RNN-Stega: Linguistic Steganography Based on Recurrent Neural Networks

Zhong-Liang Yang, Xiao-Qing Guo, Zi-Ming Chen, Yong-Feng Huang, *Senior Member, IEEE*, and Yu-Jin Zhang

***Abstract*—Linguistic steganography based on text carrier auto-generation technology is a current topic with great promise and challenges. Limited by the text automatic generation technology or the corresponding text coding methods, the quality of the steganographic text generated by previous methods is inferior, which makes its imperceptibility unsatisfactory. In this paper, we propose a linguistic steganography based on recurrent neural networks, which can automatically generate high-quality text covers on the basis of a secret bitstream that needs to be hidden. We trained our model with a large number of artificially generated samples and obtained a good estimate of the statistical language model. In the text generation process, we propose fixed-length coding and variable-length coding to encode words based on their conditional probability distribution. We designed several experiments to test the proposed model from the perspectives of information hiding efficiency, information imperceptibility, and information hidden capacity. The experimental results show that the proposed model outperforms all the previous related methods and achieves the state-of-the-art performance.**

***Index Terms*—Linguistic steganography, recurrent neural network, automatic text generation.**

I. Introduction

IN THE monograph on information security [1], Shannon summarized three basic information security systems: encryption system, privacy system, and concealment system. Encryption system encodes information in a special way so that only authorized parties can decode it while unauthorized ones cannot. It ensures the security of information by making the message indecipherable. Privacy system is mainly to restrict access to information so that only authorized users have access to important information. Unauthorized users cannot access it by any means under any circumstances. However, while these two systems ensure information security, they also expose the existence and importance of information, making it more vulnerable to get attacks, such as interception and cracking [2], [3]. Concealment system is very much different from these two secrecy systems. It uses various carriers to embed confidential information and then transmit through public channels, hide the existence of confidential information to achieve the purpose of not being easily suspected and attacked [4]. Due to its extremely powerful information hiding capacity, concealment system plays an important role in protecting trade secrets, military security and even national defense security.

Manuscript received March 14, 2018; revised July 8, 2018 and September 3, 2018; accepted September 8, 2018. Date of publication September 24, 2018; date of current version January 30, 2019. This work was supported by the National Natural Science Foundation of China under Grant U1536201, Grant U1536207, and Grant U1636113. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Dinu Coltuc. *(Corresponding author: Yong-Feng Huang.)*

Z.-L. Yang, Y.-F. Huang, and Y.-J. Zhang are with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: yfhuang@tsinghua.edu.cn).

X.-Q. Guo is with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong.

Z.-M. Chen is with the International School, Beijing University of Posts and Telecommunications, Beijing 100876, China.

Digital Object Identifier 10.1109/TIFS.2018.2871746

Steganography is the key technology in a concealment system. Steganography shares many common features with the related but fundamentally quite different data-hiding technology called watermarking [5], [6]. Firstly, although both steganography and digital watermarking are used to conceal information in the carrier, the primary goal of steganography is to hide the existence of information. However, for digital watermarking, the primary goal is to resist modification. Secondly, we usually expect the amount of information embedded in the concealment system to be as large as possible. On the contrary, the amount of information embedded in the digital watermark is generally not large. Additionally, the information hidden in a concealment system is usually not regular, but the information embedded in the digital watermark system is usually well-designed. These very different requirements imposed on these two applications make their designs and analyses quite distinct.

A concealment system can be illustrated by Simmons' "Prisoners Problem" [4]: Alice and Bob are in jail, locked up in separate cells far apart from each other, and they wish to devise an escape plan but cannot be perceived by the warden Eve. Faced with this situation, Alice and Bob intend to hide their true information in the normal carrier. We can model this task in a mathematical way as follows. Alice and Bob need to transmit some secret message m in the secret message space $\mathcal{M}$. Alice gets a cover x from the cover space $\mathcal{C}$. Under the guidance of a certain key k_A in the keys spacespace$K$, the mapping function f is used to map x to s which is in the hidden space $\mathcal{S}$, that is:

$$Emb:\ \mathcal{C} \times \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{S}, \quad f(x, k_A, m) = s. \tag{1}$$

Bob uses the extraction function g to extract the correct secret message m from the hidden object s under the guidance of the key k_B in the keys space K:

$$Ext:\ \mathcal{S} \times \mathcal{K} \rightarrow \mathcal{M}, \quad g(s, k_B) = m. \tag{2}$$

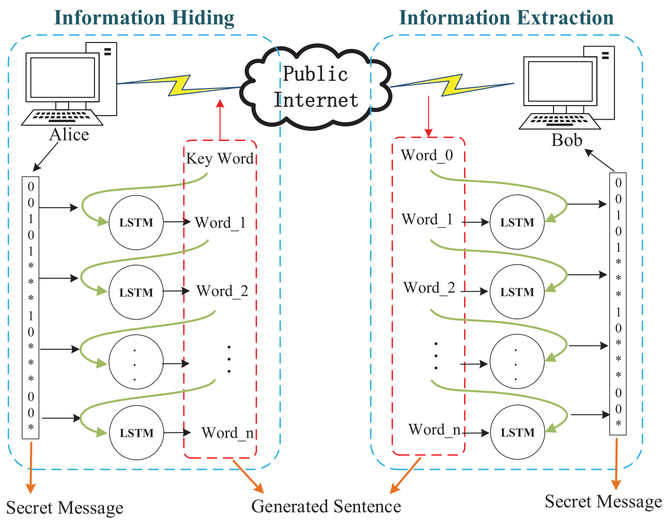
In order not to expose the existence of the hidden information, it is usually required that the elements in $\mathcal{S}$ and $\mathcal{C}$ are exactly the same, that is $\mathcal{S} = \mathcal{C}$. But generally speaking, this mapping function will affect the probability distributions, named $P_{\mathcal{C}}$ and $P_{\mathcal{S}}$. In order to prevent suspicion, we generally hope that the steganographic operation will not cause big differences in the distribution of carriers in the semantic space, that is:

$$d_f(P_{\mathcal{C}}, P_{\mathcal{S}}) \leq \varepsilon. \tag{3}$$

There are various media forms of carrier that can be used for information hiding, including image [7], [8], audio [9], [10], text [11]–[18] and so on [19]. Text is the most widely used information carrier in daily life [20], the use of text as information hiding carrier has great research value and practical significance. However, compared with image and audio, texts have a higher degree of information coding, resulting in less redundant information, which makes it quite challenging to hide information inside [8]. For the above reasons, text steganography has appealed to a tremendous proportion of researchers' interests. In recent years, more and more text based information hiding methods have emerged [13]–[15], [21].

J. Fridrich [8] has summarized that, in general, steganography algorithms can utilize three different fundamental architectures that determine the internal mechanism of the embedding and extraction algorithms: steganography by cover selection, cover modification, and cover synthesis. In steganography by cover selection, Alice first encodes all the covers in a cover set and then selects different covers for transmission to achieve the covert message delivery. The advantage of this approach is that the cover is always "100% natural", but an obvious disadvantage is an impractically low payload [22]–[24]. The most studied steganography paradigm today is the steganography by cover modification. Alice implements the embedding of confidential information by modifying a given carrier. This kind of method has a wide range of applications on multiple carriers such as images [25], speeches [10], and texts [26]–[28]. But generally speaking, the redundant information space of images and speeches are relatively large, so a proper amount of modification will not cause great visual [25] or auditory changes [10]. For text, it has a higher information coding degree, whereas the amount of information redundancy is less, which limits the size of modifiable space [26]–[28]. Therefore, the methods based on modification are difficult to achieve a sufficiently high hidden capacity on the text carrier. The third type of method is steganography by cover synthesis. Alice automatically generates a carrier based on the confidential message that needs to be delivered, and embeds covert information during the generation process. This type of method usually has a high hidden capacity and is therefore considered a very promising research direction in the field of text steganography [17], [20], [21], [29]–[31]. However, previous works have been difficult to generate high-quality readable texts, so the concealment of this kind of method is limited [29]–[32]. Therefore, how to design a better model and generate higher quality text carriers has become an urgent problem in this field.

In this paper, we proposed a text steganography based on Recurrent Neural Networks (RNN-Stega), which falls into the third category that can automatically generate high-quality steganographic texts based on the secret bitstream that needs



Bit Stream: 0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 1 0 1 0 1 1 0 1 0 0 1 1 0 1 1 0 1
Generated Sentence: i'm surprised that the film was made in the early 1980s.
Bit Stream: 1 1 0 0 1 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0
Generated Sentence: the only thing that i can say is that the movie is a bit too long.

Fig. 1. The overall framework of the proposed method. The sender uses the proposed model to generate a set of natural texts based on the secret bitstream that needs to be transmitted, and then sends them over the open network channel. The receiver uses the decoding algorithm to decode the natural texts and obtain the confidential information. The example below shows the different sentences generated ultimately according to different hidden bitstreams.

to be hidden. The overall framework of the proposed method has been shown in Figure 1. We trained our model with a large number of samples written by people and obtained a good estimate of the statistical language model. According to the well-trained statistical language model, we are able to automatically generate high-quality texts. In the text generation process, we code each word reasonably based on its conditional probability distribution, and then control the text generation according to the bitstream. We can finely adjust the encoding part to control the information embedding rate, which in return ensures that the concealment and hidden capacity can be optimized at the same time through fine control. In this way, our model can guarantee an excellent concealment while achieving a higher hidden capacity compared to other methods.

In the remainder of this paper, Section II introduces related work, including steganography based on automatic text generation and Recurrent Neural Networks. A detailed explanation of the RNN-Stega and algorithm details of information hiding and extracting are elaborated in Section III. The following part, Section IV, presents the experimental evaluation results and gives a comprehensive discussion. Finally, conclusions are drawn in Section V.

II. Related Work

A. Steganography Based on Automatic Text Generation

Compared with other methods, the steganography methods based on automatic text generation are characterized by the fact that they do not need to be given carrier texts in advance. Instead, they can automatically generate a textual carrier based on confidential information. This type of method omits $\mathcal{C}$ in Equation (1), and the embedding function becomes:

$$Emb\colon\ \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{S}, \quad f(k_A, m) = s. \tag{4}$$

However, they still have to satisfy formula (3), that is, the steganographic operation should minimize the impact of the carrier on the semantic spatial distribution. This could be very hard, but also very promising.

For a long time, a great many researchers have tried to use automatic text generation technology to achieve information hiding. For example, in the early stage, in order to ensure that the generated text is consistent with the training sample in the probability distribution of the characters, Wayner [29] proposed a mimic function. They constructed a Huffman tree to learn the statistical distribution of each character in the training samples beforehand, and then generated texts with the same character statistical distribution. However, they only considered the statistical distribution of the characters in the sentences but did not consider the semantic information, thus the texts they generated were meaningless and had poor concealment. In addition, Chapman and Davida [30] tried to use syntactic template or syntax structure tree to generate texts, they expected the generated texts could conform these syntactic rules. Obviously, the texts generated by this method follow a very simple pattern, and they generally don't look smooth enough.

These early works told us that for these high-level information coding carriers such as text, when considering the automatic generation of corresponding steganographic covers, we should not only consider the similarity of probability distributions at the character level, but should consider more from a higher semantic level. We should generate steganographic carriers that are more in line with the semantic expression patterns of training samples.

Therefore, a lot of researchers combine text steganography with statistical natural language processing, and a large number of natural language processing techniques have been used to automatically generate steganographic text [14], [20], [31]–[33]. Most of these works use the Markov chain model to calculate the number of common occurrences of each phrase in the training set and obtain the transition probability. Then the transition probability can be used to encode the words and achieve the purpose of hiding confidential information in the text generation process [20], [31]–[33]. This kind of method greatly improve the quality of the generated texts compared to the previous methods. However, due to the limitations of the Markov model, the textual steganographic carriers they generate are still not perfect.

In the field of statistical natural language processing, they usually use statistical language model to model a sentence. A language model is a probability distribution over sequences of words, which can be expressed by the following formula:

$$\begin{aligned} p(S) &= p(w_1, w_2, w_3, \ldots, w_n) \\ &= p(w_1)p(w_2 \mid w_1) \ldots p(w_n \mid w_1, w_2, \ldots, w_{n-1}), \end{aligned} \tag{5}$$

where S denotes the whole sentence with a length of n and w_i denotes the i-th word in it. $p(S)$ assigns the probability to the whole sequence. It is actually composed of the product of n conditional probabilities, each of the conditional probability calculates the probability distribution of the n-th word when the first $n-1$ words are given, that is $p(w_n \mid w_1, w_2, \ldots, w_{n-1})$. In fact, almost all automatic text generation algorithms are designed to fit such a statistical language model or to obtain an optimal conditional probability estimate for the n-th word [20], [31], [34], [35]. Nevertheless, the inherent drawback of Markov model is that it makes two approximations in the process of estimating such a conditional probability. Firstly, when the Markov model calculates the conditional probability of the n-th word, it only consider the previous few words rather than all the previous $n-1$ th word. For example, for a Markov model with second-order, it turns to be:

$$p(w_n \mid w_1, w_2, \ldots, w_{n-1}) \approx p(w_n \mid w_{n-2}, w_{n-1}). \tag{6}$$

The reason for this approximation is the need of the second approximate, which is, Markov model does not actually calculate the conditional probability, but uses the frequency to approximate the probability. The calculation formula is as follows:

$$p(w_n \mid w_{n-2}, w_{n-1}) \approx \frac{count(w_{n-2}, w_{n-1}, w_n)}{count(w_{n-2}, w_{n-1})}, \tag{7}$$

where $count(w_{n-2}, w_{n-1}, w_n)$ is the number of occurrences of phrase $\{w_{n-2}, w_{n-1}, w_n\}$ in the training set. However, according to the large number theorem [36], only when the number of sample is more enough, the frequency can be approximated as the probability. Therefore, under this restriction, the order of the Markov model cannot be too large, otherwise the number of occurrences of phrases in the training set will be too small, so the first approximation must be performed. It is precisely because of these two approximations that the Markov model cannot obtain an optimal conditional probability estimate as well as an ideal statistical language model. Therefore, the steganographic text generated based on Markov's method is not perfect [20], [31]. Then, how to generate a higher quality text carrier and realize information hiding in the generation process becomes an urgent problem that needs to be addressed.

B. Recurrent Neural Network

Automatically generating high-quality texts has always been hard because its high coding degree. In recent years, with the extensive application of deep neural network technology in Natural Language Processing, there has been an increasing number of text generation related work, like image captioning [34], [37], neural machine translation [35], dialogue systems [38], and so on. Most of these works can be put into the same technical framework, called Encoder-Decoder framework. They use different techniques to embed input information into a vector, and then use Recurrent Neural Networks to generate texts based on this input vector.

Recurrent Neural Network [39] is a special artificial neural network model. Unlike other deep neural networks, RNN is actually not "deep" in space, the simplest RNN can have only one hidden layer. The fundamental feature of a Recurrent

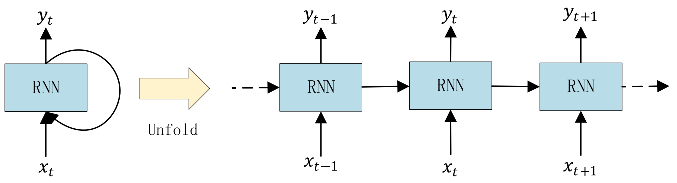


Fig. 2. The structure of RNN. The fundamental feature of a Recurrent Neural Network is that the network contains a feed-back connection at each step, so it can be extended in time dimension and form a "deep" neural network in time dimension.

Neural Network is that the network contains a feed-back connection at each step, so it can be extended in time dimension and form a "deep" neural network in time dimension, which has been shown in Figure 2.

For a recurrent neural network that has only one hidden layer, it can be described in the following set of formulas:

$$\begin{cases} h_t = f_h(W_h \cdot x_t + U_t \cdot h_{t-1} + b_h), \\ y_t = f_o(W_o \cdot h_t + b_o), \end{cases} \tag{8}$$

where x_t and y_t indicate the input and output vector at t-th step respectively, h_t represents the vector of hidden layer, $W_.$, $U_.$ and $b.$ are learned weight matrices and biases, f_h and f_o are nonlinear functions, where we usually use $tanh$ or $softmax$ function.

Due to its special structural characteristics, RNN is very suitable for modeling sequential signals. Theoretically, the simplest RNN model as described in Equation (8) can deal with arbitrary length sequence signals. However, due to the gradient vanish problem [40], it cannot handle with the problem of long-range dependence effectively. But its improved algorithm, Long Short-Term Memory (LSTM) model [41], can effectively solve this problem by elaborately designed unit nodes. The main improvement of LSTM is the hidden layer unit, which is composed of four components: a cell, an input gate, an output gate and a forget gate. It can store the input information of the past time into the cell unit so as to overcome the problem of long distance dependence, and realize the modeling of long time series. An LSTM unit can be described using the following formulas:

$$\begin{cases} I_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \\ F_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \\ C_t = F_t \cdot C_{t-1} + I_t \cdot \tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \\ O_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \\ h_t = O_t \cdot \tanh(C_t). \end{cases} \tag{9}$$

where I_t indicates the input gate, it controls the amount of new information to be stored to the memory cell. The forget gate, which is F_t, enables the memory cell to throw away previously stored information. In this regard, the memory cell C_t is a summation of the incoming information modulated by the input gate and previous memory modulated by the forget gate F_t. The output gate O_t allows the memory cell to have an effect on the current hidden state and output or block its influence. For simplicity, we denote the transfer function of LSTM units by $f_{LSTM}(*)$. We need to notice that when we calculate the output at time step t, the information we used based on the input vector at time step t, also includes the information stored in the cells at the previous $t-1$ moments. Therefore, the output at time step t can be written as

$$y_t = f_{LSTM}(x_t \mid x_1, x_2, \ldots, x_{t-1}). \tag{10}$$

Comparing Equation (10) and Equation (5), we find that if we consider the signal x_i at each time point in formula (10) as the i-th word in the sentence, it can exactly represent the conditional probability distribution of each word in the text, which is $p(w_n \mid w_1, w_2, \ldots, w_{n-1})$, and then it can perfectly model the statistical language model of the text, which has been shown as Formula (5). It is because of the characteristics of RNN that we can model each sentence as a time series signal, and the i-th word in the sentence represents the i-th signal response. Therefore, RNN is very popular in the field of Natural Language Processing (NLP), especially in text generation related fields [34], [35], [38].

Compared with the previous automatic text generation algorithms based on Markov model [20], [31], [32], RNN with LSTM units is obviously more reasonable in textual modeling. It completely abandons the double approximation of the Markov model. Firstly, it does not limit the signal for the previous $n-1$ moments when calculating the conditional probability distribution at the n-th moment. Instead, it tries to fuse the signals of the first $n-1$ moments as much as possible, and a lot of works have proved that it can indeed extract and integrate semantic information for long enough texts [35]. Secondly, when it calculates the conditional probability at the n-th time, it does not use frequency to approximate it. It is expected that the model can automatically learn such a probability distribution from a large number of text samples, using a well-designed loss function.

Through the above comparison, RNN can better model the text than the Markov model so as to obtain a better conditional probability distribution and also a better statistical language model. In this way, RNN has the ability to generate higher quality natural text carriers with higher concealment, which we will explain in detail in the experimental part.

Perhaps most closely related to our work is the model proposed by Fang *et al.* [21]. They also use LSTM to build a text concealment system. While similarly motivated, our work differs from theirs in a number of ways. Firstly, they divide the dictionary in advance and fix the code for each word. Then in the generation process, the most appropriate word in the corresponding subset is selected as the output according to the code stream. By contrast, we dynamically code each word according to its conditional probability distribution at each moment in the generation process. We fully consider the conditional probability distribution at each moment and use it as much as possible, thus the generated text is more natural and of higher quality. For Tina's model, if they want to adjust the information embedding rate, they need to regroup and encode all the words in the entire dictionary, and as the embedding rate increases slowly, the quality of the generated texts will drop very quickly. Secondly, if their concealment system is applied, in addition to the network structure and parameters, the communication parties need to share the grouped dictionary. What's more, each time they adjust the embedding rate, they have to re-share the divided dictionary.

Taking the factors above into consideration, our model is thus more convenient as it uses dynamic coding and eliminates the necessity to share fixed-coded codebooks. In the experimental part, we will conduct detailed comparisons.

III. RNN-Stega Methodology

In order to achieve large-capacity text information hiding while guaranteeing the naturalness of the steganographic texts, that is, guaranteeing the hiding capacity while ensuring a sufficiently high degree of concealment, in this work, we conduct a text steganography based on Recurrent Neural Networks, which can automatically generate high-quality texts based on the input bitstream. A detailed explanation of the proposed model has been shown in Figure 3. This section describes the details of the proposed model, which consists of three main modules: automatic text generation module, information hiding module and information extraction module. The automatic text generation module is mainly to model the natural texts and use the self-learning ability of the neural network to learn a good statistical language model from a large number of samples, as well as estimating the conditional probability distribution at each moment. The information hiding module realizes the capability of hiding a secret bitstream by properly coding the conditional probability distribution at each moment, including fixed-length encoding (FLC) and variable-length encoding (VLC). The information decoding module simulates the receiving end. After receiving the natural text generated by the proposed model embedded with the covert information, they need to decode it and obtain the confidential message.

A. Automatic Text Generation Based on RNN

In the process of automatic text generation, we mainly take advantage of RNN's powerful ability in feature extraction and expression for sequential signals. It can fuse the signals of the previous $t-1$ moments and calculate the probability distribution of the signal at time t, as shown in Equation (10).

As we have mentioned before, each sentence S can be regarded as a sequential signal and the i-th word $Word_{S_i}$ can be viewed as the signal at time step i. When we use LSTM to generate sentences, we input the i-th word of sentence S at time step i. The first layer of our neural network is an embedding layer and it maps each word to a dense semantic space with a dimension of d, that is $Word_{S_i} \in \mathbb{R}^d$. Then, for each sentence S, we can illustrate it with a matrix $S \in \mathbb{R}^{L\times d}$, where the i-th row indicates the i-th word in sentence S and L is the length of it, that is

$$S = \begin{bmatrix} Word_{S_1} \\ Word_{S_2} \\ \vdots \\ Word_{S_L} \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{L,1} & x_{L,2} & \cdots & x_{L,d} \end{bmatrix} \quad (11)$$

In general, a recurrent neural network consists of multiple network layers, each of which with multiple LSTM units. We use n_j to indicate the LSTM units number of j-th hidden layer U^j, so the units of j-th layer can be represented as

$$U^j = \{u_1^j, u_2^j, \cdots, u_{n_j}^j\}.$$

For the first hidden layer, the input of each unit u_i^1 at time step t is the weighted sum of the elements in $Word_{St}$, that is

$$u_{i,t}^1 = \sum_{k=1}^{d} w_{i,k}^1 \cdot x_{j,k} + b_{i,t}^1, \quad (12)$$

Where $w_{\cdot}^1$ and b^1. are learned weights and biases. Then the output value of u_i^1 at time step t is

$$o_{i,t}^1 = f_{LSTM}(u_{i,t}^1) = f_{LSTM}(\sum_{k=1}^{d} w_{i,k}^1 \cdot x_{j,k} + b_{i,t}^1). \quad (13)$$

We can use a vector Out_t^j to represent the output of the j-th hidden layer at time step t, and each element in Out_t^j indicates the output value of each unit in the j-th hidden layer at time step t, that is

$$Out_t^j = f_{LSTM}(x_t) = [o_{1,t}^j, o_{2,t}^j, \ldots, o_{n_j,t}^j]. \quad (14)$$

Previous work has shown that within a certain range, the more layers of neural network in space, the stronger the ability to extract and express features [42]. So we stack the network with multiple layers of LSTM units. Adjacent hidden layers can be connected by a transfer matrix. For example, the transfer matrix between l-th layer and $(l+1)$-th layer can be represented as a matrix $W^l \in \mathbb{R}^{n_l \times n_{l+1}}$:

$$W^l = \begin{bmatrix} w_{1,1}^l & w_{1,2}^l & \cdots & w_{1,n_{l+1}}^l \\ w_{2,1}^l & w_{2,2}^l & \cdots & w_{2,n_{l+1}}^l \\ \vdots & \vdots & \ddots & \vdots \\ w_{n_l,1}^l & w_{n_l,2}^l & \cdots & w_{n_l,n_{l+1}}^l \end{bmatrix} \quad (15)$$

Then the input of each unit u_i^l in the l-th hidden layer at time step t is the weighted sum of the output values of the units in the previous layer, that is

$$u_{i,t}^l = Out_t^{l-1} \cdot W^{l,m} = \sum_{k=1}^{n_{l-1}} w_{i,k}^l \cdot o_{k,t}^{l-1} + b_{i,t}^l. \quad (16)$$

And the output of the l-th layer at time step t is

$$\begin{aligned} Out_t^l &= f_{LSTM}(Out_t^l) = [o_{1,t}^l, o_{2,t}^l, \ldots, o_{n_l,t}^l], \\ o_{i,t}^l &= f_{LSTM}(u_{i,t}^l) = f_{LSTM}(\sum_{k=1}^{n_{l+1}} w_{i,k}^l \cdot o_{k,t}^l + b_{i,t}^l). \end{aligned} \quad (17)$$

As we have mentioned before, the output at time step t is not only based on the input vector of the current time x_t, but also the information stored in the cells at the previous $(t-1)$ moments according to Equation(10). Therefore, the output of the l-th hidden layer at time step t can be regarded as a summary of all previous t moments, that is, the information fusion of the previous t words $\{Word_1, Word_2, \ldots, Word_t\}$. Based on these features, after all the hidden layers, we add a softmax layer to calculate the probability distribution of the $(t+1)$-th word. To be more specific, we define the Prediction Weight(PW) as matrix $W_P \in \mathbb{R}^{n_l \times N}$, that is

$$W_P = \begin{bmatrix} w_{1,1}^p & w_{1,2}^p & \cdots & w_{1,N}^p \\ w_{2,1}^p & w_{2,2}^p & \cdots & w_{2,N}^p \\ \vdots & \vdots & \ddots & \vdots \\ w_{n_3,1}^p & w_{n_3,2}^p & \cdots & w_{n_l,N}^p \end{bmatrix} \quad (18)$$

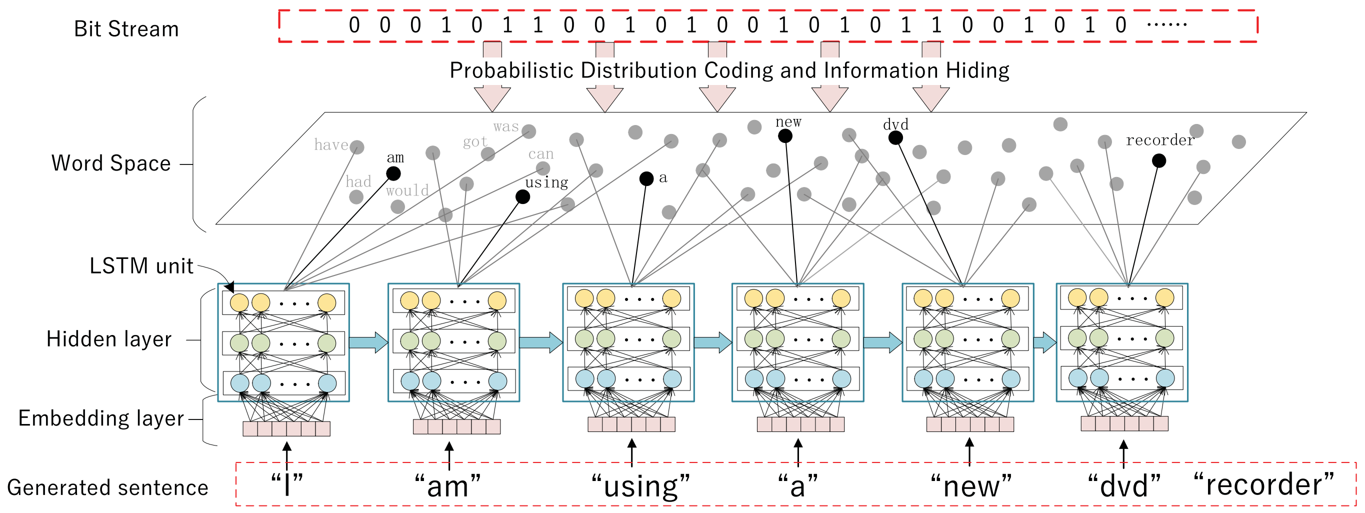


Fig. 3. A detailed explanation of the proposed model and the information hiding algorithm. The top of the figure is the bitstream that needs to be hidden, and the bottom is a natural sentence generated by the RNN based on the hidden information. The RNN model used in our algorithm is shown in the middle of the graph. It contains an embedded layer, multiple network layers with multiple LSTM units, and an output layer. The output layer is the probability distribution of the next word. We encode the probability distribution of words, and then select the corresponding word according to the secret bitstream, so as to achieve the purpose of hiding information.

where N indicates the number of words in dictionary D. Then we use this learned matrix W_P to calculate the score for each word in the dictionary D, that is

$$y_i = \sum_{k=1}^{n_l} w_{k,i}^{p} \cdot o_{i,t}^{l} + b_{i,t}^{p}, \tag{19}$$

where W_P and b^p are learned weight matrix and bias, the values in weight matrix W_P reflect the importance of each feature in o^l. The dimension of the output vector y is N. In order to calculate the probability of next word at each moment, we refer to the previous works [34], [35], [37], [38], adding a softmax classifier to the output layer to calculate the possible probability of each word:

$$p(Word_{D_i} \mid Word_{S_1}, Word_{S_2}, \ldots, Word_{S_t}) = \frac{\exp(y_i)}{\sum_{j=1}^{N} \exp(y_j)}. \tag{20}$$

All the parameters of the neural network, including each word vector, need to be obtained through training. In order to obtain a statistical language model consistent with the training samples, we define the loss function of the whole network to be a negative *log* of the statistical probability of each sentence:

$$\begin{aligned} Loss &= -log(p(S)) \\ &= -log(p(Word_1, Word_2, Word_3, \ldots, Word_L)) \\ &= -log(p(Word_1)p(Word_2 \mid Word_1) \\ &\quad \ldots p(Word_L \mid Word_1, Word_2, \ldots, Word_{L-1})) \\ &= -\sum_{t=1}^{L} log(p(Word_t \mid Word_1, \ldots, Word_{t-1})). \end{aligned} \tag{21}$$

In the training process, we update network parameters using backpropagation algorithm [43]. After minimizing the loss function through the iterative optimization of the network, we will get a language model that increasingly meets the statistical characteristics of the training samples.

B. Information Hiding Algorithm

In the information hiding module, we mainly code each word based on their conditional probability distribution, which is $p(w_n \mid w_1, w_2, \ldots, w_{n-1})$, to form the mapping relationship from the binary bitstream to word space. Our thought is mainly based on the fact that when our model is well trained, there is actually more than one feasible solution at each time point. After descending the prediction probability of all the words in the dictionary D, we can choose the top m sorted words to build the Candidate Pool (CP). To be more specific, suppose we use c_i to represent the i-th word in the Candidate Pool, then the CP can be written as

$$CP = [c_1, c_2, \ldots, c_m].$$

In fact, when we choose a suitable size of the Candidate Pool, any word c_i in CP selected as the output at that time step is reasonable and will not affect the quality of the generated text, so it becomes a place where information can be hidden. It is worth noting that at each moment we choose different words, the probability distribution of the words will be different at next time step according to the Equation(10). After we get the Candidate Pool, we need to find an effective encoding method to encode the words in it.

Obviously, different coding methods will affect the final performance of the entire algorithm. Here we provide two coding methods, one is fixed-length coding (FLC) based on a perfect binary tree, and the other is variable-length coding (VLC) based on a Huffman tree. A perfect binary tree is a binary tree in which all interior nodes have two children and all leaves have the same depth. It is easy to know that when the depth of a perfect binary tree is d, its number of nodes is $2^d - 1$ and the number of leaf nodes is 2^{d-1}. Huffman tree

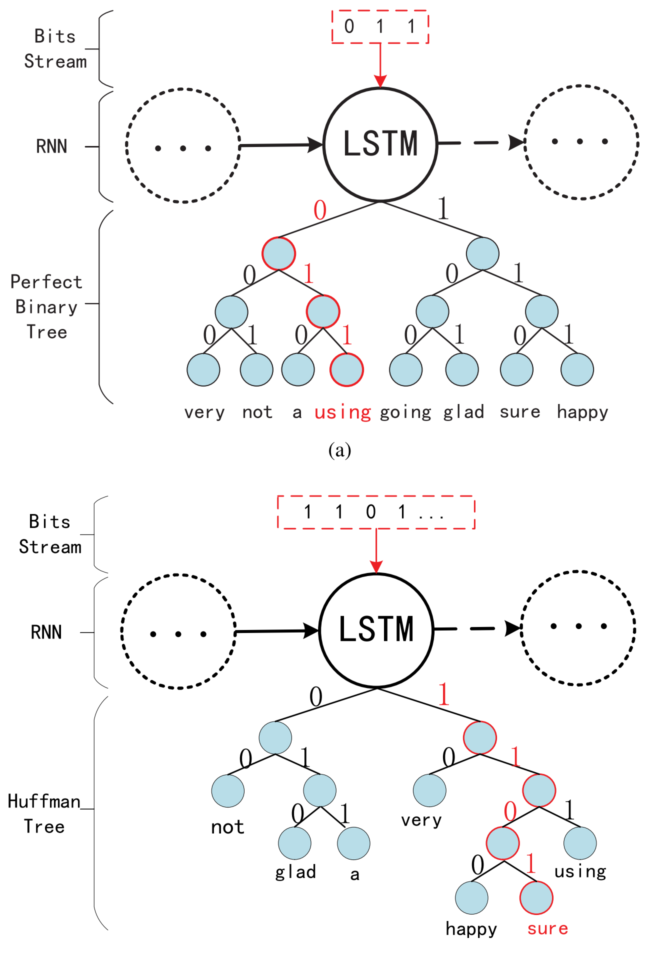


Fig. 4. Encoding words in Candidate Pool using (a) fixed-length coding (FLC) based on a perfect binary tree, or (b) variable-length coding (VLC) based on a Huffman tree.

is also a kind of binary tree. The difference is that it takes more into consideration of the probability distribution of each source symbol in the construction process, and can ensure that the code length required by the symbol with higher coding probability is shorter [44]. In both FLC and VLC, we represent each word in the Candidate Pool with each leaf node of the tree, the edges connect each non-leaf node (including the root node) and its two child nodes are then encoded with 0 and 1, respectively, with 0 on the left and 1 on the right, as shown in Figure 4.

Comparing these two coding methods, the advantages of FLC are that they are simple and efficient, and the length of each word's code is certain. The advantage of VLC is that the difference in the conditional probability distribution of each word is fully considered, which makes the words with higher probability more easily selected so that the quality of the generated text will be better. But the drawback is that we need to build a Huffman tree at every iteration in the generation process, which will affect the generation efficiency. In the experimental part, we will make a detailed comparison of these two coding methods.

When the words in the Candidate Pool are all encoded, the process of information hiding is to select the corresponding leaf node as the output of the current time according to the binary code stream that needs to be hidden. Before hiding information, the size of the candidate pool(CPS) at each moment must be determined first, which is denoted as m. For fixed-length coding, m must be an exponential power of 2, *i.e.* $m = 2^k$. Thus, the binary code length of each word is k. The process of information hiding is to read k bits at one time, find the corresponding coded leaf node from the root node of the perfect binary tree, and use its corresponding word as the output of the current time to complete the k-bit hiding process. For variable-length coding, it is not necessary to let m be an exponential power of 2; it only needs to construct a Huffman tree based on the probability distribution of words in the Candidate Pool. In the process of information hiding of VLC, the bitstream to be hidden is sequentially read at each moment, and then searched from the root node of the Huffman tree in order, until it encounters a leaf node. After then, the corresponding word is output at the current time, and the information embedding is completed. When all the confidential information is embedded, during each subsequent iteration, the model will choose and output the word with highest probability to guarantee the quality of the generated sentence.

In order to avoid the condition that two equal sequences of bits produce two equivalent text sentences, we constructed a keyword list. We counted the frequency of the first word of every sentence in the collected texts dataset. After sorting them in descending order, we choose the 100 most frequent words to form the keyword list. During the generation process, we will randomly select the words in the keyword list as the beginning of the generated steganographic sentence. Thus, even for the same input bitstream, since the first word is randomly selected, the conditional probability distribution of each subsequent word is inconsistent, so the resulting sentence will be completely different.

Algorithm details of the proposed information hiding method are shown in Algorithm 1. With this method, we can generate a large number of smooth natural sentences according to the input secret bitstream. And then these generated texts can be sent out through the open channel to achieve the purpose of hiding and sending confidential information with a high concealment.

C. Information Extraction Algorithm

Information hiding and extraction are two completely opposite operations. After receiving the transmitted sentence, the receiver needs to correctly decode the confidential information contained therein. The process of information hiding and extraction is basically the same. It is also necessary to use the same RNN network to calculate the conditional probability distribution of each word at each moment, then construct the same Candidate Pool and use the same coding method to encode the words in the Candidate Pool.

After receiving the texts which contain multiple sentences, the receiver Bob inputs the first word of each sentence as a key into the RNN which will calculate the distribution probability

Algorithm 1 Information Hiding Algorithm

```
Input:
  Secret bitstream: $B = \{0, 0, 1, 0, 1, \ldots, 0, 1, 0\}$
  Size of Candidate Pool(CPS): $m$
  Keyword list: $A = \{key_1, key_2, \ldots, key_F\}$
Output:
  Multiple generated steganography text: $Text = \{S_1, S_2, \ldots, S_N\}$

1: Data preparing and RNN training;
2: while not the end of B do
3:     if not the end of current sentence then
4:         Calculate the probability distribution of the next word according to the previously generated words using well trained RNN;
5:         Descending the prediction probability of all the words and select the top $m$ sorted words to form the Candidate Pool(CP);
6:         Construct a binary tree(fixed-length coding) or a Huffman tree(variable-length coding) according to the probability distribution of each word in the CP and encode the tree;
7:         Read the binary stream, and search from the root of the tree according to the encoding rules until the corresponding leaf node is found and output its corresponding word;
8:     elseãŁŁãŁŁ
9:         Random select a keyword $key_i$ in the keyword list $A$ as the start of the next sentence;
10: if not the end of current sentence then
11:     Select the word with the highest probability outside the Candidate Pool as the output of current time;
12:     Select the word with the highest probability at each moment as output until the end of the sentence;
13: return Generated sentences
```

Algorithm 2 Information Extraction Algorithm

```
Input:
  Multiple generated sentences: $Text = \{S_1, S_2, \ldots, S_N\}$.
  Size of Candidate Pool(CPS): $m$
Output:
  Secret bitstream: $B = \{0, 0, 1, 0, 1, \ldots, 0, 1, 0\}$

1: for each sentence $S$ in Text do
2:     Enter the first word of the sentence $S$ into the trained RNN as the Key;
3:     for each word in $Word_i$ sentence $S$ do
4:         Calculate the probability distribution of the next word according to the previously words using RNN;
5:         Descending the prediction probability of all the words and select the top $m = 2^k$ sorted words to form the Candidate Pool (CP);
6:         Using fixed-length coding or variable-length coding to encode words in the Candidate Pool;
7:         if $Word_i$ in CP then
8:             Based on the actual accepted word $Word_i$ at each moment, determine the path from the root node to the leaf node;
9:             According to the tree coding rule, ie, the left side of the child node is 0 and the right side is 1, extract the corresponding bitstream and append to $B$;
10:        else
11:            The information extraction process ends;
12: return Extracted secret bitstream $B$;
```

of the words at each subsequent time point in turn. At each time point, when Bob gets the probability distribution of the current word, he firstly sorts all the words in the dictionary in descending order of probability, and selects the top m words to form the Candidate Pool. Then he builds the perfect binary tree or Huffman tree according with the same rules to encode the words in the candidate pool. Finally, according to the actual transmitted word at the current moment, the path of the corresponding leaf node to the root node is determined so that we can successfully and accurately decode the bits hidden in the current word. By this way, the bitstream hidden in the original texts can be extracted very quickly and without errors.

IV. Experiments and Analysis

In this section, we designed several experiments to test the proposed model from the perspectives of information hiding efficiency, information imperceptibility and information hidden capacity. For the information hiding efficiency, we tested the average time needed to generate a fixed-length text and hid the fixed-length bitstream. For the imperceptibility, we compared and analyzed the quality of the texts generated at different embedding rates with the training text, and the ability to resist steganographic detection. For the hidden capacity, we analyzed how much information can be hidden in the generated texts, and compared it with some other text steganography algorithms. Before testing, in section 4.1, we introduce the dataset we used in this study. In section 4.2, we show the structure and parameters setting of the Recurrent Neural Networks used in the proposed model and the details of model training. In section 4.3, we introduce the performance evaluation metric used in this work and the evaluation results. Also, we conduct a detailed analysis and discussion of the evaluation results.

A. Data Preparing and Model Training

Since we expect that the proposed model can automatically imitate and learn the sentences written by humans, we need a large amount of human-written natural texts to train our model, after which we can obtain a similar enough statistical language model. In the framework of concealment systems, we generally assume that Alice and Bob are passing information through the public channel. So we chose three large-scale text datasets containing the most common text media on the Internet as our training sets, which are Twitter [45], movie reviews [46] and News [47].

For Twitter, we chose the sentiment140 dataset published by Alec Go *et al.* [45]. It contains 1,600,000 tweets extracted

TABLE I
THE DETAILS OF THE TRAINING DATASETS

Dataset	Twitter [45]	IMDB [46]	News [47]
Average Length	9.68	19.94	22.24
Sentence Number	2,639,290	1,283,813	1,962,040
Words Number	25,551,044	25,601,794	43,626,829
Unique Number	46,341	48,342	42,745

using the Twitter API. For the movie review dataset, we chose the widely used IMDB dataset published by Maas *et al.* [46]. The texts of the two datasets above are of the social media type. In addition, we also chose a news dataset [47] containing relatively more standard texts to train our model. It contains 143,000 articles from 15 American publications, including the New York Times, Breitbart, CNN, etc. The topics of the dataset are mainly politically related and the published time is mainly between 2016 and July 2017.

Before model training, we need to conduct data pre-processing, which mainly consists of converting all words into lowercase, deleting special symbols, emoticons, web links, and filtering low-frequency words. After pre-processing, the details of the training datasets are shown in Table I.

Our framework is implemented with Python based on Tensorflow [48]. GeForce GTX 1080Ti GPU and CUDA 8.0 are used for training acceleration. Almost all the parameters in our model can be obtained through training, but there are still some hyper-parameters need to be determined. To determine these hyper-parameters, we designed multiple sets of comparative experiments. Finally, to synthesize the performance of all aspects of the model, the hyper-parameters of our model are set as follows: each word is mapped to an 800-dimensional vector, and the number of LSTM hidden layers is 3, with each layer containing 800 LSTM units. We used *tanh* as nonlinear activation function σ in Equation(9). During model training, in order to strengthen the regularization and prevent overfitting, we adopt the dropout mechanism [49] during the training process. We chose Adam [50] as the optimization method. The learning rates are initially set as 0.001 and batch size is set as 128, dropout rate is 0.5.

B. Evaluation Results and Discussion

In this section, we tested and analyzed the performance of the proposed model from three aspects: information hiding efficiency, information imperceptibility and hiding capacity.

1) Information Hiding Efficiency: Information hiding efficiency calculates how long the model takes to hid a fixed length of confidential information. For the proposed model, since the words in the candidate pool need to be tree-coded at each iteration, the size of the candidate pool (CPS) will significantly affect the efficiency of information hiding. In this experiment, we tested the efficiency of hiding information at different embedding rates. Specifically, for each CPS, we generated $1,000$ texts, each of which was limited to 50 words. We also tested the hiding efficiency of the model proposed by Fang *et al.* [21]. For Tina's model, the number of blocks in their dictionary (ie, Bins Number) corresponds to the CPS. When these two parameters are equal, the embedding rates of these two models are consistent. Therefore, we use this indicator as the basis for comparison. The results are shown in Table II and Figure 5.

TABLE II
THE AVERAGE TIME REQUIRED FOR EACH MODEL TO GENERATE A SENTENCE CONTAINING 50 WORDS AT DIFFERENT EMBEDDING RATES

CPS	[21]	Our-FLC	Our-VLC
2	6.366±0.670	3.846±0.596	4.514±0.850
4	8.641±1.100	4.651±0.624	8.195±1.002
8	15.578±1.890	4.776±0.628	14.098±1.329
16	29.059±5.663	4.830±0.673	25.592±2.118
32	46.070±8.921	4.854±0.654	46.703±3.733

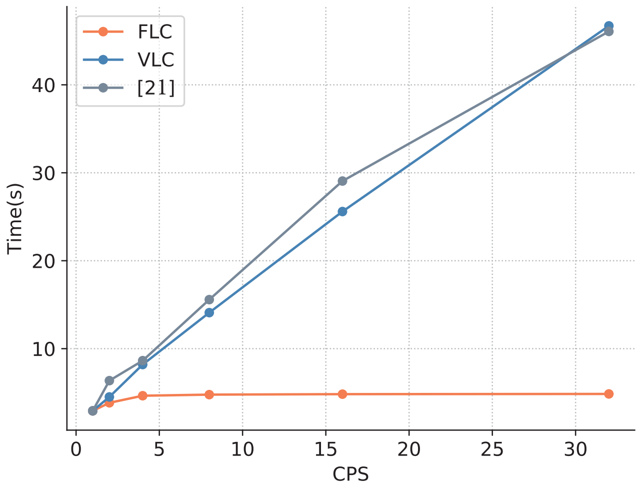


Fig. 5. The average time required for each model to generate a sentence containing 50 words at different embedding rates.

From Table II and Figure 5, we can find that, firstly, for both of the proposed model and Tina's model [21], as the embedding rate increases, the time required to generate a text carrier of a specific length gradually increases. Secondly, for the proposed model, constructing a Huffman tree takes more time than constructing a binary tree during each iteration, so VLC takes more time than FLC to generate a certain length of text. However, when we compared Tina's model with the proposed model, we found that Tina's model required almost the same amount of time to hid information as the VLC model, and even slightly higher than the VLC model. In this comparison, the FLC model has an obvious advantage in the efficiency of information hiding. We can see that the average time the FLC model takes to generate a 50-words text is about 2.928-4.854 seconds (when CPS changes from 1 to 32), which shows that the proposed model has a very high information hiding efficiency.

2) Imperceptibility Analysis: The purpose of a concealment system is to hide the existence of information in the carrier to ensure the security of important information. Therefore,

TABLE III
THE MEAN AND STANDARD DEVIATION OF THE PERPLEXITY RESULTS OF DIFFERENT STEGANOGRAPHY METHODS AT DIFFERENT EMBEDDING RATES ON EACH DATASET

Methods		Markov Model Based		LSTM Based			
Dataset	bpw	[20]	[31]	[21]	Our-FLC	Our-VLC	bpw(VLC)
IMDB [46]	1	430.38±107.09	212.58±168.42	29.73±4.70	**20.17±3.95**	20.23±4.18	1.000
	2	432.16±110.23	269.51±155.14	50.94±9.07	29.88±6.34	**23.68±4.82**	1.838
	3	430.61±107.19	304.71±148.39	87.02±17.65	50.09±11.43	**28.04±6.44**	2.498
	4	436.19±109.27	332.32±137.60	186.87±44.95	98.68±24.79	**36.31±9.11**	3.145
	5	433.95±110.50	348.36±131.98	391.30±85.03	211.37±55.72	**47.68±16.62**	3.721
News [47]	1	485.47±126.80	243.57±198.82	38.99±18.91	**17.47±6.83**	17.90±7.74	1.000
	2	487.65±133.83	302.43±186.00	70.85±14.51	50.52±18.07	**34.81±14.01**	1.777
	3	483.03±128.49	326.62±170.20	128.39±36.02	81.46±34.02	**41.23±15.21**	2.467
	4	493.30±129.99	368.07±165.91	255.86±83.36	134.99±44.08	**55.47±19.07**	3.153
	5	485.31±132.12	382.99±151.82	450.40±162.64	272.45±86.91	**66.01±23.15**	3.855
Twitter [45]	1	445.16±180.21	184.09±121.98	30.70±8.31	**19.29±4.39**	20.16±4.93	1.000
	2	445.64±166.67	257.36±135.78	48.24±10.34	31.41±7.18	**24.00±5.86**	1.825
	3	448.52±173.66	302.66±134.94	120.50±29.05	54.69±14.09	**29.83±7.89**	2.482
	4	440.26±159.91	333.20±134.40	201.15±49.33	102.56±27.196	**40.18±11.30**	3.137
	5	440.08±166.40	349.78±124.67	349.08±88.04	216.56±58.53	**55.00±17.31**	3.826

the imperceptibility of information is the most important performance evaluation factor of a concealment system. Generally speaking, as shown in Equation (3), we generally expect that the steganographic operation will not cause differences in the distribution of carriers in the semantic space. For the steganography methods based on the formula (1), it is possible to ensure that the statistical distribution characteristics of the carrier are unchanged by modifying the region in which the carrier is not sensitive [10], [25]. The model proposed in this paper is carried out under the framework of Equation (3), that is, it automatically generates a steganographic carrier based on confidential information without being given a carrier in advance. However, it is also necessary to abide by Equation (3), that is, the generated text carrier should be as consistent as possible with the statistical distribution of the normal carrier, which is actually more challenging. Therefore, in this section, we designed multiple sets of experiments to verify the performance of this aspect of the proposed model.

First, we need to test whether the sentences generated by our model are close enough to the non-steganographic carrier (*i.e.* training texts) on the statistical language model, otherwise it would be very easy to distinguish. In information theory, $perplexity$ is a measurement of how well a probability distribution or probability model predicts a sample. It can be used to compare probability models. In the field of Natural Language Processing, $perplexity$ is a standard metric for sentence quality testing [21], [39], [51]. It is defined as the average per-word log-probability on the test texts:

$$\begin{aligned} perplexity &= 2^{-\frac{1}{n} log\, p(s_i)} \\ &= 2^{-\frac{1}{n} log\, p(w_1, w_2, w_3, \ldots, w_n)} \\ &= 2^{-\frac{1}{n}\sum_{j=1}^{n} log\, p(w_i | w_1, w_2, \ldots, w_{j-1})}, \end{aligned} \tag{22}$$

where $s_i = \{w_1, w_2, w_3, \ldots, w_n\}$ is the generated sentence, $p(s_i)$ indicates the probability distribution over words in sentence s_i and the probability is calculated from the language model of the training texts. By comparing Equation (22) with Equation (5), we find that $perplexity$ actually calculates the difference in the statistical distribution of language model between the generated texts and the training texts. The smaller its value is, the more consistent the generated text is with the statistical distribution of the training text.

In order to objectively reflect the performance of the proposed model, we choose three text steganographic methods, which are also based on text automatic generation technology, as our baseline. Both models proposed in [20] and [31] are steganographic algorithms that use Markov models for automatic text generation. The model proposed by Fang *et al.* [21] also use LSTM to generate steganographic texts. While similarly motivated, the biggest difference is that they divide the dictionary in advance and fix the code for each word. However, we dynamically code each word according to the conditional probability distribution in the generation process.

We trained each model on three datasets, and for each embedding rates, we generated 1,000 sentences for testing. The mean and standard deviation of the $perplexity$ were tested and the results are shown in Table III and Figure 6. Since the number of embedded bits per word(bpw) in VLC is uncertain, we calculated the average number of bits hidden in each word of the generated text, which is listed on the right side of Table IV as bpw(VLC). The samples of the generated steganographic sentences can be found in the supplemental materials.

Based on these results, we can draw the following conclusions. Firstly, on each dataset, for each steganography algorithm(except for [20]), as the embedding rate increases,

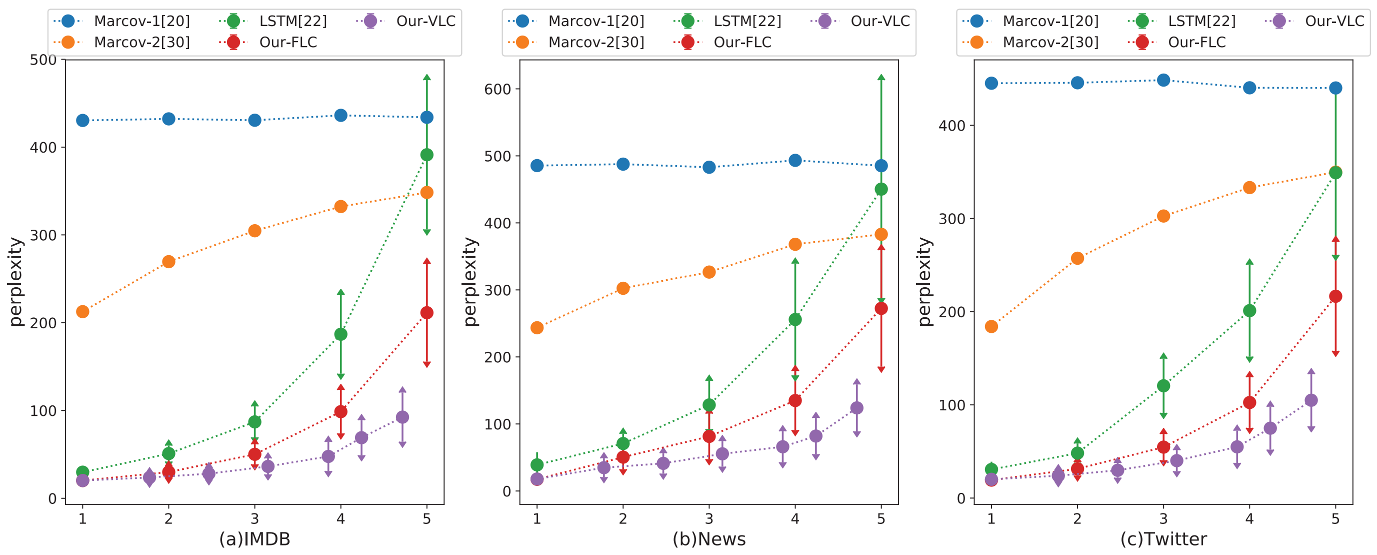


Fig. 6. The results of different steganography methods at different embedding rates on each dataset.

TABLE IV

For Each Data Set, the Earth Mover's Distance (EMD) of the Generated Steganographic Text and the Training Text in the High-Dimensional Semantic Space at Different Embedding Rates

Methods		Markov Model		LSTM Based			
Dataset	bpw	[20]	[31]	[21]	FLC	VLC	bpw
IMDB	1	2.404	1.889	1.824	**1.807**	1.816	1.000
	2	2.405	1.976	1.967	1.905	**1.831**	1.838
	3	2.387	2.131	2.149	2.085	**1.868**	2.498
	4	2.387	2.267	2.306	2.144	**1.913**	3.145
	5	2.394	2.354	2.314	2.178	**1.935**	3.721
News	1	2.574	2.071	2.061	1.969	**1.961**	1.000
	2	2.581	2.082	2.115	2.032	**2.017**	1.777
	3	2.596	2.245	2.296	2.159	**2.080**	2.467
	4	2.610	2.394	2.433	2.297	**2.108**	3.153
	5	2.587	2.520	2.485	2.398	**2.120**	3.855
Twitter	1	1.737	1.308	1.229	**1.212**	1.223	1.000
	2	1.918	1.496	1.452	1.366	**1.235**	1.825
	3	1.920	1.722	1.699	1.548	**1.272**	2.482
	4	1.937	1.905	1.783	1.647	**1.248**	3.137
	5	2.044	2.052	2.034	1.792	**1.235**	3.826

the *perplexity* will gradually increase. That is, the statistical language distribution difference between the generated text and the training samples will gradually increase. The reason is that as the number of bits hidden in each word increases, the word selected as the output is increasingly controlled by the hidden bits in each iterative process, and it is increasingly difficult to select the words that match the statistical distribution of the training text best. For [20], they neglect the transition probability of each word in the iterative process, so no matter how many words are selected as candidates, the *perplexity* of the generated text will remain at a high level. Secondly, the LSTM-based text generation method is better than the Markov model-based text generation method. This is also consistent with our analysis in the related work section. When fitting the statistical language model of the training text, the LSTM-based automatic text generation method has a better performance because there is no double approximation of the statistical language model like the Markov model. So the generated texts are also more in line with the statistical distribution of the training texts. Thirdly, although both are LSTM-based text automatic generation steganography algorithms, the proposed model has a better performance than [21] at the same embedding rate on each dataset. This situation is not the cause of LSTM, because our two models used the same trained LSTM model for text generation in the experimental phase. The difference in the final results is due to the difference in the coding part. Tina's model [21] uses fixed coding, that is, by dividing the dictionary containing all words in advance and then encoding the words. This kind of coding method may bring some disadvantages. For example, as the embedding rate increases, the number of blocks divided by dictionaries increases. At some iterations, it is even possible that it is hard to find a suitable word as an output in the block corresponding to the code stream. This makes the quality of their generated text decline rapidly as the embedding rate increases, and may even be worse than the Markov model-based approach. However, for the proposed model, since we fully consider the conditional probability distribution of each word and dynamically code each word in the coding process, the text we generate is more in line with the statistical distribution law of the training text. Finally, when comparing the two proposed encoding methods, namely FLC and VLC, although VLC's information encoding efficiency is lower than FLC, the quality of generated text by VLC is better. The major reason is that the FLC method ignores the difference in the conditional distribution probability of each word in the Candidate Pool during encoding process. The VLC method makes full use of the difference in the conditional probability distribution for each word in the Candidate Pool. It performs Huffman coding according to the different conditional probabilities of each word, so the words that are more in line with the language model have shorter coding and are easier to be selected as the output at each iteration. This makes the text it generates closer to the statistical distribution of the training text.

In addition to comparing the differences in the statistical language models of each generated model, we further compared the overall statistical distribution similarity between the steganographic text generated by each model and the training samples. Le and Mikolov [52] have proposed a method for extracting high-dimensional semantic features of text, and a large number of experiments have proved that the distribution of text in the high-dimensional space can reflect its semantic relevance.

We first randomly extracted 1,000 samples from each training data set, and then trained Le's model with the remaining sentences to obtain the mapping relationship from the training set to the high-dimensional semantic space. The dimension of the semantic space is set to be 100. Then we used this mapping model to map 1,000 steganographic texts generated by each algorithm and the randomly extracted 1,000 training texts (without participating in model training) into high-dimensional semantic space. Figure 7 shows the results of dimensionality reduction and visualization using t-Distributed Stochastic Neighbor Embedding (t-SNE) [53] technique, including Tina's model [21] and the proposed FLC and VLC models at different embedding rates. Results in Figure 7 are based on the IMDB dataset, and the results of the other datasets can be found in the supplemental materials. From Figure 7, we can see that as the embedding rate increases, the distribution of steganographic text generated by Tina's model [21] in the semantic space gradually spreads and deviates from the distribution of training text. Although there are also some deviations in the proposed model, the overall distribution is still in the same area as the training text. Especially for the VLC model, even if the embedding rate increases to 3.145 bits/word, the distribution of steganographic text and training text overlaps a lot.

In order to quantitatively compare the various methods, we calculated the Wasserstein Distance [54] of the text generated by each algorithm and the training text in the high-dimensional semantic space under different embedding rates. In statistics, the Wasserstein distance, also known as Earth Mover's Distance(EMD), is a measure of the distance between two probability distributions over a region D. The Earth Mover's Distance is defined as follows:

$$EMD(P, Q) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j} d_{i,j}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j}}. \tag{23}$$

Where P, Q are two distributions in the same space, m and n respectively represent the number of samples in P and Q, d_{ij} represents the distance between p_i and q_j, and f_{ij} can be regarded as the weight of the connection between p_i and q_j. For our case, P and Q respectively represent the distribution of the steganographic texts generated by each algorithm and the training samples in the semantic space, so $m = n = 1000$. For d_{ij}, we calculate the Euclidean distance. Since each sentence we generate is independent and equally important, the weight f_{ij} is taken as 1. The final test results are shown in Table IV. From Table IV we can get the similar conclusion, that is, compared with other methods, the steganographic text generated by the proposed model is closer to the training text in the high-dimensional semantic spatial distribution.

TABLE V

THE PROPORTIONS OF STEGANOGRAPHIC TEXT GENERATED BY EACH ALGORITHM THAT ARE CORRECTLY DETECTED BY THE STEGANALYSIS METHOD PROPOSED IN [55]

Dataset	bpw	[20]	[31]	[21]	FLC	VLC	bpw
IMDB	1	97.5%	20.5%	14.0%	**4.0%**	8.0%	1.000
	2	96.5%	38.5%	16.0%	**6.5%**	8.5%	1.838
	3	98.5%	59.5%	18.5%	5.0%	**4.5%**	2.498
	4	95.0%	65.5%	22.0%	**6.0%**	**6.0%**	3.145
	5	97.5%	64.5%	50.5%	19.5%	**6.5%**	3.721
News	1	95.5%	62.0%	8.5%	8.5%	**6.5%**	1.000
	2	93.5%	85.0%	18.0%	13.0%	**9.5%**	1.777
	3	97.0%	92.5%	43.0%	17.0%	**8.5%**	2.467
	4	95.0%	97.5%	64.5%	26.0%	**11.0%**	3.153
	5	97.5%	95.0%	80.0%	24.5%	**7.5%**	3.855
Twitter	1	93.0%	42.0%	11.0%	**6.0%**	7.0%	1.000
	2	95.5%	55.0%	23.5%	8.5%	**5.5%**	1.825
	3	97.0%	70.5%	31.0%	12.0%	**9.0%**	2.482
	4	97.5%	76.0%	42.5%	13.5%	**8.5%**	3.137
	5	94.5%	72.0%	67.0%	18.5%	**10.5%**	3.826

To further compare the concealment of each steganographic algorithms, we tested and compared the ability of each method to resist existing steganalysis methods. We mixed the $1,000$ steganographic sentences generated by each model together, then selected 80% of them as the negative training samples and the remaining 20% as the negative testing samples. After that, an equal amount of text was randomly selected from the training set to form training and testing positive samples. Then, we used the steganalysis method proposed by Meng *et al.* [55] to perform a simulated detection attack on this sample set. Finally, we compared the proportions of steganographic text generated by each algorithm that are correctly detected. The results are shown in Table V.

According to the results in Table V, we can clearly see that the steganographic text generated by the method based on the Markov model [20], [31] is the highest proportion to be detected. In addition, comparing the proposed model with Tina's model [21], when the embedding rate is gradually increasing, the detection proportion of Tina's model [21] is gradually increasing. At 5 bits/word, even 80% of the samples on the Twitter data set are detected. Comparing the two models proposed in this paper, the VLC model still shows stronger information imperceptibility. Under each dataset, in most cases, the proportion of detected samples is less than 10%.

In addition, we further tested the ability of proposed model and Tina's model's [21] to resist steganalysis at high embedding rates(more than 3 bits/word) respectively.

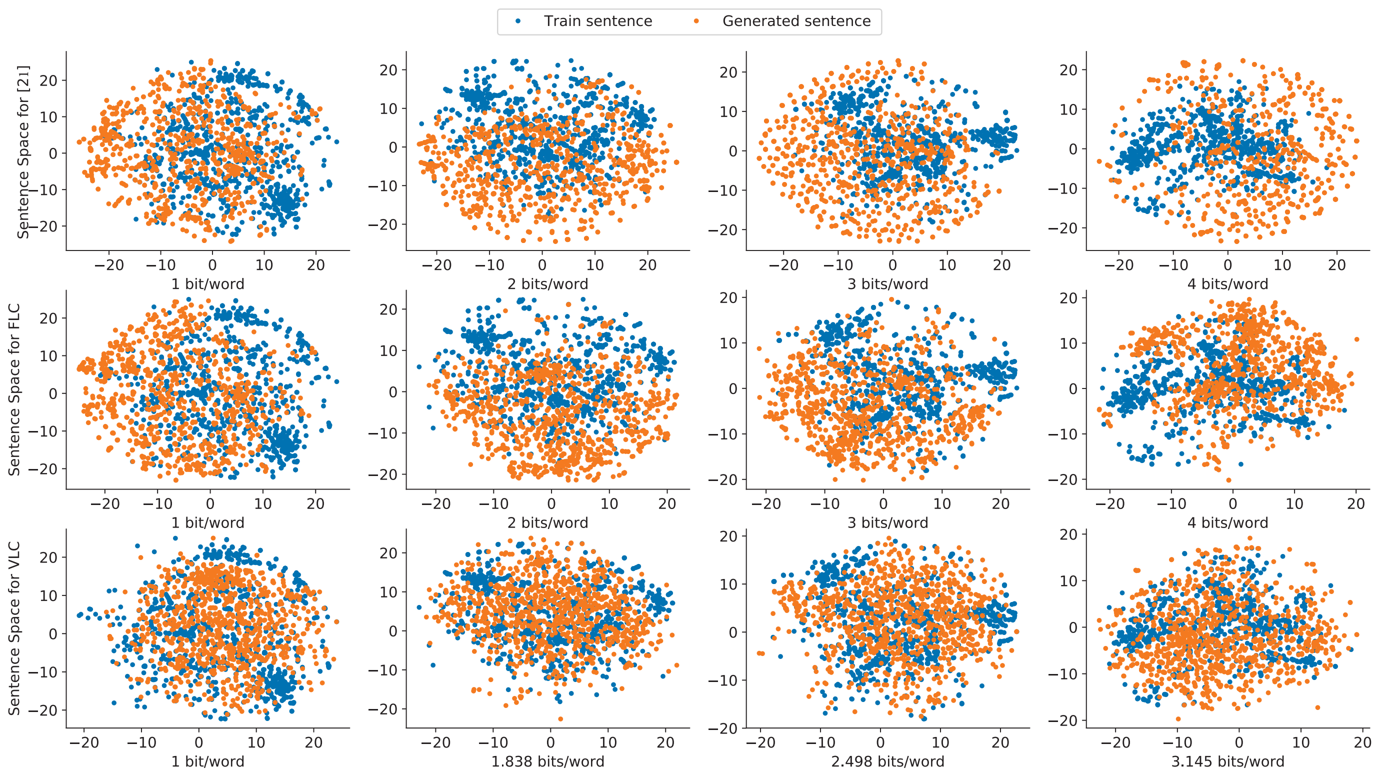


Fig. 7. The distribution difference between the generated steganographic text and the training text in the semantic space under different embedding rates. We use the [52] algorithm for semantic space mapping, and then use the t-SNE [53] algorithm to reduce and visualize.

We implemented three text steganalysis algorithms proposed in [55]–[57] to model existing text steganographic detection methods. In these three steganalysis methods, [56] first calculated the statistical correlation, which is measured by N-window mutual information, of the words in the generated steganographic text, and then use SVM to classify the given text into stego-text or normal text. Reference [57] proposed statistical text steganalysis tools based on Bayesian Estimation and Correlation Coefficient methodologies. In addition to the above-mentioned existing text steganalysis methods, we also implemented currently state of the art text classifier which was proposed in [58], to test if it can distinguish the generated steganographic text from the actual training text. Table VI records the detection accuracy of three steganalysis methods for different steganography algorithms at different embedding rates. Here, *Accuracy* calculates the proportion of true results (both true positives and true negatives) among the total number of cases examined:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}, \tag{24}$$

where TP is true positives, TN is true negatives, FP is False Positive and FN is False Negative. The closer the accuracy is to 0.5, indicates that the generated steganographic text is less likely to be detected, which means that the model has stronger ability to resist steganalysis. From Table VI, we can draw a similar conclusion with previous experiments that the proposed model's ability to resist steganalysis detection is stronger than [21]. It is worth mentioning that, although there still a certain percentage of steganographic texts generated by the proposed model were correctly distinguished when facing the current state of the art text classifier [58], the recognition proportion is much lower than the previous method.

3) Hidden Capacity Analysis: Embedding Rate(ER) calculates how much information can be embedded in the texts, which is an important index to evaluate the performance of a steganographic algorithm. It forms an opposite relationship with concealment, which usually decreases as the embedding rate increases. Previous works can hardly guarantee high concealment and large hidden capacity at the same time. In this section, we tested and analyzed the hidden capacity of the proposed model.

The calculation method of embedding rate is to divide the actual number of embedded bits by the number of bits occupied by the entire generated text in the computer. The mathematical expression is as follows:

$$\begin{aligned} ER &= \frac{1}{N}\sum_{i=1}^{N}\frac{(L_i - 1)\cdot k}{B(s_i)} \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{(L_i - 1)\cdot k}{8 \times \sum_{j=1}^{L_j} m_{i,j}} = \frac{(\overline{L} - 1)\times k}{8 \times \overline{L} \times \overline{m}}, \end{aligned} \tag{25}$$

where N is the number of generated sentences and L_i is the length of i-th sentence. k indicates the number of bits embedded in each word and $B(s_i)$ indicates the number of bits occupied by the i-th sentence in the computer. Since each English letter actually occupies one byte in the computer, i.e. 8 bits, the number of bits occupied by each English sentence is $B(s_i) = 8 \times \sum_{j=1}^{L_i} m_{i,j}$, where $m_{i,j}$ represents the number of letters contained in the j-th word of the i-th sentence. $\overline{L}$ and $\overline{m}$

TABLE VI
RESULTS OF DIFFERENT STEGANALYSIS METHODS

Dataset	Steganalysis	bpw	[21]	FLC	VLC	bpw(VLC)
IMDB	[55]	3	0.551	0.465	0.475	2.498
		4	0.624	0.468	0.485	3.145
		5	0.712	0.468	0.477	3.721
	[56]	3	0.615	0.553	0.562	2.498
		4	0.685	0.571	0.568	3.145
		5	0.777	0.613	0.590	3.721
	[57]	3	0.830	0.552	0.527	2.498
		4	0.857	0.580	0.534	3.145
		5	0.877	0.621	0.548	3.721
	[58]	3	0.823	0.642	0.527	2.498
		4	0.872	0.720	0.582	3.145
		5	0.885	0.727	0.618	3.721
News	[55]	3	0.497	0.480	0.480	2.467
		4	0.580	0.477	0.472	3.153
		5	0.713	0.585	0.487	3.855
	[56]	3	0.642	0.520	0.513	2.467
		4	0.750	0.588	0.542	3.153
		5	0.807	0.602	0.603	3.855
	[57]	3	0.759	0.659	0.559	2.467
		4	0.825	0.625	0.524	3.153
		5	0.847	0.632	0.536	3.855
	[58]	3	0.778	0.625	0.560	2.498
		4	0.830	0.670	0.612	3.145
		5	0.905	0.735	0.647	3.721
Twitter	[55]	3	0.535	0.473	0.485	2.482
		4	0.575	0.542	0.525	3.137
		5	0.682	0.585	0.535	3.826
	[56]	3	0.665	0.572	0.531	2.482
		4	0.652	0.566	0.522	3.137
		5	0.757	0.615	0.562	3.826
	[57]	3	0.718	0.548	0.493	2.482
		4	0.753	0.563	0.523	3.137
		5	0.783	0.643	0.523	3.826
	[58]	3	0.743	0.652	0.507	2.498
		4	0.778	0.697	0.522	3.145
		5	0.825	0.707	0.615	3.721

represent the average length of each sentence in the generated text and the average number of letters contained in each word. In the actual measurement, we found that the average length of each sentence is 16.11 and the average number of letters contained in each word is 4.79, that is, $\overline{L} = 16.11, \overline{m} = 4.79$. Figure 8 shows how the information embedding rate of the proposed model changes when we generate sentences with different lengths and when the number of bits embedded in each word is different.

From Figure 8, we can find that when the length of the generated texts is constant, the embedding rate increases linearly with the number of embedding bits per word increasing. When the number of bits embedded in each word of a text

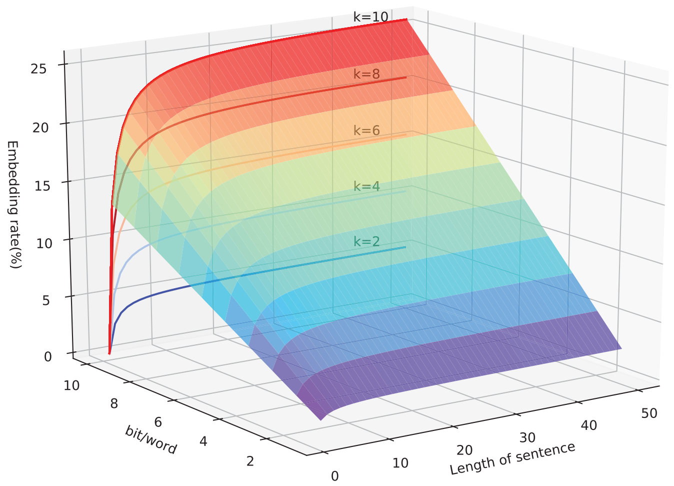


Fig. 8. The embedding rate varies with the length of the generated text and the number of bits embedded in each word in the text.

TABLE VII
THE COMPARISON OF THE EMBEDDING RATES BETWEEN THE PROPOSED MODEL AND THE PREVIOUS ALGORITHMS

Models	Embedding Rate(%)
Method proposed in [16]	0.30
Method proposed in [59]	5.42
Translation-based method proposed in [60]	0.33
CME-CIHM [61]	1.0
Method proposed in [62]	1.57
RNN-Stega (3bits/word)	**7.34**
RNN-Stega (5bits/word)	**12.23**
RNN-Stega (7bits/word)	**17.13**

is constant, the embedding rate increases with the increase of the length of generated texts. But there is an upper limit, the maximum embedding rate can reach more than 20%.

Table VII shows the comparison of the embedding rates between the proposed model and some previous algorithms which are not based on carrier automatic generation. The three lines at the bottom are the results of the proposed model when the bits embedded in each word are 3, 5, and 7. As shown in Table VII, the proposed model shows a state of the art embedding rate.

In the previous models, the embedding rate was only around 1% except the method proposed in [59], in which embedding rate can reach 5.42%. The reason why it has higher embedding rate is that it uses Chinese as the embedded language while others use English. The difference between Chinese and English is that for each Chinese word, it occupies 16 bits in the computer. While for English, each English word contains multiple English letters with an average of 4.79, and each English letter accounts for 8 bits. Thus, each English word actually occupies more bits in the computer than the Chinese word. Therefore, when the Chinese texts are used as the carrier of information embedding, the average embedding rate is much higher than that of using English texts as the embedding vector. Our model also uses English texts as the carrier of information embedding, but its information embedding rate still far exceeds the previous methods. If we change the information embedding carrier into Chinese texts, even higher information embedding rate can be obtained.

V. Conclusion

The topic that linguistic steganography based on text carrier auto-generation technology is fairly promising as well as challenging. Limited by the text automatic generation technology or the corresponding text coding methods, the quality of the steganographic text generated by previous methods is inferior, which makes its imperceptibility unsatisfactory. In this paper, we proposed a linguistic steganography based on Recurrent Neural Networks (RNN-Stega), which can automatically generate high-quality text covers on the basis of secret bitstream that needs to be embedded. We trained our model with a large number of artificially generated samples and obtained a good estimate of the statistical language model. In the text generation process, we proposed Fixed-Length Coding (FLC) and Variable-Length Coding (VLC) to encode words based on their conditional probability distribution. We designed several experiments to test the proposed model from the perspectives of information embedding efficiency, information imperceptibility and information hidden capacity. The results of the experiments show that the proposed model outperforms all the previous related methods, and achieves the state of the art performance. We hope that this paper will serve as a reference guide for the researchers to facilitate the design and implementation of better text steganography.

References

[1] C. E. Shannon, "Communication theory of secrecy systems," *Bell Labs Tech. J.*, vol. 28, no. 4, pp. 656–715, Oct. 1949.

[2] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding—A survey," *Proc. IEEE*, vol. 87, no. 7, pp. 1062–1078, Jul. 1999.

[3] L. Bernaille and R. Teixeira, "Early recognition of encrypted applications," in *Proc. Int. Conf. Passive Active Netw. Meas.*, 2007, pp. 165–175.

[4] G. J. Simmons, "The prisoners' problem and the subliminal channel," in *Advances in Cryptology*. Boston, MA, USA: Springer, 1984, pp. 51–67.

[5] M. Barni and F. Bartolini, *Watermarking Systems Engineering: Enabling Digital Assets Security and Other Applications*. Boca Raton, FL, USA: CRC Press, 2004.

[6] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker, *Digital Watermarking and Steganography*. San Mateo, CA, USA: Morgan Kaufmann, 2007.

[7] Z. Zhou, H. Sun, R. Harit, X. Chen, and X. Sun, "Coverless image steganography without embedding," in *Proc. Int. Conf. Cloud Comput. Secur.* Cham, Switzerland: Springer, 2015, pp. 123–132.

[8] J. Fridrich, *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[9] Z. Yang, X. Peng, and Y. Huang, "A sudoku matrix-based method of pitch period steganography in low-rate speech coding," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.* Cham, Switzerland: Springer, 2017, pp. 752–762.

[10] Y. F. Huang, S. Tang, and J. Yuan, "Steganography in inactive frames of VoIP streams encoded by source codec," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 2, pp. 296–306, Jun. 2011.

[11] A. Desoky, "Comprehensive linguistic steganography survey," *Int. J. Inf., Comput. Secur.*, vol. 4, no. 2, pp. 164–197, 2010.

[12] J.-F. Couchot, R. Couturier, and C. Guyeux, "STABYLO: Steganography with adaptive, Bbs, and binary embedding at low cost," *Ann. Telecommun.-Ann. Télécommun.*, vol. 70, nos. 9–10, pp. 441–449, 2015.

[13] A. Majumder and S. Changder, "A novel approach for text steganography: Generating text summary using reflection symmetry," *Procedia Technol.*, vol. 10, no. 10, pp. 112–120, 2013.

[14] Y. Luo, Y. Huang, F. Li, and C. Chang, "Text steganography based on ci-poetry generation using Markov chain model," *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 9, pp. 4568–4584, 2016.

[15] S. Mahato, D. A. Khan, and D. K. Yadav, "A modified approach to data hiding in Microsoft word documents by change-tracking technique," *J. King Saud Univ. Comput. Inf. Sci.*, to be published.

[16] B. Murphy and C. Vogel, "The syntax of concealment: Reliable methods for plain text information hiding," *Proc. SPIE*, vol. 6505, p. 65050Y, Feb. 2007.

[17] X. Ge, R. Jiao, H. Tian, and J. Wang, "Research on information hiding," *US-China Edu. Rev.*, vol. 3, no. 5, pp. 77–81, 2006.

[18] Y. Luo and Y. Huang, "Text steganography with high embedding rate: Using recurrent neural networks to generate chinese classic poetry," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur.*, 2017, pp. 99–104.

[19] N. F. Johnson and P. A. Sallee, "Detection of hidden information, covert channels and information flows," in *Wiley Handbook of Science and Technology for Homeland Security*. 2008.

[20] W. Dai, Y. Yu, Y. Dai, and B. Deng, "Text steganography system using Markov chain source model and des algorithm," *J. Softw.*, vol. 5, no. 7, pp. 785–792, 2010.

[21] T. Fang, M. Jaggi, and K. Argyraki. (2017). "Generating steganographic text with LSTMs." [Online]. Available: https://arxiv.org/abs/1705.10742

[22] J. Zhang, J. Shen, L. Wang, and H. Lin, "Coverless text information hiding method based on the word rank map," in *Proc. Int. Conf. Cloud Comput. Secur.*, 2016, pp. 145–155.

[23] Z. Zhou, Y. Mu, and Q. J. Wu, "Coverless image steganography using partial-duplicate image retrieval," *Soft Comput.*, pp. 1–12, Mar. 2018.

[24] J. Zhang, Y. Xie, L. Wang, and H. Lin, "Coverless text information hiding method using the frequent words distance," in *Cloud Computing and Security*. Cham, Switzerland: Springer, 2017.

[25] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM Syst. J.*, vol. 35, nos. 3–4, pp. 313–336, 1996.

[26] U. Topkara, M. Topkara, and M. J. Atallah, "The hiding virtues of ambiguity: Quantifiably resilient watermarking of natural language text through synonym substitutions," in *Proc. 8th Workshop Multimedia Secur.*, 2006, pp. 164–174.

[27] M. H. Shirali-Shahreza and M. Shirali-Shahreza, "A new synonym text steganography," in *Proc. Int. Conf. Intell. Inf. Hiding Multimedia Signal Process. (IIHMSP)*, 2008, pp. 1524–1526.

[28] H. Z. Muhammad, S. M. S. A. A. Rahman, and A. Shakil, "Synonym based malay linguistic text steganography," in *Proc. Innov. Technol. Intell. Syst. Ind. Appl. (CITISIA)*, 2009, pp. 423–427.

[29] P. Wayner, "Mimic functions," *Cryptologia*, vol. 16, no. 3, pp. 193–214, 1992.

[30] M. Chapman and G. Davida, "Hiding the hidden: A software system for concealing ciphertext as innocuous text," in *Proc. Int. Conf. Inf. Commun. Secur.* Springer, 1997, pp. 335–345.

[31] H. H. Moraldo. (2014). "An approach for text steganography based on Markov chains." [Online]. Available: https://arxiv.org/abs/1409.0915

[32] W. Dai, Y. Yu, and B. Deng, "Bintext steganography based on Markov state transferring probability," in *Proc. 2nd Int. Conf. Interact. Sci., Inf. Technol.*, Culture Human, 2009, pp. 1306–1311.

[33] A. Shniperov and K. Nikitina, "A text steganography method based on Markov chains," *Autom. Control Comput. Sci.*, vol. 50, no. 8, pp. 802–808, 2016.

[34] Z. Yang, Y.-J. Zhang, S. Ur Rehman, and Y. Huang, "Image captioning with object detection and localization," in *Proc. Int. Conf. Image Graph.* Cham, Switzerland: Springer, 2017, pp. 109–118.

[35] D. Bahdanau, K. Cho, and Y. Bengio. (2014). "Neural machine translation by jointly learning to align and translate." [Online]. Available: https://arxiv.org/abs/1409.0473

[36] W. Feller, "Law of large numbers for identically distributed variables," in *An Introduction to Probability Theory and its Applications*, vol. 2, 2nd ed. New York, NY, USA: Wiley, 1971, pp. 231–234.

[37] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proc. Comput. Vis. Pattern Recognit.*, 2015, pp. 3156–3164.

[38] L. Shang, Z. Lu, and H. Li. (2015). "Neural responding machine for short-text conversation." [Online]. Available: https://arxiv.org/abs/1503.02364

[39] T. Mikolov, M. Karafiát, L. Burget, J. Cernock, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. INTERSPEECH*, Makuhari, Chiba, Japan, Sep. 2010, pp. 1045–1048.

[40] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 6, no. 2, pp. 107–116, 1998.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, vol. 8689, 2014, pp. 818–833.

[43] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Readings Cognit. Sci.*, vol. 323, no. 6088, pp. 399–421, 1988.

[44] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. Inst. Radio Eng.*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[45] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," CS224N Project, Stanford, CA, USA, Tech. Rep., 2009, vol. 1, no. 12.

[46] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, 2011, pp. 142–150.

[47] A. Thompson. *Kaggle*. Accessed: Aug. 20, 2017. [Online]. Available: https://www.kaggle.com/snapcrack/all-the-news/data

[48] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. OSDI*, vol. 16. 2016, pp. 265–283.

[49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[50] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

[51] D. Jurafsky, *Speech & Language Processing*. Chennai, India: Pearson Education India, 2000.

[52] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1188–1196.

[53] L. V. D. Maaten, "Accelerating t-SNE using tree-based algorithms," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3221–3245, Oct. 2014.

[54] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vis.*, vol. 40, no. 2, pp. 99–121, Nov. 2000.

[55] P. Meng, L. Hang, W. Yang, Z. Chen, and H. Zheng, "Linguistic steganography detection algorithm using statistical language model," in *Proc. Int. Conf. Inf. Technol. Comput. Sci.*, 2009, pp. 540–543.

[56] Z. Chen *et al.*, "Linguistic steganography detection using statistical characteristics of correlations between words," in *Proc. Int. Workshop Inf. Hiding*. Berlin, Germany: Springer, 2008, pp. 224–235.

[57] S. Samanta, S. Dutta, and G. Sanyal, "A real time text steganalysis by using statistical method," in *Proc. IEEE Int. Conf. Eng. Technol. (ICETECH)*, 2016, pp. 264–268.

[58] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. (2016). "Bag of tricks for efficient text classification." [Online]. Available: https://arxiv.org/abs/1607.01759

[59] T. Y. Liu and W. H. Tsai, "A new steganographic method for data hiding in Microsoft word documents by a change tracking technique," *IEEE Trans. Inf. Forensics Security*, vol. 2, no. 1, pp. 24–30, Mar. 2007.

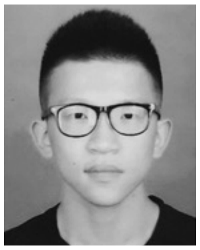
[60] R. Stutsman, C. Grothoff, M. Atallah, and K. Grothoff, "Lost in just the translation," in *Proc. ACM Symp. Appl. Comput.*, 2006, pp. 338–345.

[61] X. Chen, H. Sun, Y. Tobe, Z. Zhou, and X. Sun, "Coverless information hiding method based on the chinese mathematical expression," in *Proc. Int. Conf. Cloud Comput. Secur.* Cham, Switzerland: Springer, 2015, pp. 133–143.

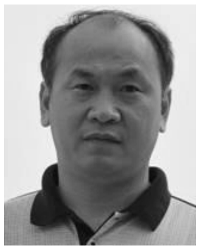
[62] Z. Zhou, Y. Mu, N. Zhao, Q. M. J. Wu, and C. N. Yang, *Coverless Information Hiding Method Based on Multi-Keywords*. Cham, Switzerland: Springer, 2016.

Zhong-Liang Yang received the B.S. degree from the School of Electronic Science and Technology, Sichuan University, in 2015. He is currently pursuing the Ph.D. degree with the Department of Electronic Engineering, Tsinghua University, Beijing. His research interests include information hiding and natural language process.

Xiao-Qing Guo received the B.Eng. degree from the School of Biological Science and Medical Engineering, Beihang University, Beijing, in 2018. She is currently pursuing the Ph.D. degree with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong. Her current research interests mainly focus on computer vision and natural language processing.



Zi-Ming Chen is currently pursuing the B.Eng. degree with the Beijing University of Posts and Telecommunications. He is expecting to receive his B.Eng. degree in 2019 and planning to pursue the M.S. degree in computer science further. His research interests now focus on natural language process and computer vision.



Yong-Feng Huang (SM'16) received the Ph.D. degree in computer science and engineering from the Huazhong University of Science and Technology in 2000. He is currently a Professor with the Department of Electronic Engineering, Tsinghua University. He has published six books and over 100 research papers on computer network, multimedia communications, and security. His research interests include multimedia network security, steganography and steganalysis, big data mining, and next-generation Internet.

Yu-Jin Zhang received the Ph.D. degree in applied science from the State University of Liège, Liège, Belgium, in 1989. In 1993, he joined the Department of Electronic Engineering, Tsinghua University, Beijing, China, where he has been a Professor of image engineering since 1997 and a Ph.D. Supervisor since 1998. From 1994 to 2003, he was the Deputy Director of the Institute of Image and Graphics, Tsinghua University. Since 2014, he has been a tenured Professor and Director of the Institute of Media Cognition and Intelligent System.