Finally, CA confirms that stego meets the requirements and gives permission to proceed to the next step.

(3) **Extraction Phase**: An LLM named extraction agent(EA) is instructed to extract the **type** of sentence. Since the **type** represents an interval of probability $[l, h]$, the decoding involves computing bit stream $B \in \{0, 1\}^n$ that satisfies:

$$\sum_{i=1}^{n} b_i * 2^{-i} \in [l, h] \tag{7}$$

$$\sum_{i=1}^{n} b_i * 2^{-i} \pm b_n * 2^{-n} \notin [l, h] \tag{8}$$

Metrics	PPL	Distinct-3	GPT-4 score
AC-6B	2065.73	0.8024	5.6381
AC-6B-int4	2206.96	0.8009	5.3290
RS-6B	2027.34	0.8050	5.6419
RS-6B-int4	2085.65	0.8001	5.1578
Ours-6B	**869.79**	**0.8753**	**7.3624**
Ours-6B-int4	855.70	0.8742	7.1527

Table 1: Linguistic quality of the generated texts. AC(Ziegler et al., 2019) stands for arithmetic coding of generated stegos and RS stands for randomly generated covers. 6B and 6B-int4 stand for ChatGLM2-6B and its 4-bit quantified version.

3 Experiment & Result

We use ChatGLM2-6B and ChatGLM2-6B-int4 as agents. ChatGLM2-6B-int4 is a weaker version of ChatGLM2-6B, but this model is extremely fast and only uses 6GB of GPU RAM.

Ontology-entity tree is hand-crafted by the authors with the help of PaddleNLP UIE model(PaddleNLP, 2021). We use UIE to extract the entities in sentences and place them in the subnode of the corresponding ontology. Dataset used to construct the empirical semantic distribution is LCCC(Wang et al., 2020), a large scale cleaned dataset containing 12 million daily conversations.

AC(Ziegler et al., 2019) is used as a baseline in our experiments for 2 reasons. It produces high quality stegoes and the embedding capacity is close to the entropy limit.

For our experiments, we used a server equipped with 4 RTX 3090 GPUs. The experiments consist of 3 parts. First, we measured the quality of our stegos and compared them with stegos generated by AC(Ziegler et al., 2019) and model-generated covers by random sampling. Then we tested the robustness of our method and AC against attacks that ignore/preserve the semantics of the original sentence.

3.1 Quality of Stegos

The linguistic quality of stegos is estimated by perplexity(PPL), distinct-n, and GPT-4 semantic rationality score. The PPL and distinct-n are calculated as follows:

$$PPL = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log p(x_i|\mathbf{x}_{1:i-1})\right) \quad (9)$$

$$\text{distinct-}n = \frac{count(unique\, ngrams)}{count(ngrams)} \quad (10)$$

Metrics	ER		MSR
	bit/sentence	bit/token	
AC-6B	2.5695	0.1788	0.459
AC-6B-int4	3.6863	0.2648	0.376
RS-6B	-	-	0.463
RS-6B-int4	-	-	0.457
Ours-6B	**28.5088**	0.3958	**0.893**
Ours-6B-int4	27.8945	**0.4130**	0.884

Table 2: Embedding rate and mission success rate of AC, RS and ours. For symbolic steganographic methods like AC, the embedding rate is usually estimated in bits per token. But for our framework, we embedded bits into entities and arranged them into a sentence, so we measure the embedding rate in bit per sentence. The bit per token of our framework is an average expectation.

PPL represents the fluency of stegos and distinct-n measures the diversity.

The prompt used for GPT-4 to measure semantic rationality is:

You are a professional linguist, analyse the following sentences in terms of their semantic fluency and rationality and give them a score between 0 and 10.

For this part of the experiment, we utilized ChatGLM2-6B/ChatGLM2-6B-int4 to generate text. AC(Ziegler et al., 2019) was employed to generate stegos, while the models were allowed to perform random sampling to generate covers. During generation, we set the top-p truncation to 0.8 and the temperature to 0.8, following the generation configurations used by ChatGLM(Du et al., 2022). The results are presented in Tab.1.

Our framework generates stegos with a lower PPL than AC/RS. This is due to the CA checking the fluency of the stegos and providing prompts for the GA to regenerate. The feedback from CoT significantly improves the quality of the stegos.

We also tested the embedding rate (ER) and the 'mission success rate' (MSR), which indicates the probability of generated texts meeting the requirements in prompts. Further details can be found in Tab.2.

Since the prompt is very restrictive on the model output, the entropy of the model predicted distribution is relatively low. This leads to the extreme phenomenon that in some generated sentences, which are mostly longer than 5 tokens, AC is not able to embed a single bit. However, such sentences are ubiquitous in application scenarios. This re-

Attacks		Random				Paraphrase	SC		
		Insert	Delete	Replace	Swap		SNR=5	SNR=15	SNR=60
AC-6B	$\lvert T\rvert = 1$	0.034	0.012	0.016	0.010	0	0	0	0.0021
	$\lvert T\rvert = 2$	0.049	0.025	0.044	0.037	0	0	0	0.0021
	$\lvert T\rvert = 3$	0.073	0.057	0.065	0.028	0	0	0	0.0021
	$\lvert T\rvert = 4$	0.070	0.056	0.061	0.021	0	0	0	0.0021
AC-6B-int4	$\lvert T\rvert = 1$	0.031	0.027	0.026	0.017	0	0	0	0.0051
	$\lvert T\rvert = 2$	0.052	0.032	0.059	0.047	0	0	0	0.0037
	$\lvert T\rvert = 3$	0.065	0.053	0.052	0.038	0	0	0	0.0003
	$\lvert T\rvert = 4$	0.072	0.033	0.057	0.052	0	0	0	0.0000
Ours-6B	$\lvert T\rvert = 1$	**0.933**	**0.837**	**0.840**	**0.848**	**0.4203**	**0.8364**	**0.8370**	**0.8446**
	$\lvert T\rvert = 2$	0.874	0.704	0.701	0.712	0.2869	0.7187	0.7819	0.7898
	$\lvert T\rvert = 3$	0.852	0.625	0.627	0.619	0.2340	0.6249	0.7243	0.7339
	$\lvert T\rvert = 4$	0.814	0.577	0.555	0.561	0.2111	0.5279	0.6683	0.6780
Ours-6B-int4	$\lvert T\rvert = 1$	**0.931**	**0.821**	**0.823**	**0.817**	**0.4175**	**0.7778**	**0.7836**	**0.8132**
	$\lvert T\rvert = 2$	0.869	0.700	0.681	0.702	0.2819	0.7601	0.7676	0.7764
	$\lvert T\rvert = 3$	0.832	0.608	0.607	0.605	0.2273	0.5943	0.7321	0.7334
	$\lvert T\rvert = 4$	0.774	0.545	0.542	0.513	0.1827	0.5330	0.6599	0.6862

Table 3: Decoding success rates of AC and ours, under attacks that ignore/preserve semantics. $|T|$ represents the length of **type**.

sult indicates that the entropy is compressed by LLMs and prompts with clear requirements. The redundant space on the symbolic space has become difficult to use.

With the help of feedback CoT, the MSR of ours is about 2 times of AC and RS. The average number of loops in feedback CoT is 0.6312. 58.15% of the sentences are allowed for output without regeneration and 29.25%/8.31%/4.22% of the sentences require 1/2/3 iterations. Since the MSR of RS is 0.463, the MSR of the simplest rejection sampling with n iterations can be estimated as

$$MSR_n = \sum_{i=1}^{n} 0.463 * (1 - 0.463)^{n-1} \tag{11}$$

$$= 1 - (1 - 0.463)^n \tag{12}$$

So to increase the MSR to 0.893, n is about 3.5945. Feedback CoT can reduce the number of iterations to 17% of the simplest rejection sampling.

3.2 Robustness Against Attacks that Ignore or Preserve Semantics

In this section, we first tested the robustness of our method and AC against attacks that ignore semantics. These attacks include random insert/delete/replace/swap tokens in a sentence. We tested the robustness of our method and AC against attacks that preserve semantics. These attacks include paraphrasing and semantic communication (SC)(Xie et al., 2021; Qin et al., 2023). Unlike randomly changing 1∼2 tokens, these attacks completely change all of the tokens, but preserve the meaning of the original sentence. Details of these attacks are given below.

Random Insert. Copies a random token from the phrase and inserts it at a random position in the phrase.

Random Delete. Deletes a random token from the block.

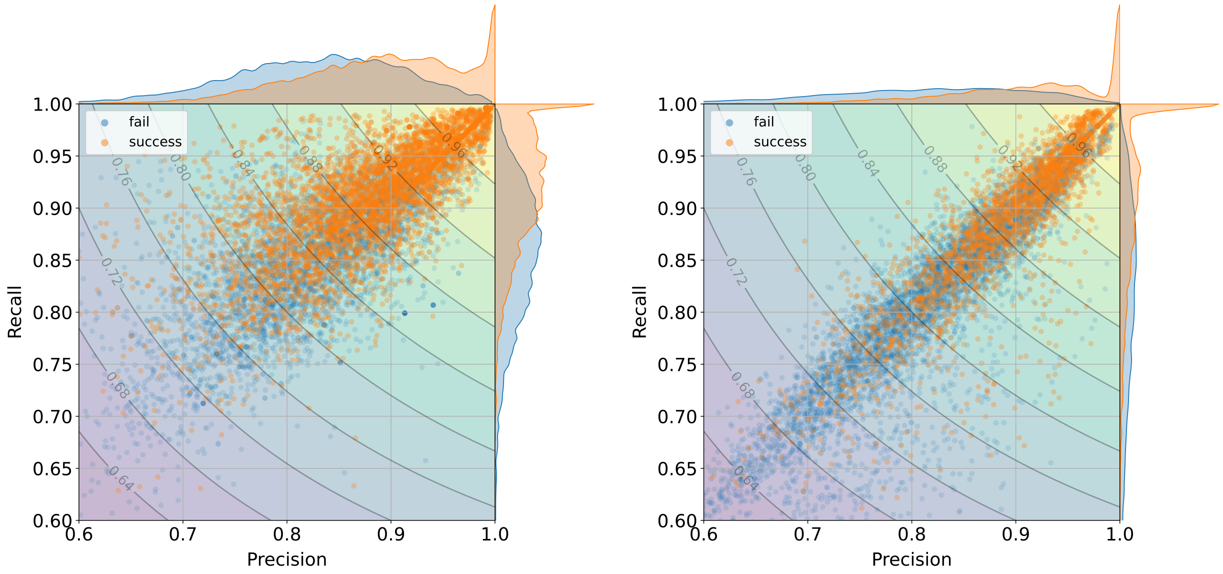
Random Replace. Copies a random token from the block and replaces it with another random token in the block.

Random Swap. Pick two random tokens from the phrase and swap them.

Paraphrase. The prompt we used for GPT-4 to paraphrase is: *You are an excellent editor. Rewrite the following sentences, keeping them about the same length and leaving the semantics as unchanged as possible.*

Semantic communication. Semantic communication methods(Xie et al., 2021; Qin et al., 2023) are designed to overcome the extremely high noise level. These methods have a probability of transmitting the correct meaning of sentences instead of the correct symbols.

Results are shown in Tab.3. The probability dis-



(a) Bertscore after paraphrasing

(b) Bertscore after semantic communication(SNR=5)

Figure 3: Bertscore(Zhang et al., 2020) of stegos attacked by paraphrasing and semantic communication. The contour line in the middle represents F1 score, and the estimated marginal distributions of the two samples are plotted on the top and right sides.

tribution predicted by LLM is changed so that AC's decoding is a disaster. In most cases, AC cannot decode the correct secret bits, and in most of these surviving examples, the attacks are targeted at the end of the sentence. Therefore, the prefix of the decoded bits is likely to be the same as the encoded bits, which will pass the test and be judged a success.

The situation with our stegos is completely different. Our stegos shows explicit robustness against these attacks. Since secret bits are embedded in entities, attacks that randomly change tokens have a relatively low probability of destroying these entities. In some cases, the tokens that denote entities are changed, but the LLMs are able to correctly extract entities from perturbed tokens. This part of the robustness depends on the ability of the LLMs to correct sentences.

Paraphrasing and SC completely change the sentences. As mentioned before, when the tokens are changed and the model prediction is different, AC is unable to decode. Our stegos retain some robustness against paraphrasing, and more than half can be decoded under SC.

However, paraphrasing and SC seem to subtly change the semantics. We measure the BertScore(Zhang et al., 2020) of our stegos and paraphrasing/SC stegos to clarify the semantic noise level.

As shown in Fig. 3, the samples that could be decoded correctly are concentrated in the high-F1 region. In the paraphrased samples whose Bertscore F1 is more than 0.8/0.9, the decoding success rate is more than 60%/75%. The statistics of semantic communication in the same situation is 85%/90%. The result shows that most of our stegos remain robust under attacks that preserve semantics well.

4 Conclusion

In this paper, we proposed a semantic steganography framework based on LLMs. We used ontology-entity tree to build the semantic space, leveraged Feedback CoT for rejection sampling, and applied AC for efficient encoding and decoding. Experiments showed that stegos generated by our framework are robust against attacks that ignore or preserve semantics. The embedding capacity of our framework is much higher than traditional symbolic steganography, while the quality of stegos generated by our framework is also better. The only component that need to be shared prior to covert communication is ontology-entity tree with the probabilities of its nodes. Since ontology-entity trees are available on the Internet and the methods of assigning probabilities are open, it is easy to apply our method to construct covert communications.

5 Limitations

Similar to symbolic steganography, our framework is difficult to operate in a low semantic entropy state. When the entities and relations in a sentence are fixed, there is no redundancy to embed bits. However, this condition is even more difficult for symbolic steganography since it takes into account the entropy at the token level. . The LLM used for this framework will affect the quality and robustness of the stegos. Therefore, we recommend using those large LLMs with open APIs. However, if the local use of LLMs is a necessity, the need for GPU resources becomes a limitation. In our experiments, we used ChatGLM-6B-int4, which requires a maximum of 6GB of GPU RAM. Calculated as the product of GPU memory and time in use, generating a sentence takes about 12.0754 GB·s.

6 Ethics Statement

We propose a steganography framework based on LLMs. Due to the convenience of accessing LLMs, this method may have an impact on the security of LLMs generated texts. In our future work, we will study the detection method against LLMs generated steganographic texts.

References

Rewon Child David Luan Dario Amodei Ilya Sutskever Alec Radford, Jeffrey Wu. 2019. Language models are unsupervised multitask learners.

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

Falcon Z. Dai and Zheng Cai. 2019. Towards near-imperceptible steganographic text.

Christian Schroeder de Witt, Samuel Sokota, J. Zico Kolter, Jakob Foerster, and Martin Strohmeier. 2023. Perfectly secure steganography using minimum entropy coupling.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

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

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335.

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

PaddleNLP. 2021. Paddlenlp: An easy-to-use and high performance nlp library. `https://github.com/PaddlePaddle/PaddleNLP`.

Zhijin Qin, Huiqiang Xie, and Xiaoming Tao. 2023. Mem-deepsc: A semantic communication system with memory. In *ICC 2023 - IEEE International Conference on Communications*, pages 3854–3859.

Jiaming Shen, Heng Ji, and Jiawei Han. 2020. Near-imperceptible neural linguistic steganography via self-adjusting arithmetic coding.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Yida Wang, Pei Ke, Yinhe Zheng, Kaili Huang, Yong Jiang, Xiaoyan Zhu, and Minlie Huang. 2020. A large-scale chinese short-text conversation dataset. In *Natural Language Processing and Chinese Computing*, pages 91–103, Cham. Springer International Publishing.

Huiqiang Xie, Zhijin Qin, Geoffrey Ye Li, and Biing-Hwang Juang. 2021. Deep learning enabled semantic communication systems. *IEEE Transactions on Signal Processing*, 69:2663–2675.

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

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

Siyu Zhang, Zhongliang Yang, Jinshuai Yang, and Yongfeng Huang. 2021. Linguistic steganography: From symbolic space to semantic space. *IEEE Signal Processing Letters*, 28:11–15.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert.

Zachary M. Ziegler, Yuntian Deng, and Alexander M. Rush. 2019. Neural linguistic steganography.

A Case Study

A.1 Generation & Regeneration

As Sec.2.4 presents, the generated stegos will be checked by CA and regenerate if necessary. With the entities “many years” and “Mr. Kee”, an simple example of regeneration is shown below:

Prompt: *Write a sentence containing the following keywords: many years, Mr Kee. In this case, many years is the element of time category, and no other time-like element should appear in the sentence. Mr Kee is an element in person category, no other person-like element should appear in the sentence.*

Generation v0: *Mr. Kee dedicated many years to teaching the children of the village.*

Feedback of CA: *“Children” should not be in the sentence because it is an element in person category.*

Regeneration v1: *Mr. Kee toiled for many years on his beloved farm.*

Feedback of CA: *“Farm” should not be in the sentence because it is an element in location category.*

Regeneration v2: *Mr. Kee fostered wisdom over many years.*

Feedback of CA: *Good. No errors.*

It is important to mention that the completeness of entities affects the rationality of stegos. Since we decide that some entities should be in the sentence and the others should not, the sentence might be somewhat strange if the chosen entities are not related.

A.2 Paraphrase & Semantic communication

We trained the model proposed by (Xie et al., 2021) in Chinese corpus. The impact of paraphrasing and semantic communication is presented below.

Entities: 演员,观众

Stego: 电影中的群众演员为影片增色不少，他们的不懈努力得到了观众的高度认可。

Paraphrased stego: 影片里的临时演员为电影增添了丰富的色彩，他们孜孜不倦的付出赢得了广大观众的赞誉。

Stego after semantic communication(SNR=5): 电影中的阮演员为诲真相产生了不少，他们的弹努力得到了观众的高度认可认可。

Stego after semantic communication(SNR=15): 中的群众演员为多个灵感色不少，他们的不懈努力得到了观众的高度认可。

Stego after semantic communication(SNR=60): 中的群众演员为多个冲突色不少，他们的不懈努力得到了观众的高度认可。

In this case, the entities 演员 and 观众 are not changed after paraphrase. Even those decoding methods based on retrieving are able to decode the correct bits.

Although the sentence attacked by semantic communication appears to make no sense, it still contains the correct entities and can be decoded appropriately.