mathematics

Article

Novel Linguistic Steganography Based on Character-Level Text Generation

Lingyun Xiang [1,2,3], Shuanghui Yang [2], Yuhang Liu [2], Qian Li [4] and Chengzhang Zhu [5,*]

1 Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, Changsha University of Science and Technology, Changsha 410114, China; xiangly@csust.edu.cn
2 School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, China; agentccs15@gmail.com (S.Y.); lyh826811587@gmail.com (Y.L.)
3 Hunan Provincial Key Laboratory of Smart Roadway and Cooperative Vehicle-Infrastructure Systems, Changsha University of Science and Technology, Changsha 410114, China
4 Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia; qian.li-7@student.uts.edu.au
5 Academy of Military Sciences, Beijing 100091, China
* Correspondence: kevin.zhu.china@hotmail.com

Received: 31 July 2020; Accepted: 7 September 2020; Published: 11 September 2020

Abstract: With the development of natural language processing, linguistic steganography has become a research hotspot in the field of information security. However, most existing linguistic steganographic methods may suffer from the low embedding capacity problem. Therefore, this paper proposes a character-level linguistic steganographic method (CLLS) to embed the secret information into characters instead of words by employing a long short-term memory (LSTM) based language model. First, the proposed method utilizes the LSTM model and large-scale corpus to construct and train a character-level text generation model. Through training, the best evaluated model is obtained as the prediction model of generating stego text. Then, we use the secret information as the control information to select the right character from predictions of the trained character-level text generation model. Thus, the secret information is hidden in the generated text as the predicted characters having different prediction probability values can be encoded into different secret bit values. For the same secret information, the generated stego texts vary with the starting strings of the text generation model, so we design a selection strategy to find the highest quality stego text from a number of candidate stego texts as the final stego text by changing the starting strings. The experimental results demonstrate that compared with other similar methods, the proposed method has the fastest running speed and highest embedding capacity. Moreover, extensive experiments are conducted to verify the effect of the number of candidate stego texts on the quality of the final stego text. The experimental results show that the quality of the final stego text increases with the number of candidate stego texts increasing, but the growth rate of the quality will slow down.

Keywords: linguistic steganography; LSTM; automatic text generation; character-level language model

1. Introduction

Steganography is the art of hiding secret information within another public and innocuous medium, e.g., image [1,2], audio [3], video [4] or text [5,6], in an inconspicuous manner. It plays an important role in the field of information security to provide a safe and secure way for confidential data and communications [7–9]. Currently, as the text data that people use the most daily are a suitable carrier for steganography, linguistic steganography has drawn great attention in recent years. However, it is a challenging task because of the few redundant embedding spaces existing in the text context and the requirement of sophisticated natural language processing technologies.

Linguistic steganography embeds the secret message into the content of a text. Currently, it can be mainly divided into three categories: text modification-based [10], coverless [11] and text generation-based linguistic steganography [12]. Linguistic steganography based on text modification takes advantage of equivalent linguistic transformations to slightly modify the text content to hide the secret message while preserving the meaning of the original text. The linguistic transformations include syntactic transformations [10,13], synonym substitutions [14–17], misspelled word substitutions [18] and so on. This type of linguistic steganography has high imperceptibility, but limited embedding capacity, as the alternative transformations in a text are always very rare. Moreover, compared with the corresponding cover text, the stego text with hidden information still has some deviation and distortion in statistics and linguistics, and it is easy to discover the existence of the hidden information by using linguistic steganalysis technology [19–21].

In order to resist attacks from various steganalysis methods, researchers began to study coverless linguistic steganography. Coverless means there is no need to make any modification to the original cover carrier [22]. Coverless linguistic steganography directly generates or extracts secret information from the true and unmodified natural text. The earliest method [11] divided the secret information into independent keywords, then used the Chinese character mathematical expression to extract the positioning label of each keyword and, finally, combined the label and the keyword to retrieve the large-scale texts to obtain the confidential text, which is unchanged and can be employed to carry the secret information. This type of method mainly makes great efforts to design different labels for locating the keywords of secret information [23] and retrieve a large-scale text dataset [24–27] to gain one or more appropriate confidential texts. Although the various coverless linguistic steganographic methods can completely resist existing linguistic steganalysis attacks, their embedding capacities are extremely low. In the worst case, one confidential text can only successfully hide one keywords. Moreover, there is the problem of inefficiency, which can become prohibitive in practice.

Both former types of linguistic steganography have the problem of low embedding capacity, but text generation-based linguistic steganography solves this problem well. This type of method does not require an original text in advance. It always employs language models or natural language processing techniques to generate pseudo-natural language texts to carry secret information [28,29]. Since there are more locations available for accommodating secret information and there is no upper limit to the length of the generated text, such a method has great embedding capacity. Early text generation-based methods [30] lacked sufficient artificial intelligence to automatically generate arbitrary text with high quality. The resulting stego text was prone to errors that did not conform to grammar or common sense; its sentences were incomplete and caused inconsistency in contextual semantics, and its content was poorly readable. Specifically, it is difficult to ensure that the linguistic statistical features of the generated stego are consistent with the normal natural text, so they are easy to detect by steganalysis [31].

To improve the quality of the generated stego text to enhance the security of the secret information, researchers make great efforts to utilize promising automatic text generation technologies combined with information hiding operations [5,12,32,33]. One typical work used the Markov chain to calculate the occurrence of each word in the training set and obtain the probability of migration and, finally, used the probability of migration to encode words to achieve the purpose of embedding information in the process of text generation. With deep learning making significant progress in the field of natural language processing, researchers have introduced text generation models based on deep learning into the field of linguistic steganography. Fang et al. [34] proposed a steganographic method using an LSTM-based text generation model, which successfully solved the problem that the Markov model could not obtain the ideal language model. Yang et al. [35] proposed a similar linguistic steganography method based on the recurrent neural network. They designed two kinds of coding methods, fixed length and variable length, to encode each word in terms of the probability distribution of words for embedding information. These two coding methods make linguistic steganography more practical. Although the linguistic steganographic method based on deep learning can greatly improve the quality and embedding capacity of the stego text, regardless of whether the information is embedded or extracted, it requires more resources and runs slower.

In order to improve the running speed of linguistic steganography based on deep learning and increase the length of secret information that can be embedded in each word, we propose a linguistic steganography method based on character-level text generation. The method automatically generates stego text by using the LSTM-based character-level language model to predict the next character instead of the next word, embedding at least 1 bit of secret information in each character. Meanwhile, to ensure the quality of stego text, based on the same secret information, we first produce multiple stego text candidates by adjusting the start strings and then concatenate a well-designed selection strategy to find the best stego text with the highest quality. The experimental results show that the proposed method can generate stego text more quickly and has higher embedding capacity than similar methods. Moreover, we conduct experiments to analyse the effect caused by the number of candidate stego texts and find a proper number from the experimental results. Experimental results show that the quality of the stego text can be improved by performing the stego text selection strategy.

The rest of this paper is organized as follows. First, in Section 2, we briefly introduce the existing related work. Section 3 then introduces the framework of the proposed method and its main modules. Subsequently, Section 4 details the information hiding and extraction algorithm of the proposed method. Section 5 details the experimental results and analysis. Finally, the conclusion of this paper is given in Section 6.

2. Related Work

2.1. Long Short-Term Memory Network

Early text generation-based linguistic steganographic methods were unable to generate high-quality stego text due to the underdevelopment of text generation technology. In recent years, with the application of recurrent neural networks (RNNs) in text generation, the quality of the generated text has improved significantly.

Unlike other deep neural networks [36,37], RNN is a special artificial neural network, which has many hidden layers. A basic RNN can have only one hidden layer. RNN is well suited for sequence modelling problems. It contains a feed-back connection at each time step, so it can be extended in the time dimension and form a deep neural network, whose structure is shown in Figure 1.

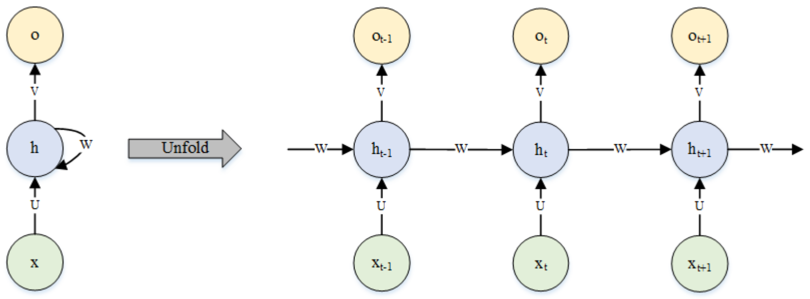


Figure 1. The structure of a simple RNN.

In each time step t, the RNN accepts the input vector x_t and the hidden state vector h_{t-1} to produce the next hidden state vector h_t by the following equations:

$$\begin{cases} h_t = f\left(W \cdot x_t + U \cdot h_{t-1} + b_h\right) \\ o_t = V \cdot h_t + b_o \end{cases} \tag{1}$$

where W, U, V denote the learned weight matrices, b_h and b_o denote the hidden bias vector and output bias vector and f is the hidden layer function, which is a nonlinear function and commonly is the tanh or softmax function.

In theory, RNN can process an input sequence of any length. However, in practice, due to the vanishing gradient problem, it cannot effectively deal with the long-range dependencies. However, the long short-term memory (LSTM) proposed by Hochreiter and Schmidhuber [38] is better than RNN at finding and exploiting long-range context by adding memory cell vector C_t. In time step t, an LSTM network accepts x_t, h_{t-1}, C_{t-1} as inputs and then produces h_t, C_t, which are calculated as the following composite function:

$$\begin{cases} I_t = \sigma\left(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i\right) \\ F_t = \sigma\left(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f\right) \\ C_t = F_t \cdot C_{t-1} + I_t \cdot tanh\left(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c\right) \\ O_t = \sigma\left(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o\right) \\ h_t = O_t \cdot tanh(C_t) \end{cases} \tag{2}$$

where I_t, F_t, O_t refer to the input gate, forget gate and output gate, respectively. C_t is the cell activation vector. These four vectors are the same size as the hidden vector h_t. At $t = 1$, h_0, C_0 are initialized to the zero vector. σ is the logistic sigmoid function. $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o$ are to-be-learned weight matrices. b_i, b_f, b_c, b_o are to-be-learned bias vectors.

In RNN, the short-term memory h will continue to multiply, and then, the gradient disappears. In LSTM, the accumulation is used instead of the multiplication, thus solving the gradient disappearance problem [39]. Currently, LSTM has surpassed RNN in many tasks, including language modelling [40].

2.2. Text Generation-Based Linguistic Steganography

Text generation-based linguistic steganography is based on linguistics, which directly generates the stego text without a pre-specified cover text according to the rule of linguistics and natural language processing technologies [41]. The core of this method is to generate stego text in the control of the secret information by using different automatic text generation technologies and making different choices in the process of text generation. The early text generation-based linguistic steganography mainly employed rule-based template generation technology to generate stego text; for example, Reference [41] adopted context-free grammar to construct sentences, while embedding secret information through selecting different context-free grammar rules and variables in the rules. The generated stego text has the correct sentence structure, but the semantics has nothing to do with the context. Subsequently, Reference [28] provided a more flexible approach to adjust and control the attributes of the generated text to improve its quality. It tries to extract available syntactic templates from a certain specific text or English sentences based on context-free grammar, but in fact, the generated text is not smooth enough and may also generate meaningless text.

With the dramatic improvements of natural language processing technology, text generation-based linguistic steganography can generate higher quality stego text driven by a large-scale corpus to train a good statistical language model and encode the conditional probability distribution of words for the purpose of hiding secret information in the generation process. A language model can be formalized as a probability distribution over a sequence of words or characters. Namely, given a string of past words, the language model provides an estimate of the probability that any given word from an pre-defined vocabulary will be the next word. The popular language model employed in text generation-based linguistic steganography is the Markov model [5,12,32,33,42,43].

Suppose an ordered word sequence $X = \{x_1, x_2, \ldots, x_t\}$; for the automatic text generation based on the first-order Markov chain, the word at the t-th position of the sequence can be associated with the conditional probability distribution based on the $t-1$-th word; thus, the word sequence X can be represented as the product of t estimated one-gram conditional probabilities, which can be formulated as follows:

$$\begin{aligned} P\left(x_1, x_2, \cdots, x_t\right) &= P\left(x_1\right) P\left(x_2 | x_1\right) \cdots P\left(x_t | x_1 x_2 \cdots x_{t-1}\right) \\ &= P\left(x_1\right) P\left(x_2 | x_1\right) \cdots P\left(x_t | x_{t-1}\right) \end{aligned} \tag{3}$$

By utilizing the given word sequence X, the next word x_{t+1} can be generated by selecting a word from the candidate words with a high conditional probability estimated by the Markov model. As the candidate words selected as the $t+1$-th word are associated with different probabilities, linguistic steganography designs some coding approaches to encode each candidate to a code, such that the special candidate word is selected to be generated to express the appointed secret information.

Taking advantages of the Markov model for text generation, some novel linguistic steganographic methods have been presented. Moreover, Reference [42] made efforts to simplify the estimation procedure of text generation. It is assumed that all transition probabilities from a given state to other state are equal. Reference [43] cooperated the Markov chain model with the DESalgorithm to enhance the security of the secret information and presented a fixed-length coding method to encode each candidate word. However, in the process of stego text generation, they ignored the transition probability of each word, leading to the generated stego text having poor quality. Similarly, Reference [32] proposed a steganographic method based on the Markov model, which focuses on how to ensure each generated sentence is embedded with a fixed number of secret bits, but the generated result was not satisfactory due to ignoring the difference of the transition probability. Reference [5] used the Markov chain model to generate particular ci-poetry learning from a given corpus to hide secret information. To overcome the quality degradation of the generated stego text caused by the fixed-length coding, Reference [12] combined the Markov model and Huffman coding to propose a method to automatically generate stego text. During the text generation, each time, a Huffman tree was constructed for the candidate words according to their conditional probabilities. The word whose code matched the secret bits was selected to be generated as the stego word.

Although Markov model-based linguistic steganographic methods have improved the quality and reliability of the generated text compared with the previous methods, there are still some problems; for example, the generated texts are not natural enough to avoid being discovered by steganalysis, due to the limitations of the Markov model. With recent advances in neural networks, language modelling based on neural networks has begun to show satisfactory performances [44]. As a result, some linguistic steganographic methods based on neural networks have emerged. Reference [34] firstly introduced the LSTM to learn the statistical language model of natural text and explored an LSTM-based steganography framework, which can generate texts with different genres and good quality by training different models. Reference [35] proposed a linguistic steganography based on the recurrent neural network. It used a full binary tree and a Huffman tree to dynamically encode the conditional probability distribution of each candidate word, so that the secret information was embedded into the selected word according to the codes of the candidate words. Meanwhile, some researchers paid attention to the generation of specific semantic texts. Luo and Huang [45] proposed a steganography method to produce Chinese classic poetry by using the encoder-decoder framework. Tong et al. [46] presented an RNN encoder-decoder model to generate Chinese pop music lyrics to hide secret information. The generated specific texts embedded with secret information were in a certain form, meeting the visual and pronunciation requirements.

Neural network-based linguistic steganography has significantly improved the quality of the generated stego texts. It is worth noting that the aforementioned methods all employ word-level language models to generate text. The word-level text generation methods usually require large vocabularies to store all the words in a large-scale corpus, so that the input and output are always extremely complex, and the model demands billions of parameters. The character-level models tried to overcome this issue [47], which can produce powerful words. The experimental results in [48] also showed that even context representation can be generated to capture the characteristics of the morphology and semantics. Since the number of characters in a language is small, the input and output of the character-level language model is simple, and the character-level language model has the advantage of modelling out-of-vocabulary words. Therefore, in this work, we propose novel linguistic steganography using an LSTM-based character-level language model (LSTM-CLM) to generate stego text, while improving the information hiding efficiency and embedding capacity.

3. Character-Level Linguistic Steganography

In this section, we introduce our proposed character-level linguistic steganography (CLLS) method, which combines LSTM-based character-level text generation and stego text selection to generate high-quality stego text. For the text generation task, the secret message serves as the supervision to guide the generation process using the LSTM-based character-level language model (LSTM-CLM). To infer the best stego text for a given secret message, we leverage different start strings to improve the diversity of the generated stego texts and let them vote to decide which stego text is the best one for the embedded information. Therefore, the key task for this problem is to estimate the quality and security of the generated stego texts, so as to select the best one. To address the problems in the task of linguistic steganography, our method naturally integrates the text generation approach and the text selection approach. Next, the framework of the proposed method will be described in detail.

3.1. Framework

The framework of the proposed CLLS is shown in Figure 2. CLLS consists of two processes: information hiding and information extraction. Information hiding process mainly contains two modules: the stego text generation based on LSTM-CLM and the stego text selection, which will be introduced in the next subsections. Given a secret message, the LSTM-based stego text generation module automatically generates a candidate stego text for each start string under the control of the secret message, using an LSTM-based character-level language model (LSTM-CLM); while the stego text selection module considers the quality of the stego text to select the best one as the final stego text from all candidate stego texts. Namely, we leverage the first module to generate a number of candidate stego texts and the second module to find the high-quality stego text for the given secret message. The information extraction process extracts the embedded secret message from the stego text.

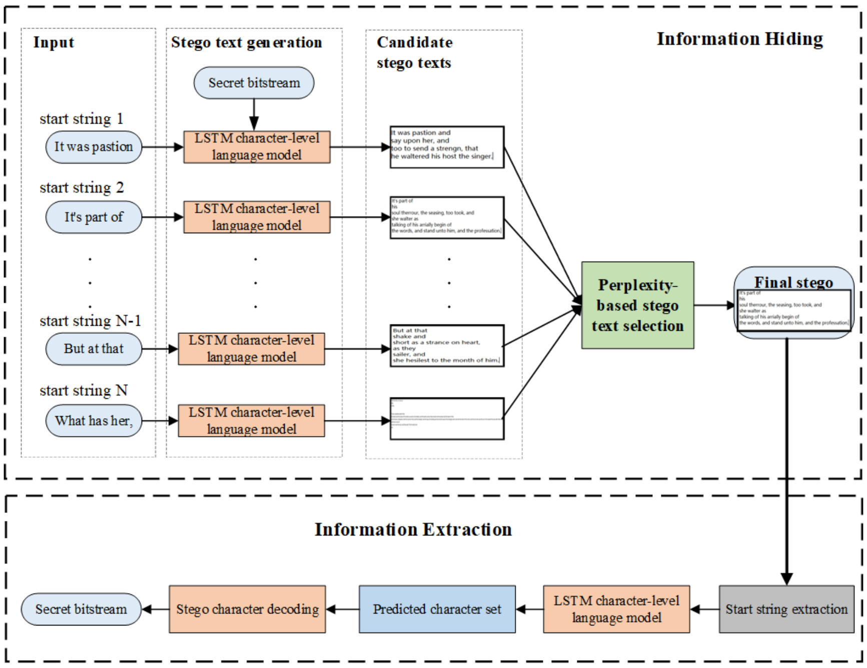


Figure 2. The framework of our proposed linguistic steganography.

The overall framework of CLLS is summarized below:

- Information hiding:

  (1) Generate randomly N start strings according to the training corpus as the input.
  (2) Train an LSTM-CLM and accept a start string as the input of the trained LSTM-CLM to generate a stego text. By coding the conditional probability distribution of candidate characters at each time of LSTM-CLM, select the character having the special code to match the given secret message to be generated, so as to generate N candidate stego texts by employing N different start strings for a given secret message.
  (3) Calculate the perplexity value of each candidate stego text to find the stego text with the highest quality and output it as the final stego text.

- Information extraction:

  (1) Extract the start string from the received stego text according to the shared parameters.
  (2) Input the extracted start string into the same trained LSTM-CLM as that in the information hiding process. The trained LSTM-CLM will produce the conditional probability distribution of all possible characters at each time. Adopt the same coding method;a predicted candidate character set is obtained, and each candidate character is encoded into a unique code.
  (3) By querying the code of the stego characters, decode the characters in the stego text into binary, so as to retrieve the embedded secret message.

3.2. Stego Text Generation Based on the LSTM-CLM

The key of the process of information hiding is to generate a stego text by using an LSTM-based character-level language model. The stego text we generate is a kind of sequence signal, while RNN is very suitable for sequence modelling. However, RNN cannot solve long-term dependence, which can lead to gradient disappearance and cause the parameters to not be updated. By adding a memory cell, the LSTM solves this problem successfully. Therefore, we finally use the LSTM to build the model for generating stego text.

The LSTM-CLM estimates the probability distribution over the sequence of characters by using the LSTM model. In the task of text generation, the LSTM-CLM can be formulated as a discrete character sequence prediction problem. During the prediction, the LSTM-CLM estimates the probability of the character x appearing at t time, $P(x)$ or $\prod P(x_t|x_{<t})$, which means the output character depends on the previous input characters. During the stego text generation process, we mainly use the ability of the LSTM and character language model in the modelling of the character sequence to complete the generation of stego text.

The architecture of stego text generation based on the LSTM-CLM is shown in Figure 3. The first step is to encode each input character in the start string into a $|G|$-dimensional vector space. $|G|$ is the size of the character table collected from a corpus. In this paper, the character table includes 26 letters in the English alphabet and punctuation marks. According to the frequency of each character appearing in the corpus, all characters in the character table are sorted, and then, according to their positions, each character is one-hot encoded as a $|G|$-dimensional vector, in which only one element is one and the rest are zero. Commonly, the i-th element of the i-th character in the ordered character table is set to one. Set the number of characters in the start string to l_s, then the start string can be encoded as a matrix $X \in R^{l_s \times |G|}$, and the i-th row in the matrix X represents the one-hot code of the i-th character in the start string.

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_{l_s} \end{pmatrix} = \begin{pmatrix} x_{1,1} & \cdots & x_{1,|G|} \\ \vdots & \ddots & \vdots \\ x_{l_s,1} & \cdots & x_{l_s,|G|} \end{pmatrix} \tag{4}$$

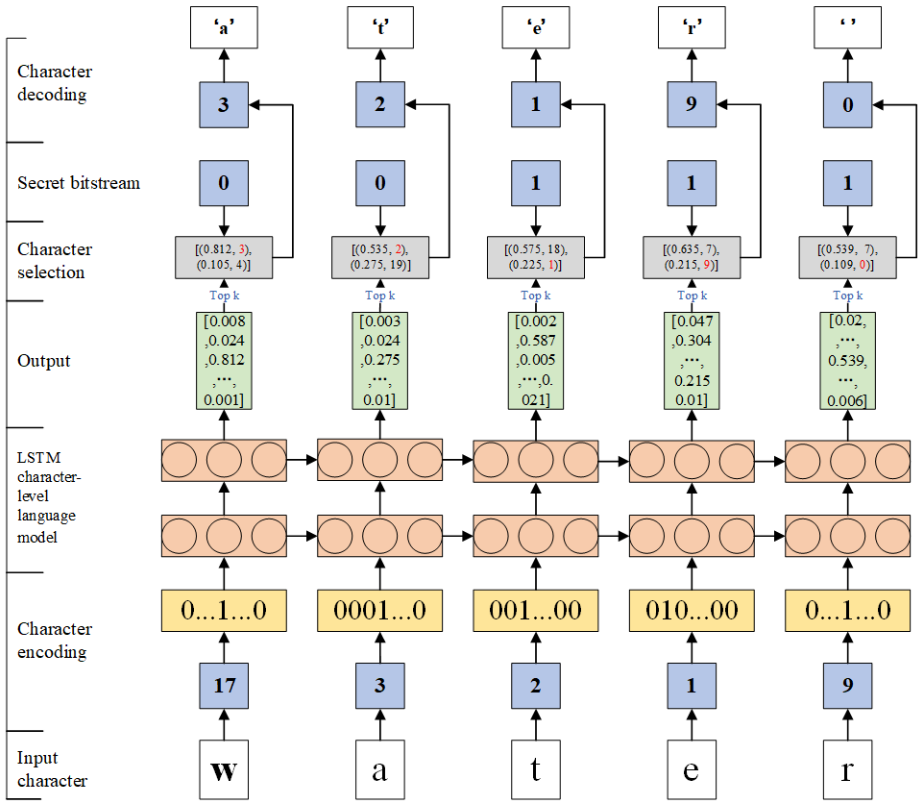


Figure 3. Architecture of the stego text generation based on the LSTM-character-level language model (CLM).

The LSTM-CLM receives one-hot encoded input characters X as the input vector and then estimates the next character probabilities by performing Equation (2). At time t, an output vector $O_t = [o_t^1, o_t^2, \cdots, o_t^{|G|}]$ is predicted, where the element o_t^j is the non-normalized probability of the j-th character in the character table. o_t^j indicates the possibility of the j-th character to be the $(t+1)$-th character of the generated text. o_t^j can be normalized by the following equation:

$$softmax\left(o_t^j\right) = \frac{exp(o_t^j)}{\sum_{i=1}^{|G|} exp(o_t^i)} \tag{5}$$

Generally speaking, there is actually more than one suitable character that can be selected to be the next character at each time, when they have high estimated probabilities. After sorting all the characters in the character table in descending order by their associated probabilities O_t, the first k characters with high probabilities are selected to construct a candidate character set. Since the probabilities of candidate characters are always high, an arbitrary candidate character can be selected to be generated, which does not have a great influence on the quality of the generated text. Therefore, we imperceptibly hide the secret message by controlling the selection of candidate characters to generate text.

When encoding characters in a candidate character set, we use a fixed length coding method whereby each character is encoded to a code with length s, where $k = 2^s$. As the characters in the set are ordered, the coding rule in this paper is to encode them in ascending order, namely using the binary number represented by the index of the corresponding character in the set as its code. For example, when $k = 2$, $s = 1$, suppose the two characters in the candidate set are a_1 and a_2, respectively, then a_1 and a_2 will be encoded as "0" and "1", respectively. When $k = 4$, $s = 2$, for a candidate set $\{a_1, a_2, a_3, a_4\}$, whose candidate characters have been sorted in descending order according to their estimated probabilities, the encoding results are shown in Table 1.

Table 1. The encoding result of the candidate character set.

Code	Index	Candidate Character
00	0	a_1
01	1	a_2
10	2	a_3
11	3	a_4

After all the candidate characters are encoded, a certain character, whose code is consistent with the current embedded secret bitstream, is selected as the current output of the generated stego text. For example, if the embedded secret bitstream is "01", then a_2 is selected as the next character. Each generated character can have s bits of secret information embedded. The current generated character will be one-hot encoded and added to the input matrix X to perform the training of the LSTM-CLM to generate the next character. When all the secret bitstream is embedded, it is possible that the last generated character is not a terminator of a complete sentence. In order to solve this problem, we will continue to generate characters with the highest probabilities until a terminator is encountered.

Taking $k = 2$, an example in Table 2 is given to describe the process of selecting candidate characters according to the embedded secret bitstream. The candidate characters are already ordered. Firstly, two candidate characters "a" and "o" are obtained by taking "w" as the input, as the current secret bitstream is "0", so candidate character "a" with a higher probability than "o" is selected. At present, the input of the LSTM-CLM should be updated to "wa", and then, new candidate characters "t" and "y" are obtained. According to the current secret bitstream, candidate character "t" is selected. Finally, the stego text "water" is generated and embedded secret bitstreams "00111".

Table 2. An example of embedding a secret bitstream into the generated character.

Input String	Candidate Characters	Possible Combination	Secret Bitstream	Stego Text
"w"	"a", "o"	"wa", "wo"	0	"wa"
"wa"	"t", "y"	"wat", "way"	0	"wat"
"wat"	"c", "e"	"wate", "watc"	1	"wate"
"wate"	"s", "r"	"water", "wates"	1	"water"
"water"	"s", " "	"waters", "water"	1	"water"

3.3. Stego Text Selection

As the stego text generation process is automatically controlled by the secret information, each character included in the stego text is not always selected as the one with the highest prediction probability, so the quality of the stego text will vary with the selected characters with different conditional probabilities. The quality of the stego text directly influences the imperceptibility and security of its carried secret information. In order to improve the imperceptibility of the stego text, we design a selection strategy to select a best stego text with a high quality from multiple candidate generated stego texts.

For the same secret information, we randomly generate N start strings and then use each start string to generate a candidate stego text. Different start strings will lead to a completely different stego text in a wide variety of quality. In order to select a high quality stego text from the candidates, we use perplexity to evaluate the quality of a generated stego text and select the one with the lowest perplexity value as the final stego text. The perplexity of a candidate stego text cst_j is calculated as follows:

$$\begin{aligned} perplexity(cst_j) &= 2^{-\frac{1}{n}\sum_{i=1}^{n}\log p(s_i)} \\ &= 2^{-\frac{1}{n}\sum_{i=1}^{n}\log p_i\left(w_1,w_2,\cdots,w_{n_i}\right)} \\ &= 2^{-\frac{1}{n}\sum_{i=1}^{n}\log p_i(w_1)p_i(w_2|w_1)\cdots p_i\left(w_n|w_1,w_2,\cdots,w_{n_i-1}\right)} \end{aligned} \tag{6}$$

where $s_i = \{w_1, w_2, \cdots, w_{n_i}\}$ represents the i-th generated sentence in the stego text cst_j, n_i denotes the number of words in the i-th generated sentence, $p(s_i)$ represents the probability distribution in the sentence s_i, $p(w_k)$ represents the probability distribution of the word w_k and n is the total number of sentences in cst_j. Although we generate stego text at the character level, we still calculate the perplexity by words to evaluate the quality of a generated stego text.

Setting the number of candidate stego texts as N, and denoting the candidate stego texts as $CST = \{cst_1, \cdots, cst_N\}$, the perplexity of the j-th candidate stego text is $perplexity\left(cst_j\right)$. Then the final stego text is selected by:

$$Stegotext(SM) = \underset{cst_j \in CST}{\arg\min}\, perplexity\left(cst_j\right) \tag{7}$$

where SM is the secret information and $Stegotext(\cdot)$ denotes the information hiding function of the proposed method.

4. Information Hiding and Extraction Algorithm

The proposed method includes two process: information hiding and information extraction. The algorithms of these two processes are elaborated in the subsequent subsections.

4.1. Information Hiding

The proposed character-level linguistic steganography must pre-define some parameters to generate the stego text. Furthermore, the parameters should be shared with the information hiding and information extraction algorithm to successfully extract the secret information. The information hiding algorithm of the proposed method is shown in Algorithm 1. Through this algorithm, a high-quality stego text can be generated with a fast running speed and large embedding capacity. Theoretically, s bits can be embedded into a character.

Algorithm 1 Information hiding algorithm.

```
Input:
    Secret message: $SM$
    The number of bits embedded in each character: $s$
    The number of previous characters to be employed for predicting the next character: $t$
    The number of candidate stego texts: $N$
    The number of words in a start string: $n_s$
Output:
    Stego text: $ST$
1: Data preprocessing and training an LSTM-based character-level language model (LSTM-CLM)
   using a large-scale corpus;
2: Generate randomly $N$ start strings including $n_s$ words; Denote the start string set as $SSL =
   \{ssl_1, ssl_2, \ldots, ssl_N\}$;
3: Denote the candidate stego text set as $CST = \{cst_1, \ldots, cst_N\}$, and initialize each text in $CST$ to
   the corresponding start string;
4: Set $q = 1$;
5: while $q \leq N$ do
6:     Select the $q$-th start string $ssl_q$ from $SSL$;
```

```
7:  Calculate the character number $l_s$ included in $ssl_q$;
8:  Convert $SM$ into a binary bit stream $SM_1$, and calculate its byte length $l_d$, which is converted
    into an 8 bit binary bit stream $LM$. Update $SM_1$ by $SM_1 = LM + SM_1$, which is the actual
    information embedded into the generated text. Here, the first 8 bits are employed to store the
    byte length of the secret information, which is useful for the information extraction algorithm to
    locate the characters' embedded information. Denote the bit length of $SM_1$ as $l_d$.
9:  set $i = l_s + 1, j = 1$;
10: while $j \leq l_d$ do
11:     Take the first $t$ characters $w_{i-t}, \ldots, w_{i-1}$ in $cst_q$ as the input of the LSTM-CLM, and then,
        the LSTM-CLM outputs the probability distribution of all characters;
12:     Sort all characters in the character table according to their predicted conditional probabilities
        in descending order, and select the first $2^s$ characters to construct the candidate character set;
13:     According to the coding rule, the character whose encoding value equals the value of the $j$-th
        bit to the $(j + s - 1)$-th bit of $SM_1$ is selected as the next character $w_i$;
14:     Attached $w_i$ to the $cst_q$;
15:     $i++$
16:     $j = j + s$
17: end while
18: while $w_i$ is not a terminator do
19:     Take the first $t$ characters in $cst_q$ as the input of the LSTM-CLM, and then, the LSTM-CLM
        outputs the probability distribution of all characters; select the character with the highest
        probability as the next character $w_i$.
20:     Attach $w_i$ to $cst_q$;
21:     $i++$
22: end while
23: Update $CST$;
24: $q++$;
25: end while
26: Calculate the perplexity value of a candidate stego text in $CST$, and select the text with the
    minimum perplexity as $ST$;
27: return $ST$.
```

4.2. Information Extraction

Information extraction is the recovery of embedded secret information from stego text, which is opposite of information hiding. The process of information embedding and information extraction is basically the same. They all need to use the same trained LSTM to estimate the conditional probability distribution of all characters at each moment, then construct the same set of candidate characters and use the same coding method to encode the characters.

After receiving the stego text, the receiver inputs the entire stego text into the LSTM-CLM. The LSTM-CLM selects the start string of the corresponding length and inputs it to obtain the probability distribution of all characters to predict the next characters. When the LSTM-CLM returns the probability distribution of all characters for the next character, the receiver needs to construct the candidate character set and obtain the codes of the candidate characters by employing the coding rule. Therefore, the character in the stego text can be decoded. The details of the information extraction algorithm are shown in Algorithm 2.

Algorithm 2 Information extraction algorithm.

```
Input:
    Stego text: $ST$
    The number of words in a start string: $n_s$
    The number of bits embedded in each character: $s$
    The number of previous characters to be employed for predicting the next character: $t$
Output:
    Secret message: $SM$
1: Load a trained LSTM-CLM, whose parameters are the same as those of the LSTM-CLM employed
   in the information hiding algorithm;
2: Select the first $n_s$ words of the stego text as the start string; and set its character length to $l_s$;
3: Calculate the character length of the stego text as $l_e$;
4: set $i = l_s + 1$;
5: while $i \leq l_e$ do
6:     Take the $t$ characters $w_{i-t}, \dots, w_{i-1}$ before the $i$-th character $w_i$ in the stego text as the input of
       the LSTM-CLM, and then, the LSTM-CLM outputs the character probability distribution of the
       next character;
7:     Sort the predicted probability of all characters employed in the LSTM-CLM in descending order
       and select the first $2^s$ characters to construct the candidate character set;
8:     According to the same coding rule used in the information hiding algorithm and the position of
       $w_i$ located in the candidate character set, decode the $s$ bit stream embedded in $w_i$ and attach it to
       the extracted bit stream string $SM_1$;
9:     if $(i - l_s) = \lceil \frac{8}{s} \rceil$ then
10:        calculate the decimal value $l_d$ of the first eight bits in the bit stream string $SM_1$, such that the
           length of the embedded secret message is $8 \times l_d$ bits, and then, update $l_e$ to $l_e = l_s + \frac{8 \times l_d}{s}$,
           namely only $\frac{8 \times l_d}{s}$ characters are employed to carry the secret message. Meanwhile, eliminate
           the first eight bits from $SM_1$;
11:    end if
12:    $i++$;
13: end while
14: Convert the bit stream string $SM_1$ to the character string, which is the embedded secret message
    $SM$;
15: return $SM$.
```

5. Experimental Results and Analysis

In this section, we present the experimental results and analysis to verify the performance of the proposed method.

5.1. Experimental Setup

As the LSTM-CLM requires a large-scale corpus to be trained so as to capture the statistical characteristics of natural texts, we selected the Gutenberg corpus for model training. The Gutenberg corpus comes with the NLTK library in Python. The details of the Gutenberg corpus are shown in Table 3:

Table 3. The details of the Gutenberg corpus.

Item	Value
Average Length of Sentence	26.6
Sentence Number	98,552
Word Number	2,621,613
Unique Character	71

In the experiments, we implemented our proposed method CLLS based on TensorFlow. While training the LSTM-CLM, we used a two layer LSTM network. Each layer contained 128 or 256 LSTM units. We employed the Adam optimizer to optimize the model and the cross-entropy loss function as the loss function. Meanwhile, the batch size was set as 32, the learning rate initialized as 0.01, the training epoch set as 12, and the dropout set as 0.5. We trained the LSTM-CLM on the NVIDIA GeForce GTX TITAN X.

5.2. Performance Analysis

In the experiments for generating stego text, we randomly selected a fragment from the natural text in the Gutenberg corpus as the secret information. The start strings were also randomly extracted from the Gutenberg corpus. The number of words in the start string was limited to 3–10. The number s of bits embedded in each character was set as one. The number t of previous characters to be employed for predicting the next character was set as 50. We conducted extensive experiments to generate huge stego texts for testing the performance of the proposed method.

(1) The efficiency of stego text generation:

We employed the average time required for generating stego text with a designated length to measure the efficiency of the proposed method. We fixed the size of the candidate character set to two, that is each stego character was embedded into 1 bit secret information. We calculated the average time of generating 1000 stego texts, each of which contained 50 words. At the same time, we compared with the other two similar linguistic steganographic methods [34,35], both of which are based on word-level text generation. Reference [34] used LSTM, and [35] used RNN to generate 50 words stego texts to observe the consumed time. The comparison results are shown in Table 4.

Table 4. The average time of generating the same number of words. CPS, candidate pool size; CSS, candidate character set size; CLLS, character-level linguistic steganographic method.

Methods	CPS/CCS	Time
Method in [34]	2	5.695 s
Method in [35]	2	3.25 s
CLLS-lstm128	2	0.642 s
CLLS-lstm256	2	0.822 s

In Table 4, CLLS-lstm128 and CLLS-lstm256 denote the proposed method whose size of LSTM units is 128 and 256, respectively. By adopting different numbers of LSTM units in the LSTM-CLM, we tried to verify the reliability of the proposed method. CPS is the size of the candidate pool in [34,35], and CCS is the size of the candidate character set in the proposed method. From Table 4, we can see that our proposed method generates stego text the fastest compared with the other two methods. This indicates that our proposed method can hide secret information more efficiently. Moreover, with the LSTM units' size increasing, the consumed time increases.

(2) Information hiding capacity:

Information hiding capacity is an important indicator to assess the performance of steganography. As the information hiding capacity always increases with the size of the stego text, we took the embedding rate to measure how much secret information can be embedded in a stego text. In this paper, the embedding rate is defined as the ratio of the number of bits actually embedded to the total number of bits of the entire generated stego text. The embedding rate can be calculated as follows:

$$ER = \frac{S}{L} \tag{8}$$

where S is the number of secret bits actually embedded and L is the bit length of the entire stego text. We selected a typical text modification-based linguistic steganography proposed in [10], two coverless linguistic steganography methods proposed in [11,27], and a text generation-based linguistic steganography proposed in [35] for comparison. The comparison results of the embedding rate are shown in Table 5:

From Table 5, we can see that the proposed method has a much higher embedding rate than the previous methods. The text modification-based and coverless linguistic steganographic have the disadvantage of very low embedding rates. In theory, the method in [35] can improve the embedding rate by enlarging the candidate pool, i.e., embedding more bits into each generated word. In the same way, CLLS can further raise the embedding rate by expanding the candidate character set to embedding more bits into each character. When the character in a secret message is encoded into seven bits, each generated character can be embedded into at least 1 bit; in this case, the ideal embedding rate should be 1/7 at least. However, during the stego text generation process, we randomly generated more characters after finishing the embedding of the secret bitstream, until the end of text appeared. There existed start string without carrying secret information. Therefore, the practical embedding rate in the case of embedding 1 bit into each character was lower than 1/7, as shown in Table 5.

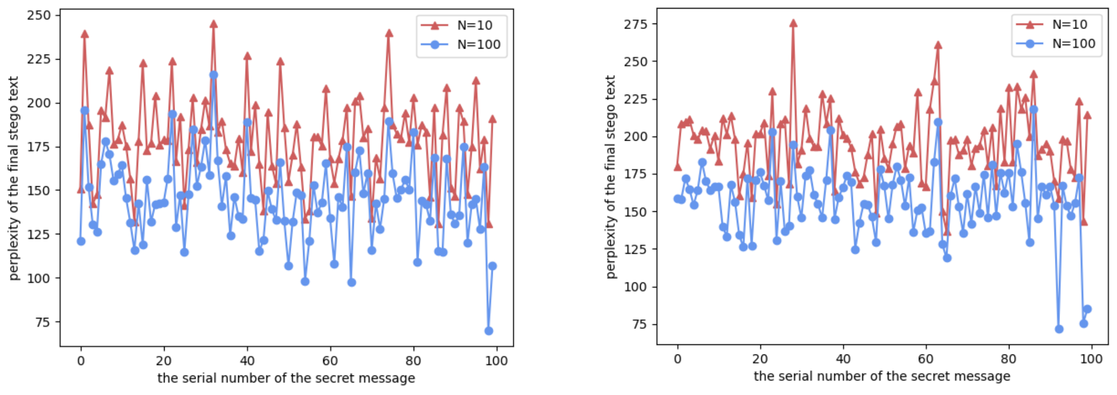
Table 5. The comparison of the embedding rate.

Methods	Embedding Rate (%)
Method in [10]	0.30
Method in [11]	1.0
Method in [27]	1.57
Method (3 bits/word) in [35]	7.34
CLLS-lstm128	12.56
CLLS-lstm256	12.59

(3) Imperceptibility:

The stego text selection can efficiently improve the quality of the generated stego text, thus enhancing the imperceptibility of the secret message. We performed experiments on 100 random secret message. For each secret message, N candidate stego texts were generated. The candidate stego text whose perplexity was minimum would select the final stego text. The perplexity values of the stego texts are shown in Figure 4. It can be found that the perplexities of most stego texts generated by CLLS-lstm256 were slightly greater than those of CLLS-lstm128. This may be due to the fact that the

LSTM-CLM in CLLS-lstm256 should be trained better with a larger corpus. Moreover, more candidate stego text can provide more chances to find a text with lower perplexity; thus, the perplexity of the stego text for $N = 100$ is much lower than that of the corresponding stego text for $N = 10$, as shown in Figure 4. The experimental results demonstrate the imperceptibility of the secret information to be significantly improved by the selection strategy of the proposed CLLS method.

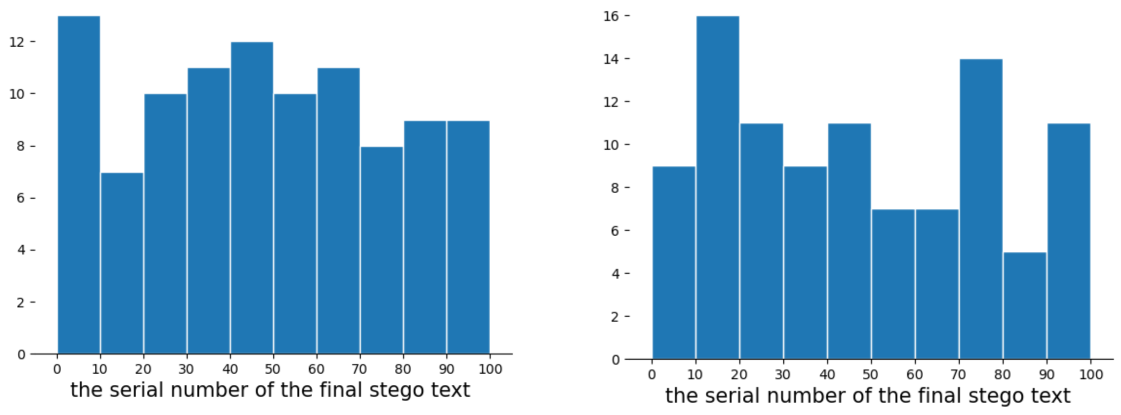


(**a**) CLLS-lstm128 (**b**) CLLS-lstm256

Figure 4. Comparison of the perplexity values of the stego texts.

5.3. Impact Analysis Caused by the Number of Candidate Stego Texts

The perplexity value of the final stego text will reduce with the number of the candidate stego texts increasing. However, the minimum perplexity value may reach a peak with very few variations for a larger N, which denotes the number of candidate stego texts. For $N = 100$, one-hundred candidate stego texts for a certain secret message are generated and numbered by using different start strings, and we select a text with the lowest perplexity value as the final stego text. For the 100 final stego texts corresponding to the 100 secret messages, the serial number of a final stego text is in the range of one to 100. After dividing the serial number range into 10 bins, then we map the serial numbers of final stego texts to the corresponding bins and count the number of times each serial number appears in the bins. Finally, a histogram is obtained as shown in Figure 5. From Figure 5, we can find that the serial number of the final stego text is nearly randomly distributed. The final stego text with a small serial number can be easily found by setting a small N. It is meaningless to generate more candidate stego texts with a large N. Although the large N can provide more chances for some secret messages to obtain the optimal stego text, the larger N is is not better, as the improvement is insignificant, consuming more time and resources.



(**a**) CLLS-lstm128 (**b**) CLLS-lstm256

Figure 5. The histogram of the serial number of the final stego text for $N = 100$.

In order to find an optimal N, we carried out experiments on CLLS-lstm128 and CLLS-lsmt256 with different N, respectively. We set $N = 10, 20, \cdots, 100$ to generate a certain number of candidate stego texts and then found the final stego texts. The average perplexity values of all final stego texts in terms of different N are shown in Figure 6. According to the results in Figure 6, it can clearly be seen that the average perplexity value decreases gradually with the increase of N. When $N \geq 60$, the average perplexity value decreases insignificantly and tends to be relatively stable for both CLLS-lstm128 and CLLS-lsmt256. Therefore, it is reasonable to choose $N = 60$ as the optimal N for the proposed method.

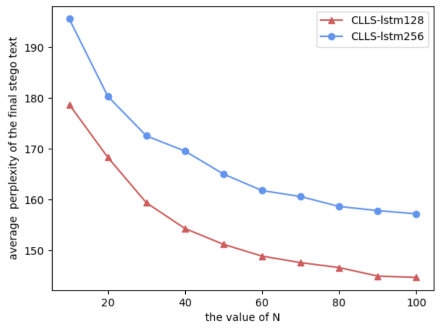


Figure 6. The average perplexity values for different N.

6. Conclusions

In this paper, we propose a linguistic steganographic method based on character-level text generation, which can automatically generate high quality stego text according to the secret message. The proposed method employs the LSTM-CLM to maintain long-term contexts to estimate the probability distribution of all characters to be the next character. The characters with the high probabilities are selected as candidates and encoded into different codes; thus, the proposed method generates the stego text by selecting different next characters to embed the secret message. Moreover, the proposed method coordinates the selection strategy to find the highest quality stego text from the candidates. We evaluate the proposed method's performance on the Gutenberg dataset. The experimental results show that the proposed method has a faster running speed and larger embedding capacity compared with some other linguistic steganographic methods. In future work, we would like to design and implement a better character-level language model supplementing word-level and subword-level information, thus improving the quality of the stego text. We are also interested in exploring other automatic text generation techniques that include text-to-text generation, meaning-to-text generation and image-to-text generation to generate more meaningful and natural stego text.

Author Contributions: Conceptualization, L.X. and Q.L.; methodology, L.X., S.Y. and C.Z.; software, S.Y.; validation, Q.L., Y.L. and C.Z.; formal analysis, S.Y. and Q.L.; investigation, S.Y. and Y.L.; resources, S.Y.; data curation, L.X. and S.Y.; writing, original draft preparation, L.X. and S.Y.; writing, review and editing, Q.L. and C.Z.; visualization, Y.L. and Q.L.; supervision, L.X.; project administration, C.Z.; funding acquisition, L.X. All authors read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China under Grants 61972057 and U1836208, the Hunan Provincial Natural Science Foundation of China under Grants 2019JJ50655 and 2020JJ4624, the Scientific Research Fund of Hunan Provincial Education Department of China under Grants 18B160 and 19A020, the Open Fund of Hunan Key Laboratory of Smart Roadway and Cooperative Vehicle Infrastructure Systems (Changsha University of Science and Technology) under Grant kfj180402 and the "Double First- class" International Cooperation and Development Scientific Research Project of Changsha University of Science and Technology (No. 2018IC25).

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

References

1. Wang, J.; Yang, C.; Wang, P.; Song, X.; Lu, J. Payload location for JPEG image steganography based on co-frequency sub-image filtering. *Int. J. Distrib. Sens. Netw.* **2020**, *16*, 1550147719899569. [CrossRef]

2. Qi, B.; Yang, C.; Tan, L.; Luo, X.; Liu, F. A novel haze image steganography method via cover-source switching. *J. Vis. Commun. Image Represent.* **2020**, *70*, 102814. [CrossRef]
3. Huang, Y.; Tang, S.; Yuan, J. Steganography in inactive frames of VoIP streams encoded by source codec. *IEEE Trans. Inf. Forensics Secur.* **2011**, *6*, 296–306. [CrossRef]
4. Sadek, M.M.; Khalifa, A.; Mostafa, M.G.M. Video steganography: A comprehensive review. *Multimed. Tools Appl.* **2015**, *74*, 7063–7094. [CrossRef]
5. Luo, Y.; Huang, Y.; Li, F.; Chang, C. Text steganography based on ci-poetry generation using markov chain model. *Ksii Trans. Internet Inf. Syst.* **2016**, *10*, 4568–4584.
6. Xiang, L.; Li, Y.; Hao, W.; Yang, P.; Shen, X. Reversible natural language watermarking using synonym substitution and arithmetic coding. *Comput. Mater. Contin.* **2018**, *55*, 541–559.
7. Yu, F.; Liu, L.; Shen, H.; Zhang, Z.; Huang, Y.; Shi, C.; Cai, S.; Wu, X.; Du, S.; Wan, Q. Dynamic analysis, circuit design, and synchronization of a novel 6d memristive four-wing hyperchaotic system with multiple coexisting attractors. *Complexity* **2020**, *2020*, 5904607.
8. Tan, Y.; Qin, J.; Xiang, X.; Ma, W.; Pan, W.; Xiong, N.N. A robust watermarking scheme in YCbCr color space based on channel coding. *IEEE Access* **2019**, *7*, 25026–25036. [CrossRef]
9. Chen, Y.; Tao, J.; Zhang, Q.; Yang, K.; Chen, X.; Xiong, J.; Xia, R.; Xie, J. Saliency detection via the improved hierarchical principal component analysis method. *Wirel. Commun. Mob. Comput.* **2020**, *2020*, 8822777.
10. Murphy, B.; Vogel, C. The syntax of concealment: Reliable methods for plain text information hiding. *Secur. Steganography Watermark. Multimed. Contents* **2007**, *6505*, 65050Y.
11. Chen, X.; Sun, H.; Tobe, Y.; Zhou, Z.; Sun, X. Coverless information hiding method based on the chinese mathematical expression. In *International Conference on Cloud Computing and Security*; Springer: Cham, Switzerland, 2015; pp. 133–143.
12. Yang, Z.; Jin, S.; Huang, Y.; Zhang, Y.; Li, H. Automatically generate steganographic text based on markov model and huffman coding. *arXiv* **2018**, arXiv:1811.04720.
13. Meral, H.M.; Sankur, B.; Ozsoy, A.S.; Gungor, T.; Sevinc, E. Natural language watermarking via morphosyntactic alterations. *Comput. Speech Lang.* **2009**, *23*, 107–125. [CrossRef]
14. Muhammad, H.Z.; Rahman, S.M.S.A.A.; Shakil, A. Synonym based Malay linguistic text steganography. In Proceedings of the Innovative Technologies in Intelligent Systems and Industrial Applications, CITISIA 2009, Monash, Malaysia, 25–26 July 2009.
15. Xiang, L.; Wu, W.; Li, X.; Yang, C. A linguistic steganography based on word indexing compression and candidate selection. *Multimed. Tools Appl.* **2018**, *77*, 28969–28989. [CrossRef]
16. Xiang, L.; Wang, X.; Yang, C.; Liu, P. A novel linguistic steganography based on synonym run-length encoding. *IEICE Trans. Inf. Syst.* **2017**, *100*, 313–322. [CrossRef]
17. Li, M.; Mu, K.; Zhong, P.; Wen, J.; Xue, Y. Generating steganographic image description by dynamic synonym substitution. *Signal Process.* **2019**, *164*, 193–201. [CrossRef]
18. Topkara, M.; Topkara, U.; Atallah, M.J. Information hiding through errors: A confusing approach. *Proc. Spie* **2007**, *6505*, 65050V.
19. Xiang, L.; Yu, J.; Yang, C.; Zeng, D.; Shen, X. A word-embedding-based steganalysis method for linguistic steganography via synonym substitution. *IEEE Access* **2018**, *6*, 64131–64141. [CrossRef]
20. Wen, J.; Zhou, X.; Zhong, P.; Xue, Y. Convolutional neural network based text steganalysis. *IEEE Signal Process. Lett.* **2019**, *26*, 460–464. [CrossRef]
21. Xiang, L.; Guo, G.; Yu, J.; Sheng, V.S.; Yang, P. A convolutional neural network-based linguistic steganalysis for synonym substitution steganography. *Math. Biosences Eng.* **2020**, *17*, 1041–1058. [CrossRef]
22. Luo, Y.; Qin, J.; Xiang, X.; Tan, Y.; Liu, Q.; Xiang, L. Coverless real-time image information hiding based on image block matching and dense convolutional network. *J. Real-Time Image Process.* **2020**, *17*, 125–135. [CrossRef]
23. Zhang, J.; Shen, J.; Wang, L.; Lin, H. Coverless text information hiding method based on the word rank map. *Int. Conf. Cloud Comput. Secur.* **2016**, *18*, 145–155.
24. Chen, X.; Chen, S.; Wu, Y. Coverless information hiding method based on the chinese character encoding. *J. Internet Technol.* **2017**, *18*, 313–320.
25. Zheng, N.; Zhang, F.; Chen, X.; Zhou, X. A novel coverless text information hiding method based on double-tags and twice-send. *Int. J. Comput. Sci. Eng.* **2020**, *21*, 116. [CrossRef]

26. Sun, H.; Grishman, R.; Wang, Y. Domain adaptation with active learning for named entity recognition. In *International Conference on Cloud Computing and Security*; Springer: Cham, Switzerland, 2016; pp. 611–622.
27. Zhou, Z.; Mu, Y.; Zhao, N.; Wu, Q.M.J.; Yang, C. Coverless information hiding method based on multi-keywords. In *International Conference on Cloud Computing and Security*; Springer: Cham, Switzerland, 2016; pp. 39–47.
28. Chapman, M.; Davida, G.I. Hiding the Hidden: A software system for concealing ciphertext as innocuous text. In *International Conference on Information and Communications Security*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 335–345.
29. Grosvald, M.; Orgun, C.O. Free from the cover text: A human-generated natural language approach to text-based steganography. *J. Inf. Hiding Multimed. Signal Process.* **2011**, *2*, 133–141.
30. Desoky, A. Nostega: A novel noiseless steganography paradigm. *J. Digit. Forensic Pract.* **2008**, *2*, 132–139. [CrossRef]
31. Yang, H.; Cao, X. Linguistic steganalysis based on meta features and immune mechanism. *Chin. J. Electron.* **2010**, *19*, 661–666.
32. Moraldo, H.H. An approach for text steganography based on markov chains. *arXiv* **2014**, arXiv:1409.0915.
33. Shniperov, A.N.; Nikitina, K.A. A text steganography method based on Markov chains. *Autom. Control Comput. Sci.* **2016**, *50*, 802–808. [CrossRef]
34. Fang, T.; Jaggi, M.; Argyraki, K. Generating steganographic text with LSTMs. *arXiv* **2017**, arXiv:1705.10742.
35. Yang, Z.; Guo, X.; Chen, Z.; Huang, Y.; Zhang, Y. RNN-Stega: Linguistic steganography based on recurrent neural networks. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 1280–1295. [CrossRef]
36. Lu, W.; Zhang, X.; Lu, H.; Li, F. Deep hierarchical encoding model for sentence semantic matching. *J. Vis. Commun. Image Represent.* **2020**, *71*, 102794. [CrossRef]
37. Wang, J.; Qin, J.H.; Xiang, X.Y.; Tan, Y.; Pan, N. CAPTCHA recognition based on deep convolutional neural network. *Math. Biosci. Eng.* **2019**, *16*, 5851–5861. [CrossRef] [PubMed]
38. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
39. Wang, B.; Kong, W.; Guan, H.; Xiong, N.N. Air quality forecasting based on gated recurrent long short term memory model in Internet of Things. *IEEE Access* **2019**, *7*, 69524–69534. [CrossRef]
40. Sundermeyer, M.; Schluter, R.; Ney, H. LSTM neural networks for language modelling. In Proceedings of the Thirteenth Annual Conference of the International Speech Communication Association, Portland, OR, USA, 9–13 September 2012; pp. 194–197.
41. Wayner, P. Mimic functions. *Cryptologia* **1992**, *16*, 193–214. [CrossRef]
42. Dai, W.; Yu, Y.; Deng, B. BinText steganography based on Markova state transferring probability. In Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, ICIS "09, Seoul, Korea, 24–26 November 2009; pp. 1306–1311.
43. Dai, W.; Yu, Y.; Dai, Y.; Deng, B. Text steganography system using markov chain source model and des algorithm. *J. Softw.* **2010**, *5*, 785–792. [CrossRef]
44. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. *Adv. Neural Inf. Process. Syst.* **2014**, 3104–3112.
45. Luo, Y.; Huang, Y. Text steganography with high embedding rate: Using recurrent neural networks to generate Chinese classic poetry. In Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec 2017, Philadelphia, PA, USA, 20–22 June 2017; pp. 99–104.
46. Tong, Y.; Liu, Y.L.; Wang, J.; Xin, G. Text steganography on RNN-Generated lyrics. *Math. Bioences Eng.* **2019**, *16*, 5451–5463. [CrossRef]
47. Sutskever, I.; Martens, J.; Hinton, G.E. Generating text with recurrent neural networks. In Proceedings of the 28th International Conference on International Conference on Machine Learning, Washington, DC, USA, 28 June–2 July 2011; pp. 1017–1024.
48. Marra, G.; Zugarini, A.; Melacci, S.; Maggini, M. An unsupervised character-aware neural approach to word and context representation learning. In *International Conference on Artificial Neural Networks*; Springer: Cham, Switzerland, 2018.