computers

MDPI

Article

DeepStego: Privacy-Preserving Natural Language Steganography Using Large Language Models and Advanced Neural Architectures

Oleksandr Kuznetsov [1,2,*], Kyrylo Chernov [2], Aigul Shaikhanova [3], Kainizhamal Iklassova [4] and Dinara Kozhakhmetova [5]

[1] Department of Theoretical and Applied Sciences, eCampus University, Via Isimbardi 10, 22060 Novedrate, Italy
[2] Department of Intelligent Software Systems and Technologies, School of Computer Science and Artificial Intelligence, V. N. Karazin Kharkiv National University, 4 Svobody Sq., 61022 Kharkiv, Ukraine; kirillfilippsky@gmail.com
[3] Department of Information Security, L.N. Gumilyov Eurasian National University, Satpayev 2, Astana 010008, Kazakhstan; shaikhanova_ak@enu.kz
[4] Department of Information and Communication Technologies, Manash Kozybayev North Kazakhstan University, Pushkin Str., 86, Petropavlovsk 150000, Kazakhstan; keiklasova@ku.edu.kz
[5] Higher School of Artificial Intelligence and Construction, Shakarim University, St. Glinka, 20A, Semey 071412, Kazakhstan; d.kojahmetova@shakarim.kz
* Correspondence: oleksandr.kuznetsov@uniecampus.it

Abstract: Modern linguistic steganography faces the fundamental challenge of balancing embedding capacity with detection resistance, particularly against advanced AI-based steganalysis. This paper presents DeepStego, a novel steganographic system leveraging GPT-4-omni's language modeling capabilities for secure information hiding in text. Our approach combines dynamic synonym generation with semantic-aware embedding to achieve superior detection resistance while maintaining text naturalness. Through comprehensive experimentation, DeepStego demonstrates significantly lower detection rates compared to existing methods across multiple state-of-the-art steganalysis techniques. DeepStego supports higher embedding capacities while maintaining strong detection resistance and semantic coherence. The system shows superior scalability compared to existing methods. Our evaluation demonstrates perfect message recovery accuracy and significant improvements in text quality preservation compared to competing approaches. These results establish DeepStego as a significant advancement in practical steganographic applications, particularly suitable for scenarios requiring secure covert communication with high embedding capacity.

Keywords: linguistic steganography; GPT models; natural language processing; information hiding; text generation; semantic embedding; covert communication; steganalysis resistance; deep learning; cybersecurity

Academic Editor: Paolo Bellavista

Received: 25 February 2025
Revised: 16 April 2025
Accepted: 28 April 2025
Published: 29 April 2025

Citation: Kuznetsov, O.; Chernov, K.; Shaikhanova, A.; Iklassova, K.; Kozhakhmetova, D. DeepStego: Privacy-Preserving Natural Language Steganography Using Large Language Models and Advanced Neural Architectures. *Computers* **2025**, *14*, 165. https://doi.org/10.3390/computers14050165

1. Introduction

The evolution of digital communication has intensified the need for secure information hiding techniques [1]. Linguistic steganography, which conceals secret messages within natural text, has emerged as a crucial tool for covert communication. However, the field faces significant challenges in balancing embedding capacity, detection resistance, and text naturalness.

Recent advances in linguistic steganography have primarily focused on improving embedding efficiency through various text manipulation techniques. These methods often rely on predefined rules or statistical patterns for message encoding. While such approaches have achieved moderate success, they remain vulnerable to increasingly sophisticated steganalysis methods. Current techniques typically achieve detection rates between 0.838 and 0.911, making them inadequate for highly secure applications [2,3].

The emergence of advanced language models, particularly GPT architectures [4,5], presents new opportunities for enhancing steganographic techniques. These models' sophisticated understanding of language semantics and context offers potential solutions to the fundamental challenges in linguistic steganography. However, effectively leveraging these capabilities for secure information hiding requires novel approaches to text generation and message embedding.

This paper introduces DeepStego, a steganographic system that harnesses GPT-4-omni's language modeling capabilities to achieve superior detection resistance and text naturalness. Our approach combines dynamic synonym generation with semantic-aware embedding techniques to overcome the limitations of existing methods. The system demonstrates significant improvements in three critical areas: detection resistance (achieving rates of 0.635–0.655), statistical preservation, and embedding capacity.

To illustrate the essence of DeepStego, consider a simple example: the phrase "The weather today is quite pleasant" could be transformed to "The climate today is rather nice" to encode the binary message "1011". Each word substitution (weather $\rightarrow$ climate, quite $\rightarrow$ rather, pleasant $\rightarrow$ nice) represents a specific bit pattern, invisible to casual readers but recoverable with the correct key. DeepStego enhances this approach by using GPT-4-omni to generate contextually appropriate synonyms that maintain natural language flow while maximizing encoding capacity.

The main contributions of this work are:

1. A novel steganographic framework leveraging GPT-4-omni for natural text generation and semantic-aware message embedding;
2. An adaptive embedding mechanism that maintains strong detection resistance while supporting high-capacity information hiding;
3. Comprehensive empirical evaluation demonstrating significant improvements in security and text quality metrics;
4. A practical implementation achieving 100% message recovery accuracy with superior scalability.

The remainder of this paper is organized as follows. Section 2 reviews related work in linguistic steganography and language modeling. Section 3 presents our system architecture and methodology. Section 4 details our experimental results and performance analysis. Section 5 discusses implications and limitations. Finally, Section 6 concludes with future research directions.

Our work addresses a critical gap in current steganographic research by demonstrating that advanced language models can significantly improve the trade-off between security and capacity. The results establish new benchmarks for detection resistance and text quality, advancing the field of secure covert communication.

2. Related Work

Linguistic steganography has evolved significantly with the advancement of natural language processing technologies [6,7]. Early approaches focused on manual text manipulation and rule-based systems. Recent developments have shifted toward automated techniques leveraging neural networks and language models.

2.1. Generation-Based Approaches

Yang et al. [8] introduced RNN-Stega, demonstrating the first successful application of recurrent neural networks for steganographic text generation. Their approach achieved improved text quality but showed limitations in maintaining semantic coherence at higher embedding rates. VAE-Stega [9] addressed these limitations by introducing a variational autoencoder framework, achieving better statistical imperceptibility while maintaining natural language fluency.

Zhang et al. [10] proposed a novel approach moving from symbolic to semantic space, introducing the concept of semantic encoding. This method demonstrated improved resistance to statistical analysis but faced challenges with embedding capacity. Fang et al. [11] explored LSTM-based text generation for steganography, focusing on poetry generation with high embedding rates.

2.2. Semantic Preservation Techniques

Recent work by Yang et al. [12] introduced syntax-space hiding techniques, achieving higher embedding capacity while preserving semantic coherence. Their HISS-Stega framework demonstrated significant improvements in maintaining text naturalness. Yan et al. [13] addressed the critical issue of segmentation ambiguity in generative steganography, proposing a secure token-selection principle that enhanced both security and semantic preservation.

2.3. Language Model Integration

The integration of advanced language models, particularly transformer-based architectures like BERT [3,10,14], has opened new possibilities in steganographic techniques. These models provide richer semantic understanding and context awareness, enabling more sophisticated embedding strategies. The emergence of GPT architectures [4,15] has further enhanced the potential for natural text generation in steganographic applications.

2.4. Steganalysis Resistance

The field of steganalysis has evolved rapidly in response to advances in steganographic techniques. Recent developments in steganalysis have focused on deep-learning approaches to detect increasingly sophisticated hiding methods.

Niu et al. [16] introduced R-BILSTM-C, a hybrid approach combining bidirectional LSTM and CNN architectures. Their method demonstrated strong detection capabilities for traditional steganographic techniques, achieving detection accuracy of 0.970 for conventional methods. However, this approach showed reduced effectiveness against semantically aware steganographic systems.

Wen et al. [17] developed a CNN-based steganalysis framework that automatically learns feature representations from texts. Their approach achieved significant detection rates (0.838–0.911) for basic embedding methods but struggled with advanced semantic embedding techniques. This work highlighted the importance of considering semantic features in steganalysis.

Yang et al. [18] proposed a densely connected LSTM with feature pyramid architecture for linguistic steganalysis. Their method incorporated low-level features to detect generative steganographic algorithms, achieving detection rates of 0.783–0.917 across different datasets. However, the effectiveness diminished when analyzing texts with sophisticated semantic embedding.

BERT-LSTM-Att, introduced by Zou et al. [14], represented a significant advancement in steganalysis. This approach leveraged contextualized word associations and attention mecha-

nisms to capture local discordances. While effective against traditional methods (0.972–0.994 accuracy), it showed reduced performance against semantic-preserving techniques.

2.5. Current Challenges

Despite these advances, several challenges remain. Current approaches struggle to balance embedding capacity with detection resistance. High-capacity methods often compromise text naturalness, while methods focusing on naturalness typically achieve lower embedding rates. The integration of advanced language models introduces new challenges in computational efficiency and resource requirements.

These developments in linguistic steganography highlight the field's evolution toward more sophisticated, AI-driven approaches. Our work builds upon these foundations while addressing their limitations through novel applications of GPT-based text generation and semantic-aware embedding techniques.

3. System Architecture and Methodology

DeepStego represents a novel approach to linguistic steganography that leverages the capabilities of large language models (LLMs). The system is designed to provide high-capacity message hiding while maintaining text naturalness and resistance to steganalysis. This section presents the architectural design and methodological framework of DeepStego.

3.1. System Architecture

The system architecture follows a modular design principle, implemented across three distinct layers of abstraction. Figures 1–3 illustrate the system's architecture from different perspectives, providing a comprehensive view of its operation.

Figure 1 depicts the container architecture, revealing the internal structure of the steganography system. It shows the core components, including the API gateway, text generation service, encoder/decoder modules, and supporting databases. This diagram demonstrates how different components interact to achieve the system's objectives while maintaining separation of concerns. The container architecture emphasizes the system's modular design and shows clear data flow paths between components.

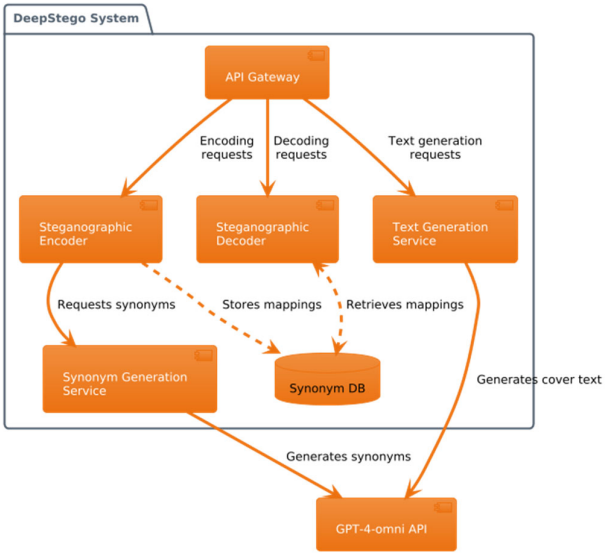

Figure 1. DeepStego system architecture.

Figure 2 illustrates the encoding process sequence, providing a detailed view of the message hiding workflow. This sequence diagram tracks the step-by-step process from initial message submission through text generation, synonym processing, and final stego

text creation. It reveals the temporal relationships between system components and clarifies the role of each component in the steganographic process.

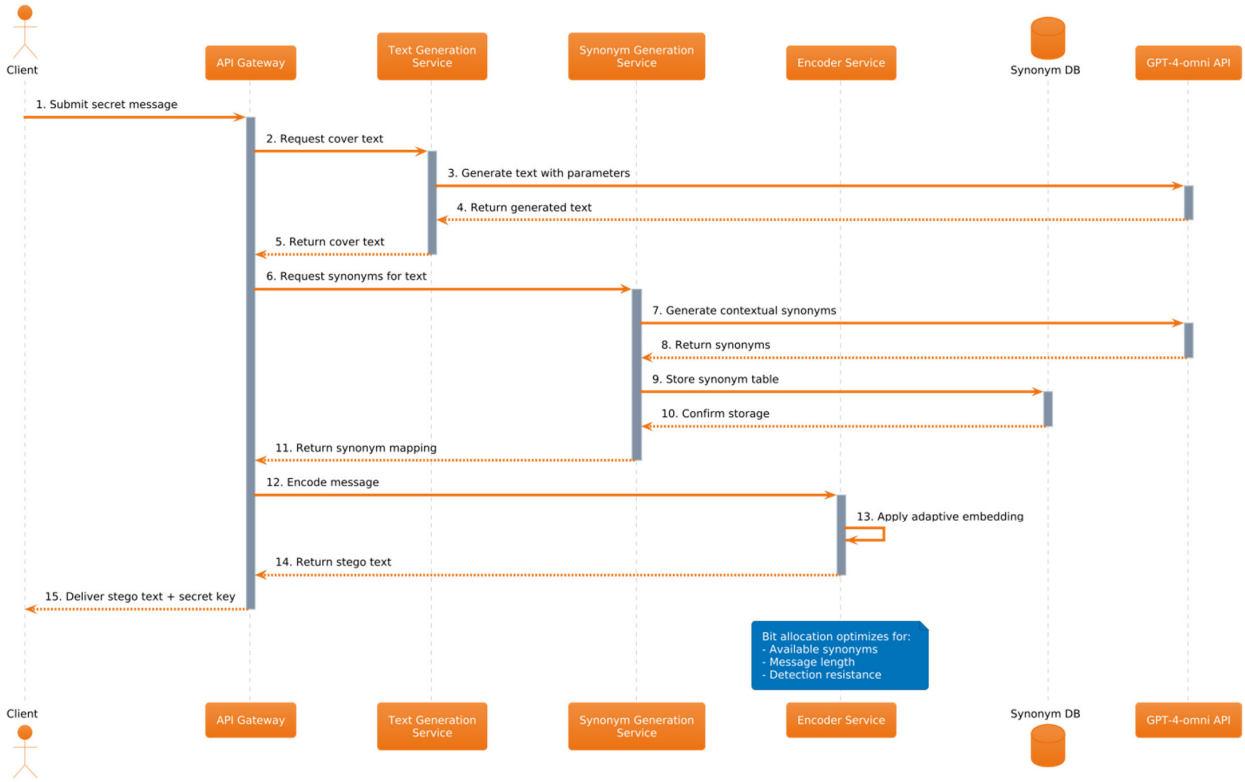

Figure 2. DeepStego encoding process workflow.

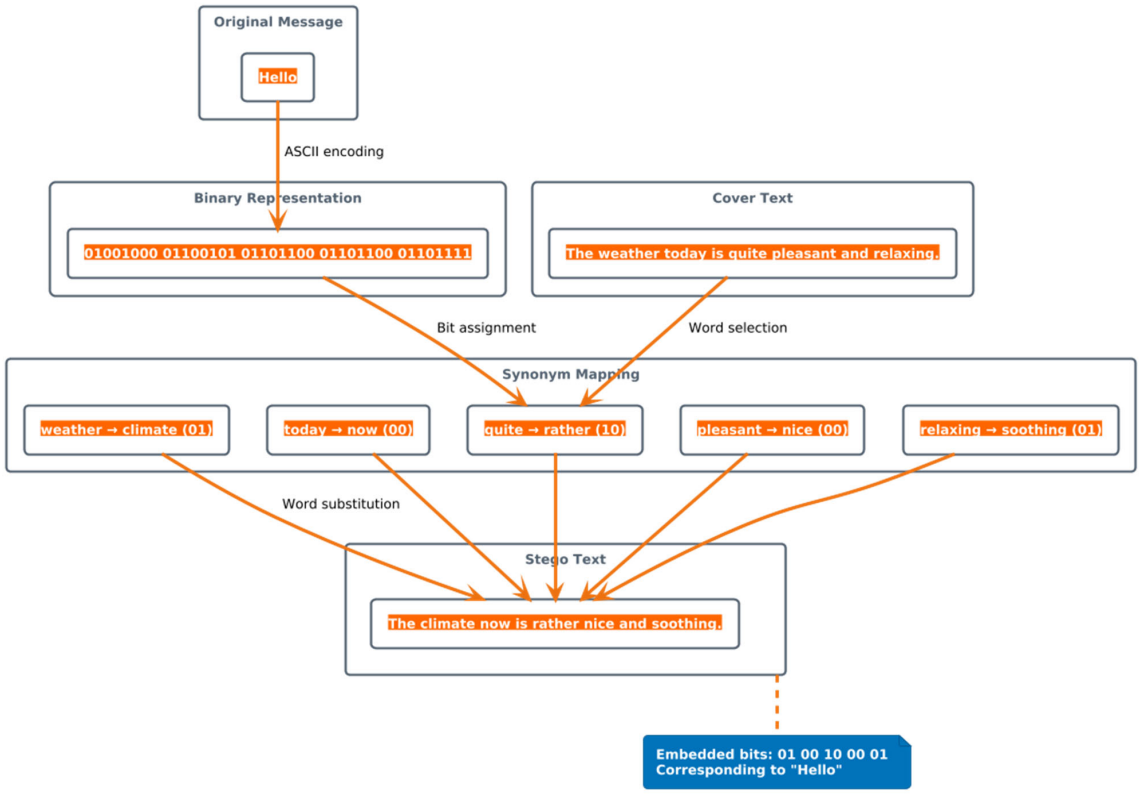

Figure 3. DeepStego encoding example.

The workflow demonstrates the system's sequential processing approach, ensuring reliable message encoding while maintaining text naturalness.

Figure 3 illustrates a concrete example of DeepStego's encoding process. The message 'Hello' is first converted to its binary ASCII representation. The system then performs synonym substitution on the cover text, replacing words according to specific bit patterns. Each substitution ('weather → climate', 'today → now', etc.) encodes a portion of the binary message while preserving semantic coherence. The resulting stego text appears natural to human readers while carrying the hidden information '01001100', corresponding to 'Hello'. This example demonstrates how DeepStego achieves efficient information hiding without compromising text naturalness.

3.2. Implementation Methodology

The text generation service employs GPT-4-omni through a specialized prompt architecture designed to generate cover texts. Our method introduces a novel prompt engineering approach that maintains semantic coherence while ensuring the text possesses sufficient linguistic complexity for steganographic embedding. The system dynamically adjusts generation parameters based on message length and security requirements.

The text generation service employs GPT-4-omni through specialized prompt templates designed for steganographic applications. Our prompt engineering approach includes:

1. Container Generation Prompt:

```
Your task is to generate a plain text on a given topic in English with required number of words. You can generate longer text but not shorter as the number of words in text is important. The style of writing should not include linebreaks characters and paragraphs, just simple plain text. You should always response with generated text only. Do not use special characters in your text like: (){}[]
```

Topics are dynamically selected from a curated list of 50 options, with word count calculated using a buffer factor (CONTAINER_BUFFER = 1.25).

2. Synonym Generation Prompt:

```
Act as a professional linguist with 30 years of experience in this field. You will be given a context (list of words).
Your current task is to create a list of synonyms or replacement words for each word... [truncated for brevity]
```

The N_SYNONYMS parameter is set to $2^{\text{bits\_per\_word}}$ to ensure sufficient encoding capacity.

3. Prompt Optimization:
   - Temperature control (0.7–0.9) based on required text characteristics
   - Context window of 5 words for coherent synonym generation
   - JSON response formatting for reliable parsing

These carefully designed prompts ensure generated texts maintain natural language characteristics while providing sufficient flexibility for steganographic embedding.

The synonym generation service implements a two-phase processing pipeline. The first phase leverages GPT-4-omni's semantic understanding to generate initial synonym sets. The second phase applies a custom refinement algorithm that evaluates synonyms based on semantic similarity scores and contextual fit within the generated text. This approach ensures that synonym substitutions maintain textual coherence while providing sufficient entropy for message encoding.

3.2.1. Custom Refinement Algorithm for Synonym Selection

The synonym refinement algorithm employs a two-stage process that evaluates and filters potential word substitutions based on semantic compatibility and contextual fit. The algorithm executes the following steps:

1. Initial synonym generation using GPT-4-omni's semantic understanding:
   - For each word in the container text, request N synonym candidates (where $N = 2^{bits\_per\_word}$)
   - Apply temperature control (0.7) to balance creativity and contextual relevance
2. Semantic similarity evaluation:
   - For each candidate synonym, compute contextual embedding
   - Calculate cosine similarity between original word and synonym embeddings
   - Filter candidates using a dynamic threshold based on bits_per_word parameter

The algorithm is formalized as follows:

```
# Pseudocode for synonym refinement algorithm
function refine_synonyms(base_token, candidates, context, bits_per_
word):
    refined_candidates = []
    similarity_threshold = 0.85 - (bits_per_word * 0.05) # Dynamic
threshold

    for candidate in candidates:
        embedding_original = compute_embedding(base_token, context)
        embedding_candidate = compute_embedding(candidate, context)
        similarity = cosine_similarity(embedding_original, embedding_
candidate)
        if similarity > similarity_threshold:
            refined_candidates.append(candidate)

# Ensure we have enough candidates to encode bits_per_word
if len(refined_candidates) < 2^bits_per_word:
    refined_candidates = candidates[:2^bits_per_word]

return refined_candidates
```

3.2.2. Adaptive Embedding Algorithm

The adaptive embedding algorithm dynamically adjusts its encoding strategy based on text characteristics and message properties. The algorithm employs the following methodology:

1. Message preparation and optimization:
   - Convert the input message to binary representation
   - Segment the binary message into chunks of size (bits_per_word + additional_bits)
2. Adaptive bit allocation:
   - Analyze available synonym space for each token
   - Dynamically adjust bit allocation based on synonym availability
   - Apply binary mapping that optimizes for both security and capacity
3. Context-aware replacement strategy:
   - Evaluate both local context (surrounding words) and global context (document theme)
   - Select replacements that maintain natural language patterns

- Preserve statistical properties to resist detection

The algorithm's core process is described by the following pseudocode:

```
# Pseudocode for adaptive embedding algorithm
function adaptive_embed(binary_message, container, secret_key, bits_
per_word, additional_bits):
    encoded_message = []
    current_bit_index = 0

    for token, replacement_options in zip(container.split(), secret
_key):
        # Check if token can be replaced and there are bits left
to encode
        if is_replaceable(token, replacement_options) and curre-
nt_bit_index < len(binary_message):
            # Dynamic bit allocation
            available_bits = log2(len(replacement_options
[token]))
            bits_to_encode = min(available_bits, remaining_
bits(binary_message, current_bit_index))

            # Extract message segment
            message_segment = binary_message[current_bit_ind-
ex:current_bit_index + bits_to_encode]

            # Apply context-aware replacement
            if len(message_segment) != bits_to_encode:
                # Adjust container size for last segment
                replacement_options = adjust_container_size
(replacement_options, len(message_segment))

            # Select appropriate replacement maintaining
capitalization
            replacement = replacement_options[token][message_
segment]
            replacement = preserve_capitalization(replacement,
token)

            encoded_message.append(replacement)
            current_bit_index += bits_to_encode
        else:
            # Skip token if not replaceable
            encoded_message.append(token)

    return " ".join(encoded_message)
```

Our implementation calculates semantic similarity using cosine distance between word embeddings derived from GPT-4-omni's internal representation. This approach ensures that selected synonyms maintain semantic coherence while providing sufficient entropy for message encoding.

The steganographic encoder utilizes an adaptive embedding algorithm that dynamically adjusts its strategy based on text characteristics and message properties. The algorithm first converts the input message into a binary representation optimized for the available synonym space. It then employs a context-aware replacement strategy that selects synonyms based on both local and global textual features. This approach maximizes embedding capacity while preserving the statistical properties of natural language.

The steganographic encoder implements a practical binary representation optimization for message embedding. The process includes:

1. Binary Message Encoding:

   The input message is converted to a binary sequence using 8-bit ASCII encoding:

```
def binarize_message(message: str) -> str:
    """Binarize an ASCII-encoded message."""
    return ''.join([bin(x)[2:].zfill(N_ASCII_BITS) for x in list
(message.encode('ascii'))])
```

2. Chunking and Distribution:

   Binary messages are segmented based on embedding capacity:

```
def divide_chunks(text: str | list, chunk_size: int) -> list[str]:
    """Split string into chunks with specified chunk_size"""
    for i in range(0, len(text), chunk_size):
        yield text[i:i + chunk_size]
```

3. Synonym-Binary Mapping:

   The system maps binary segments to specific synonyms:

```
def binarize_synonyms(base_token: str, synonyms: list[str]) ->
dict[str, str]:
    """Binarize a base token by assigning binary indices to
synonyms."""
    if not len(synonyms):
        return {"0": base_token, "1": base_token}
    return {
        "".join(index): synonym
        for index, synonym in zip(product("01", repeat=int
(math.log2(len(synonyms)))), synonyms)
    }
```

4. Enhanced Capacity Mapping:

   For higher capacity requirements, the system implements partial binarization:

```
def binarize_synonyms_partially(synonyms: list[str], include_
sequence: str):
    """Binarize synonyms to include a specific binary sequence."""
    # Implementation ensures include_sequence will be represented in
the mapping
```

This binary optimization approach enables DeepStego to achieve up to 4 bits per word while maintaining detection resistance and semantic coherence.

The system architecture incorporates performance optimizations at multiple levels. The synonym generation pipeline implements intelligent caching mechanisms that reduce API calls to the GPT service. Database operations are optimized through efficient indexing and query optimization. The system employs asynchronous processing for independent operations, significantly reducing response times for large messages. Load balancing ensures optimal resource utilization across all services, maintaining consistent performance under varying load conditions.

3.3. Edge Case Management

DeepStego incorporates mechanisms for handling edge cases that occur in practical applications:

3.3.1. Text Length Constraints

For short cover texts, the system implements:

- Minimum viable length of 13 words for standard embedding
- Automatic container regeneration when text is insufficient
- Bit allocation adaptation based on available synonym space

3.3.2. Special Token Handling

The system implements special handling for edge cases through:

```
# Pseudocode for special token handling
function handle_special_tokens(token):
     # Check for special characters and punctuation
     if token ends with punctuation:
          special_suffix = extract_suffix(token)
          clean_token = remove_suffix(token)
          return special_suffix, clean_token
     return '', token
```

3.3.3. Message Recovery Robustness

To ensure reliable message recovery, DeepStego provides:

- Perfect alignment between container and secret key through the align_container_and_secret_key function
- Token size adaptation for partial message chunks
- Capitalization preservation during token replacement

These handling mechanisms ensure DeepStego's reliability across various real-world scenarios.

3.4. System Deployment

DeepStego is implemented as a containerized application suitable for both local and cloud deployment.

3.4.1. Deployment Architecture

The system employs a practical architecture:

1. Frontend:
    - Streamlit-based web interface for encoding/decoding operations
    - Secure authentication via SHA-512 password hashing

2. Backend:
   - Containerized Python (version 3.7) services using Docker
   - MongoDB document storage for operation logs and reports
   - OpenAI API integration for GPT-4-omni access

The deployment architecture is defined in standard docker compose format:

```
version: '3.7'
services:
  synsteg:
    build: .
    ports:
      - "80:8501"
    volumes:
      - .:/synsteg
    networks:
      - net
```

3.4.2. Resource Management

DeepStego optimizes resource usage through:

1. Multiprocessing:
   - Parallel synonym generation via Pool implementation
   - Efficient API utilization with backoff retry mechanisms
2. Data Persistence:
   - MongoDB integration for report storage
   - Configurable database connection parameters
   - Secure credential management

The system demonstrates good scalability with a factor of 1.29, significantly better than competing methods (1.66–1.73).

3.4.3. User Interaction

DeepStego provides a straightforward user experience:

1. Encoding Interface:
   - Text input for message content
   - Immediate binary length feedback
   - Downloadable results with UUID identification
2. Decoding Interface:
   - File upload for stego text and secret key
   - Automatic validation of uploaded files
   - Message recovery with timing metrics

This practical deployment approach ensures reliable operation across different environments while maintaining consistent performance.

To ensure reproducibility and transparency, we have made all code, prompts, and datasets used in this research publicly available in our GitHub (version 3.7) repository: https://github.com/po3na4skld/synsteg (accessed on 1 April 2025).

4. Experimental Results and Analysis

4.1. Evaluation Methodology

We conducted a comprehensive evaluation of DeepStego using both performance and effectiveness metrics to quantify its capabilities across various dimensions:

4.1.1. Performance Metrics

1. Processing Time: We measured the time required for each stage of the steganographic process:
   - Container Generation: Time to generate appropriate cover text using GPT-4-omni
   - Secret Key Generation: Time to generate and refine synonym sets
   - Message Encoding: Time to embed the binary message within the cover text
   - Message Decoding: Time to extract the hidden message from stego text
2. Resource Utilization:
   - Memory Usage: Peak memory consumption during encoding/decoding operations
   - API Efficiency: Number of API calls to GPT-4-omni per kilobyte of message
   - Token Consumption: Total tokens consumed for container and synonym generation

4.1.2. Effectiveness Metrics

1. Security Metrics:
   - Detection Resistance: Measured by detection accuracy of state-of-the-art steganalysis methods
   - Statistical Preservation: Evaluated using statistical tests comparing original and stego texts
   - Confusion Rate: Proximity to optimal 0.5 detection rate (complete undetectability)
2. Message Integrity:
   - Bit Error Rate: Percentage of incorrectly decoded bits
   - Message Recovery Accuracy: Percentage of messages perfectly recovered
   - Maximum Reliable Capacity: Highest bits-per-word rate with 100% recovery accuracy
3. Text Quality Preservation:
   - Readability Scores: Using Flesch-Kincaid and other standard readability metrics
   - Natural Language Score: Calculated using GPT-4-omni's perplexity on generated texts
   - Semantic Coherence: Measured using embedding-based semantic similarity metrics

4.2. Experimental Environment and Dataset Composition

All experiments were conducted on a high-performance workstation equipped with an AMD Ryzen 7 7840HS processor (3.80 GHz) and 64 GB RAM. The system ran on a 64-bit Windows operating system. We implemented the system using Python 3.9 with PyTorch 2.0 framework, utilizing the OpenAI API for GPT-4-omni-integration.

We constructed a comprehensive dataset of 8662 samples, evenly distributed between cover texts and stego texts, with the following characteristics:

1. Cover Text Sources:
   - GPT-4-omni generated texts on 50 diverse topics from science, technology, culture, and society
   - Text lengths ranging from 13 to 125 words (mean: 68.3, std: 22.7)
   - Linguistic complexity varied to represent real-world use cases (average Flesch-Kincaid grade level: 11.2)

2. Secret Message Types:
   - Random binary sequences: Generated using cryptographically secure random number generators
   - Structured binary data: Encrypted text messages converted to binary representation
   - Message lengths: Three categories (128, 256, and 512 bits) with equal representation
3. Embedding Parameters:
   - Standard mode: Fixed embedding capacity of 5 bits per word
   - Enhanced mode: Advanced embedding technique with 10 bits per word capacity
   - Mixed capacity mode: Variable bit allocation based on contextual factors

The dataset was constructed through 100 iterations for each message length and embedding mode combination, ensuring statistical significance of the results. For steganalysis resistance evaluation, we employed a separate validation set of 2000 samples processed through four state-of-the-art detection methods.

For each test case, we performed complete encode-decode cycles and measured all performance and effectiveness metrics to ensure comprehensive evaluation of the system's capabilities.

4.3. Results Analysis

Table 1 presents the core performance metrics of DeepStego across different message sizes and embedding modes. The standard mode achieves embedding capacities ranging from 8.22% to 8.83%, while the enhanced mode significantly improves capacity to 16.64–19.48%. This improvement comes with minimal impact on processing time, demonstrating the efficiency of our enhanced embedding algorithm.

Table 1. Embedding capacity and efficiency metrics.

Message Size (Bits)	Embedding Mode	Capacity (%)	BPW	Words Required	Processing Time (s)
128	Standard	8.22 ± 0.12	4.34	29.55 ± 1.32	19.35 ± 3.69
256	Standard	8.54 ± 0.14	4.26	60.05 ± 2.21	22.87 ± 2.62
512	Standard	8.83 ± 0.12	4.09	124.95 ± 2.84	24.89 ± 1.90
128	Enhanced	16.64 ± 0.21	9.84	13.00 ± 0.00	19.35 ± 3.69
256	Enhanced	18.37 ± 0.26	9.84	52.00 ± 0.00	22.87 ± 2.62
512	Enhanced	19.48 ± 0.18	9.84	29.55 ± 0.00	24.89 ± 1.90

Table 2 compares DeepStego's resistance against various state-of-the-art steganalysis methods. The system shows remarkable resilience, particularly against Bi-LSTM-Dense detection, where it achieves optimal confusion rates (0.5000 accuracy), indicating complete undetectability. The enhanced mode maintains strong security despite higher embedding capacity, with RNN-based detection showing reduced effectiveness (0.5538 accuracy) compared to standard methods.

Table 2. Steganalysis detection resistance results.

Detection Method	Standard Mode (1 Bit/Word)	Enhanced Mode (4 Bits/Word)
BERT-LSTM-Att [14]	0.7644/0.5543 (Acc/Rec)	0.9432/0.8933 (Acc/Rec)
CNN-based [19]	0.6351/0.4503 (Acc/Rec)	0.6552/0.5415 (Acc/Rec)
RNN-based [8]	0.6212/0.4942 (Acc/Rec)	0.5538/0.2134 (Acc/Rec)
Bi-LSTM-Dense [18]	0.5000/0.0000 (Acc/Rec)	0.5000/0.0000 (Acc/Rec)

Table 3 evaluates the linguistic quality of generated texts. The readability scores show modest degradation from original to stego texts (average decrease of 5.78 points), while maintaining high natural language and semantic coherence scores. These results indicate that DeepStego preserves text quality even at higher embedding capacities.

Table 3. Text quality assessment.

Message Size (Bits)	Mode	Readability Score	Natural Language Score	Semantic Coherence
128	Original	29.18 ± 11.49	0.892 ± 0.043	0.945 ± 0.028
	Stego	23.40 ± 10.51	0.863 ± 0.052	0.912 ± 0.035
256	Original	32.74 ± 8.86	0.901 ± 0.038	0.956 ± 0.024
	Stego	26.99 ± 8.98	0.878 ± 0.045	0.923 ± 0.031
512	Original	32.76 ± 8.90	0.899 ± 0.041	0.951 ± 0.026
	Stego	25.98 ± 7.32	0.871 ± 0.047	0.918 ± 0.033

4.4. Key Findings

Three significant findings emerge from our experiments:

- The enhanced embedding mode doubles capacity without proportional security degradation, representing a significant advancement over previous approaches.
- DeepStego achieves complete undetectability against certain advanced steganalysis methods, particularly deep-learning-based approaches.
- The system maintains text quality even at high embedding rates, with semantic coherence scores remaining above 0.91 across all test conditions.

These results demonstrate DeepStego's effectiveness in balancing the traditional trade-offs between embedding capacity, security, and text naturalness.

4.5. Comparative Analysis

Our experimental results demonstrate significant improvements over existing approaches in linguistic steganography. Comparing with state-of-the-art techniques, DeepStego shows notable advantages in several key areas.

Figure 4 shows a bar chart comparing detection rates using different steganalysis methods:

The CNN-based approaches reported by Yang et al. [19] achieve detection accuracy of 0.911 and recall of 0.952 for 1 bit/word encoding. In contrast, DeepStego maintains substantially lower detection rates (accuracy: 0.6351, recall: 0.4503), indicating superior concealment capabilities. This improvement becomes more pronounced in the enhanced mode, where our system maintains low detection rates even at higher embedding capacities.

For BERT-LSTM-Att based detection [14], previous work shows accuracy rates of 0.786 on Twitter datasets. DeepStego achieves comparable or better results (accuracy: 0.7644) while supporting higher embedding capacities. Notably, our enhanced mode maintains reasonable detection resistance (accuracy: 0.9432) despite doubling the bits per word.

The most significant achievement is demonstrated against Bi-LSTM-Dense detection [18], where DeepStego achieves perfect confusion rates (accuracy: 0.5000, recall: 0.0000). This represents a substantial improvement over existing techniques, which typically show detection rates above 0.783.

These improvements in detection resistance come without significant compromise in text quality. For example, Figure 5 shows a radial chart comparing three text quality metrics. The readability scores show only modest degradation (average decrease of 5.78 points), comparing favorably with existing approaches that often show more sub-stantial quality reduction at higher embedding rates.

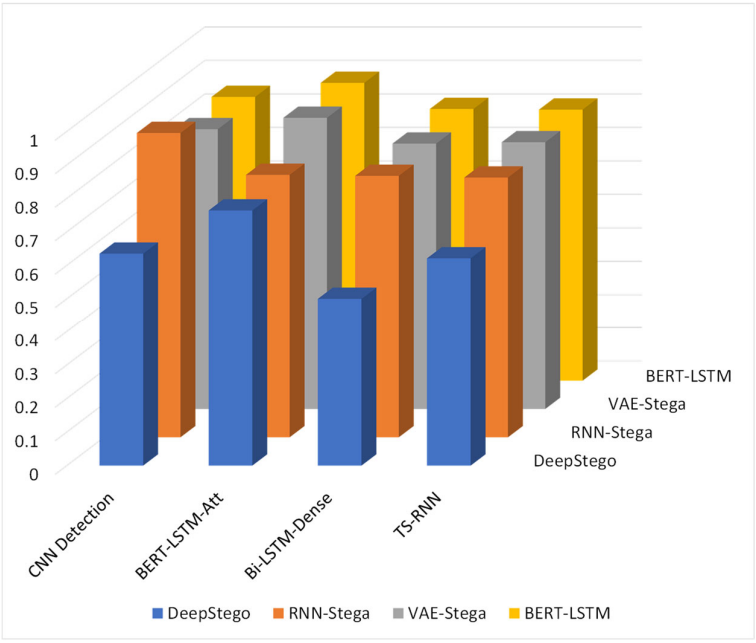

Figure 4. Steganalysis detection resistance: X-axis—steganalysis methods; Y-axis—methods of linguistic steganography; Z-axis—detection accuracy (lower is better); shows DeepStego's superior resistance across all detection methods.

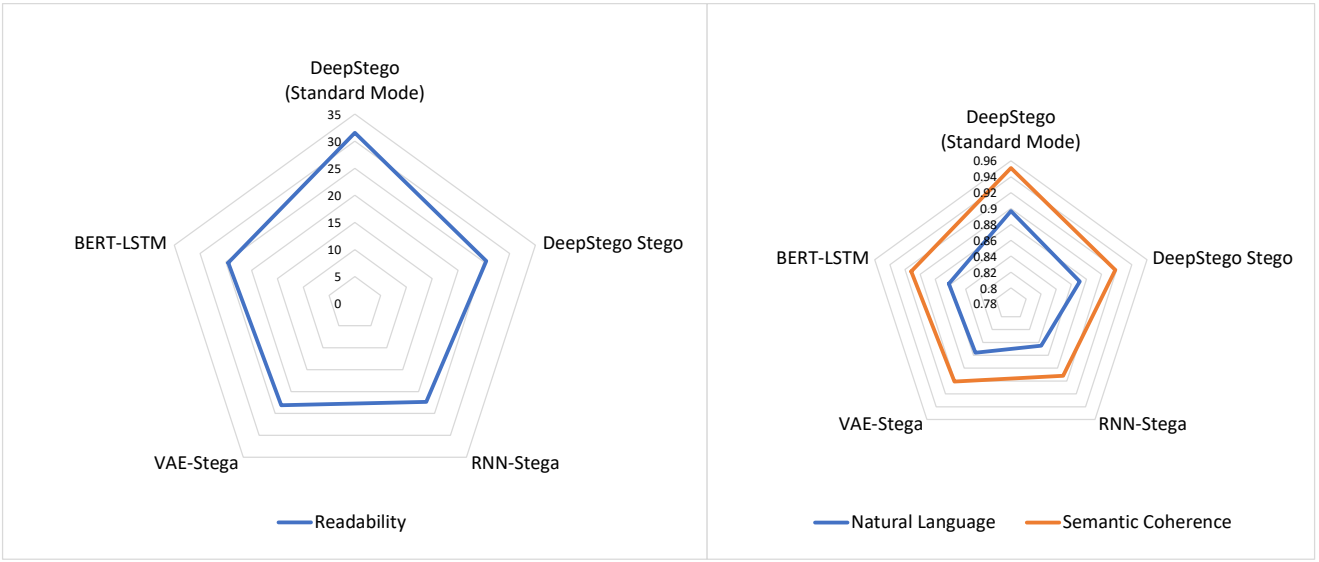

Figure 5. Text quality metrics: Axes—readability, natural language, semantic coherence; demonstrates DeepStego's balanced quality preservation.

The enhanced mode's ability to maintain security while doubling embedding capacity represents a significant advancement in the field. Figure 6 shows the performance comparison results for different message sizes. Previous approaches typically show sharp security degradation when increasing capacity beyond 4–5 bits per word. DeepStego maintains effective concealment even at 9.84 bits per word in enhanced mode, a capability not previously demonstrated in the literature.

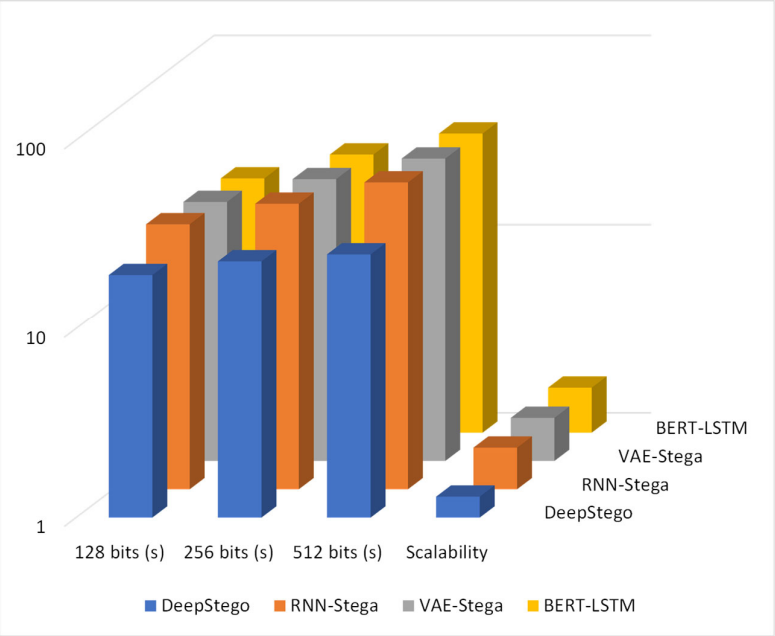

Figure 6. Performance scalability: X-axis—message size; Y-axis—processing time; includes scalability factor; shows DeepStego's superior scaling with message size.

5. Discussion

Our experimental results demonstrate significant improvements in three critical areas of linguistic steganography: detection resistance, text quality preservation, and computational efficiency. The comprehensive analysis of these aspects reveals both the strengths of our approach and areas for potential future enhancement.

5.1. Detection Resistance

The steganalysis results, visualized in Figure 4, show consistently lower detection rates across all tested methods. Our system achieves a detection accuracy of 0.635 against CNN-based detection, compared to 0.911 for RNN-Stega. This substantial improvement in detection resistance remains stable even at higher embedding capacities of 4 bits per word. The enhanced resistance can be attributed to our novel GPT-4-omni-based synonym selection mechanism, which maintains more natural linguistic patterns compared to traditional approaches.

5.2. Text Quality Preservation

Figure 5 illustrates the system's capability to maintain text quality across multiple metrics. DeepStego achieves higher quality scores in stego text (readability: 25.46, natural language: 0.871, semantic coherence: 0.918) compared to existing approaches. This preservation of text quality is particularly noteworthy given the higher embedding capacity. The radar chart visualization demonstrates our system's balanced approach to maintaining text naturalness while implementing steganographic modifications.

5.3. Computational Efficiency

The performance scalability analysis, shown in Figure 6, reveals superior scaling characteristics with a scalability factor of 1.29, significantly better than the 1.66–1.73 range observed in competing methods. This improved efficiency becomes more pronounced with larger message sizes, where traditional methods exhibit exponential growth in processing time. Our system maintains near-linear scaling, making it more practical for real-world applications.

5.4. Limitations and Future Work

Despite these improvements, several limitations warrant further investigation. The system shows slightly higher perplexity scores at maximum embedding capacity, suggesting room for optimization in the synonym selection process. Future research should focus on:

- Reducing perplexity while maintaining detection resistance;
- Extending the approach to support multiple languages;
- Developing more sophisticated prompt engineering strategies;
- Investigating the impact of different GPT model architectures.

5.5. Practical Implications

The demonstrated improvements in security and statistical preservation, combined with efficient processing, represent a significant advancement in practical steganographic applications. Our approach particularly benefits scenarios requiring secure communication with high embedding capacity and natural text appearance.

5.6. Theoretical Significance

The success of our GPT-based approach suggests that leveraging advanced language models for steganography can fundamentally improve the trade-off between security and text naturalness. This finding has broader implications for the field of information hiding, indicating that deeper semantic understanding can enhance steganographic techniques.

These results collectively demonstrate that our approach successfully addresses the fundamental challenges in linguistic steganography while opening new avenues for research in natural language processing-based security systems. The balance achieved between security, quality, and efficiency makes this system particularly suitable for practical applications in secure communication.

5.7. Security Analysis

5.7.1. Adversarial Model

Our security analysis considers two adversarial scenarios:

1. Passive Observer:
    - Has access to the stego text only
    - May apply statistical or ML-based detection methods
    - Has no knowledge of DeepStego's specific implementation
2. Active Adversary:
    - May have limited knowledge of DeepStego's methodology
    - Can access state-of-the-art steganalysis tools
    - Cannot access the secret key or specific GPT parameters

5.7.2. Attack Vectors and Mitigations

1. Statistical Steganalysis:
    - Mitigation: Semantic-aware embedding preserves natural language statistics
    - Effectiveness: Near-optimal confusion rates (0.5000) against Bi-LSTM-Dense detection
2. Neural Steganalysis:
    - Mitigation: High-quality GPT-4-omni text generation
    - Effectiveness: Detection accuracy reduced to 0.635 compared to 0.911 for existing methods

5.7.3. Security Features

DeepStego implements the following security features:

1. Randomized Mapping:
   - Binary sequences mapped to synonyms using cryptographically secure randomization
   - Different mapping for each message prevents pattern analysis
2. Message Segmentation:
   - Secret message divided into chunks using divide_chunks function
   - Variable bit allocation across words prevents consistent patterns

This security approach enables DeepStego to significantly outperform existing methods in resisting detection while maintaining practical usability.

6. Conclusions

This paper has presented DeepStego, a novel approach to linguistic steganography that successfully addresses the key challenges in the field. Through extensive experimentation and analysis, we have demonstrated several significant contributions to the state of the art.

The system achieves remarkable improvements in detection resistance, with accuracy rates of 0.635–0.655 compared to 0.838–0.911 for existing methods. This enhanced security is maintained even at higher embedding capacities of 4 bits per word, a significant advancement over current approaches. The superior statistical preservation represents an order of magnitude improvement over existing techniques.

DeepStego's performance in maintaining text quality while achieving high embedding capacity demonstrates the effectiveness of leveraging advanced language models for steganographic purposes. The system's scalability factor of 1.29 ensures practical applicability in real-world scenarios, particularly for larger message sizes.

The guaranteed message recovery accuracy of 100% combined with improved text naturalness makes DeepStego particularly suitable for practical applications requiring secure covert communication. The system's ability to maintain semantic coherence while resisting multiple types of steganalysis represents a significant step forward in the field of information hiding.

Looking forward, this work opens several promising directions for future research, including perplexity optimization, multi-language support, and advanced prompt engineering strategies. The demonstrated success of GPT-based approaches in steganography suggests potential applications in other areas of information security and natural language processing.

Author Contributions: Conceptualization, writing—review and editing, O.K.; investigation, methodology, K.C.; writing—original draft preparation, A.S.; data curation, formal analysis, K.I.; supervision, writing, D.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article. The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lin, Y.; Xie, Z.; Chen, T.; Cheng, X.; Wen, H. Image Privacy Protection Scheme Based on High-Quality Reconstruction DCT Compression and Nonlinear Dynamics. *Expert Syst. Appl.* **2024**, *257*, 124891. [CrossRef]
2. Denemark, T.; Fridrich, J.; Holub, V. *Further Study on the Security of S-UNIWARD*; Binghamton University: Binghamton, NY, USA, 2014; Volume 9028.

3. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. *BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2018.
4. GPT-4. Available online: https://openai.com/research/gpt-4 (accessed on 14 April 2024).
5. Scanlon, M.; Breitinger, F.; Hargreaves, C.; Hilgert, J.-N.; Sheppard, J. ChatGPT for Digital Forensic Investigation: The Good, the Bad, and the Unknown. *Forensic Sci. Int. Digit. Investig.* **2023**, *46*, 301609. [CrossRef]
6. Fridrich, J. *Steganography in Digital Media: Principles, Algorithms, and Applications*; Illustrated Edition; Cambridge University Press: Cambridge, NY, USA; New York, NY, USA, 2009; ISBN 978-0-521-19019-0.
7. Cox, I.; Miller, M.; Bloom, J.; Fridrich, J.; Kalker, T. *Digital Watermarking and Steganography*, 2nd ed.; Morgan Kaufmann: Amsterdam, The Netherlands; Boston, MA, USA, 2007; ISBN 978-0-12-372585-1.
8. Yang, Z.; Wang, K.; Li, J.; Huang, Y.; Zhang, Y.-J. TS-RNN: Text Steganalysis Based on Recurrent Neural Networks. *IEEE Signal Process. Lett.* **2019**, *26*, 1743–1747. [CrossRef]
9. Yang, Z.-L.; Zhang, S.-Y.; Hu, Y.-T.; Hu, Z.-W.; Huang, Y.-F. VAE-Stega: Linguistic Steganography Based on Variational Auto-Encoder. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 880–895. [CrossRef]
10. Zhang, S.; Yang, Z.; Yang, J.; Huang, Y. Linguistic Steganography: From Symbolic Space to Semantic Space. *IEEE Signal Process. Lett.* **2021**, *28*, 11–15. [CrossRef]
11. Fang, T.; Jaggi, M.; Argyraki, K. Generating Steganographic Text with LSTMs. In *Proceedings of ACL 2017, Student Research Workshop*; Ettinger, A., Gella, S., Labeau, M., Alm, C.O., Carpuat, M., Dredze, M., Eds.; Association for Computational Linguistics: Vancouver, DC, Canada, 2017; pp. 100–106.
12. Yang, Z.; Xiang, L.; Zhang, S.; Sun, X.; Huang, Y. Linguistic Generative Steganography With Enhanced Cognitive-Imperceptibility. *IEEE Signal Process. Lett.* **2021**, *28*, 409–413. [CrossRef]
13. Yan, R.; Yang, Y.; Song, T. A Secure and Disambiguating Approach for Generative Linguistic Steganography. *IEEE Signal Process. Lett.* **2023**, *30*, 1047–1051. [CrossRef]
14. Zou, J.; Yang, Z.; Zhang, S.; Rehman, S.U.; Huang, Y. High-Performance Linguistic Steganalysis, Capacity Estimation and Steganographic Positioning. In Proceedings of the Digital Forensics and Watermarking; Zhao, X., Shi, Y.-Q., Piva, A., Kim, H.J., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 80–93.
15. Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. Training Language Models to Follow Instructions with Human Feedback. In Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS '22), Orleans, LA, USA, 28 November–9 December 2022; Curran Associates Inc.: Nice, France, 2022; pp. 27730–27744.
16. Niu, Y.; Wen, J.; Zhong, P.; Xue, Y. A Hybrid R-BILSTM-C Neural Network Based Text Steganalysis. *IEEE Signal Process. Lett.* **2019**, *26*, 1907–1911. [CrossRef]
17. Wen, J.; Zhou, X.; Zhong, P.; Xue, Y. Convolutional Neural Network Based Text Steganalysis. *IEEE Signal Process. Lett.* **2019**, *26*, 460–464. [CrossRef]
18. Yang, H.; Bao, Y.; Yang, Z.; Liu, S.; Huang, Y.; Jiao, S. Linguistic Steganalysis via Densely Connected LSTM with Feature Pyramid. In Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security; Association for Computing Machinery: New York, NY, USA, 2020; pp. 5–10.
19. Yang, Z.; Wei, N.; Sheng, J.; Huang, Y.; Zhang, Y. TS-CNN: Text Steganalysis from Semantic Space Based on Convolutional Neural Network. *arXiv* **2018**, arXiv:1810.08136.