A Generation-based Text Steganography by Maintaining Consistency of Probability Distribution

Boya Yang[1], Wanli Peng[1], Yiming Xue[1*] and Ping Zhong[2]
[1] College of Information and Electrical Engineering, China Agricultural University,
Beijing, China 100083
[2] College of Science, China Agricultural University,
Beijing, China 100083
[e-mail: yangboya@cau.edu.cn, hunanpwl@cau.edu.cn, xueym@cau.edu.cn, zping@cau.edu.cn]
[*]Corresponding author: Yiming Xue

Received May 21, 2021; revised August 4, 2021; accepted Septebmer 16, 2021; published November 30, 2021

Abstract

Text steganography combined with natural language generation has become increasingly popular. The existing methods usually embed secret information in the generated word by controlling the sampling in the process of text generation. A candidate pool will be constructed by greedy strategy, and only the words with high probability will be encoded, which damages the statistical law of the texts and seriously affects the security of steganography. In order to reduce the influence of the candidate pool on the statistical imperceptibility of steganography, we propose a steganography method based on a new sampling strategy. Instead of just consisting of words with high probability, we select words with relatively small difference from the actual sample of the language model to build a candidate pool, thus keeping consistency with the probability distribution of the language model. What's more, we encode the candidate words according to their probability similarity with the target word, which can further maintain the probability distribution. Experimental results show that the proposed method can outperform the state-of-the-art steganographic methods in terms of security performance.

Keywords: Steganography, Steganalysis, Linguistic Steganography, Probability Distribution, Imperceptibility.

1. Introduction

Steganography is a subject that studies how to embed secret information into general media carriers, such as image[1][2], video[3][4], text[5][6][28][29]. Since text is widely used in daily life, text steganography has attracted more and more attention. According to the way of obtaining the carrier, the text steganography methods can be divided into two types: modification-based steganography with carrier and generation-based steganography without carrier[7]. It turns out that the modification-based approach is easy to be detected because of its explicit changes, and they cannot get fluent steganographic texts at scale[8]. In order to overcome this shortcoming, more attention has been paid to text steganography based on generation.

Generation-based steganographic methods do not have a pre-set text carrier, but use a language model to generate steganographic text directly driven by secret information. The early methods are mainly based on a special syntax structure. Wayner[9] firstly proposed a mimic function to construct a Huffman tree to learn the statistical distribution of each character. Chapman et al.[10] tried to use a syntactic template or syntax structure tree which matches the grammatical rules to generate texts. However, due to the single and fixed template, these methods can only match the grammar rules, but are hard to maintain semantic consistency.

In order to improve the quality of steganographic text, many researchers introduce natural language generation technology and embed secret information implicitly in the process of text generation. The goal of text generation is to predict the next word in a sentence using a language model, and then output an appropriate word continuously, so as to generate a sentence. In order to map the secret information into generated words, in the process of text generation, some candidate words are selected and encoded. Under the control of a specific embedding algorithm, the language model will output the corresponding words, which carry the secret information. Markov chain is one of the earliest algorithms for text generation. A transition matrix is calculated to predict the next word, and candidate words are encoded to embed the secret information[11][12]. In order to overcome the limitations of Markov chain, Fang et al.[13] used a Long Short-Term Memory (LSTM) with fixed coding to generate steganographic sentences. The good performance of LSTM in the sequence modeling task greatly improves the quality of generated steganographic text. Yang et al.[14] adopted an LSTM with dynamic coding based on conditional probability. Later, Yang et al.[15] applied a more advanced framework based on variational auto-encoder to better model the training corpus. By using a powerful language model, these methods can generate steganographic texts with high quality.

Even though a better language model is introduced, the distortion of probability distribution caused by data embedding cannot be avoided, and the steganographic text is easy to be detected statistically. In order to consider more about the statistical characteristics of the language model, some improved steganography algorithms are proposed. Dai et al.[16] measured the consistency of probability distributions, and embedded information only when it meets the condition. Ziegler et al.[17] used arithmetic coding to embed secret information, which effectively keeps more accurate conditional probability of words. These algorithms further reduce the distortion of the probability distribution of words in the generated steganographic texts by using better coding algorithms. However, they all use a greedy strategy to build a candidate pool, and the change of probability distribution caused by sampling always exists. Specifically, at each time step, the language model only outputs one of the top k words with the largest conditional probability. Especially in the case of a low embedding rate, there are few candidate words in the candidate pool, and the steganography

algorithm almost always generates the same words. This method reduces the diversity of generated tokens, destroys the overall probability distribution of the generated texts, brings the risk of being detected statistically, and inevitably leads to security issues. Therefore, it is a great challenge to design a high secure steganography method which can keep the probability distribution of the language model.

In this paper, we propose a new text steganography algorithm based on conditional probability difference (PD-Stega). Instead of using greedy strategy, the proposed method selects words with relatively small difference from the actual sample of normal text generation to build a candidate pool. What's more, the method encodes the candidate words according to their probability similarity with the target word. In this way, the proposed method can further maintain the probability distribution of the language model. Different from the existing methods to enhance security by using better coding algorithms, the proposed method can apply different coding algorithms to complete the embedding of secret information. Experimental results show that the proposed method achieves good performance in both statistical imperceptibility and subjective imperceptibility, which shows the strong anti-steganalysis ability and good text quality.

In the Section 2, related work and knowledge background are introduced. In Section 3, the process of secret information embedding is shown. In Section 4, the process of secret information extraction is introduced. In Section 5, we will show the experimental setup and the performance of the proposed steganography method. Section 6 summarizes the conclusions.

2. Related work

This section discusses existing methods to provide foundational knowledge regarding our design.

2.1 Sample-controlled text steganography

Generation-based text steganography has attracted researchers' attention for its high embedding rate and robustness. With the rapid development of text generation technology, deep neural networks are used to capture longer and higher-dimensional text features, resulting in sentences that are sufficiently coherent and semantically complete. In recent years, more and more text steganography methods generate steganographic text based on neural network models[13][14][15][16][17].

A general text generation process can be explained as follows. Given a well-trained language model and a text fragment, new text can be generated by repeating the following operations, which is shown in **Fig. 1**: (1) Input the existing sequence into the model, and obtain the conditional probability distribution of the next word from the model. (2) Sample the next word according to the distribution and certain rules. (3) Add the selected word to the end of the sequence.

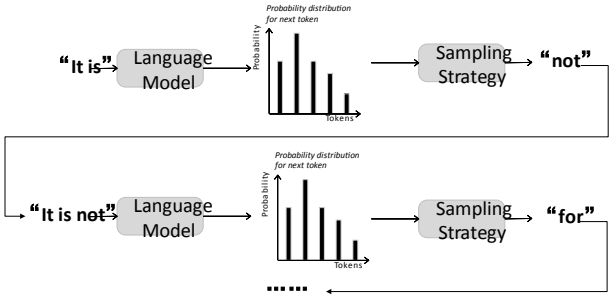

Fig. 1. The process of text generation by a language model.

Text steganography based on generation is driven by secret information. Obviously, for text generation, due to the diversity of text context and background, any sequence is possible, although their probability is different, which is determined by the conditional probability of each word in sequence. Under this premise, we can control the word selection at each moment, and embed the corresponding secret bits by outputting different words. Therefore, in the process of steganographic generation, the language model generates sentences containing information under the control of secret bits. Some algorithms will be adopted to realize the mapping between secret information and tokens. This framework embeds information by controlling the sampling of the language model, so we can summarize it as **sample-controlled steganography**.

2.2 Greedy Strategy for Building Candidate Pools

In the steganographic generation, for the purpose of embedding a certain amount of secret information, recent works usually select multiple words to build a candidate pool (CP) at each time step of generation, and then encode the words in it appropriately. Finally, the secret information can be embedded by outputting the corresponding word.

When constructing candidate pool, researchers usually adopt **greedy strategy**, that is, to select the word with the highest conditional probability each time. This strategy always samples the most likely words, which leads to the problem of text degradation. In addition, since each token to be generated depends on the recently generated text, the error is amplified. Finally, it leads to repetitive and predictable strings, and the overall probability distribution of the steganographic text changes greatly.

Many researchers use a similar framework to embed secret information. Yang et al.[14] proposed a method to encode the conditional probability of words in the candidate pool by using a full binary tree or Huffman tree at every time. However, the greedy strategy of constructing candidate pool distorts the probability distribution of generated steganographic texts. Dai et al. [16] did not change the construction method of CP, but proposed an improved coding algorithm, called patient-Huffman. The algorithm calculates the change of dictionary distribution during the steganographic generation, and only when the threshold is met can the information be embedded. They improved Yang's algorithm, but resulted in a lower embedding rate. Ziegler et al. [17] used arithmetic coding to keep more probability distribution. However, arithmetic coding only works in the candidate pool. The greedy strategy has not been changed, and the preference for high probability words still exists.

According to the above analysis, we come to the conclusion: in the construction of candidate pool, the application of greedy strategy will bring great probability distortion to the steganographic text, resulting in security risks. With the development of natural language generation technology, it is no longer a problem to generate enough natural steganographic

text satisfying subjective imperceptibility. How to minimize the influence of steganographic sampling on the probability distribution of language model and make steganography have high statistical imperceptibility has become a crucial issue.

3. The Proposed Steganography Algorithm

This section gives a general description of the proposed text steganography algorithm based on conditional probability difference (PD-Stega), and then explains how it works.

Most of sample-controlled steganography constructs a candidate pool based on greedy strategy and encodes candidate words according to the conditional probability. This pattern will reduce the diversity of generated text and change the probability distribution. Based on this, we propose a new method to construct the candidate pool according to probability difference and encode reasonably.

The basic idea of our algorithm is to make the steganographic output follow the real output of the language model. We construct the candidate pool with minimum probability distortion, and encode the candidate words according to the probability similarity, so as to keep the probability distribution of the language model. **Fig. 2** shows the overall framework of the algorithm. The information hiding process includes the training of the language model, the construction of the candidate pool and the embedding of secret information. We will introduce these three modules in turn.

3.1 Steganographic Generator

Firstly, a well-trained language model is applied as a generator. In this paper, we conduct the proposed algorithm on a 3-layer LSTM. After the word vector is input, the language model predicts the next token, calculates the probability distribution, and outputs the optimal word at the current time according to the probability.

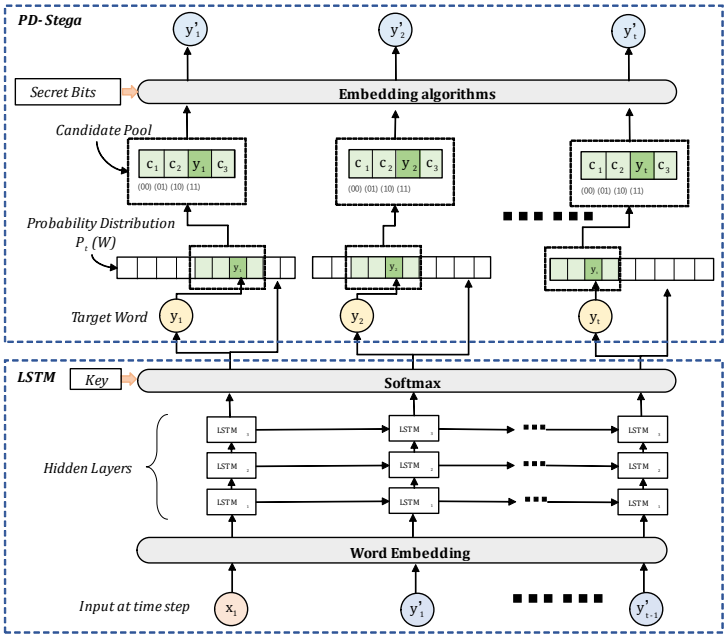

Fig. 2. The overall schematic diagram of steganographic generation.

We denote the conditional probability distribution of the dictionary W at time step t as $P_t(W)$, which is obtained from LSTM. At each time step, the hidden layer h_t of LSTM is normalized by the softmax function to obtain the predictive probability of each token in the dictionary W. It can be denoted as:

$$\mathrm{P_t}(W) = softmax_{LSTM}(h_t, \theta), \tag{1}$$

where θ is the correction parameter when calculating the probability distribution, such as temperature coefficient. h_t is the hidden layer state of LSTM at time step t, which can be denoted as:

$$h_t = f_{LSTM}(s_t \,|s_1, s_2, \dots, s_{t-1}),$$

Where $\{s_1, s_2, \dots, s_{t-1}\}$ is the previous generated token sequence. In the normal sampling process of text generation, we sample according to probability distribution $P_t(W)$ to get the next token y_t.

$$y_t \sim \mathrm{P_t}(W). \tag{2}$$

At the beginning of the generation, we execute the generator twice without data embedding, and obtain two words as the prefix, which can prevent the same sentence from being generated when the secret information is repeated.

3.2 Construction of Candidate Pool

Secondly, at each time step of steganographic generation, we obtain the current output of language model as the target word and implement PD-Stega algorithm to construct a candidate pool, which conforms to the distribution of the language model and minimizes probability distortion.

Specifically, in the process of steganographic generation, at time step t, we firstly input the current prefix into the language model and obtain the output of the language model as the target word, that is,

$$target\ word = y_t, and\ y_t \sim P_t(W), when\ t > 2. \tag{3}$$

Next, according to the conditional probability calculated by the language model (denoted as $p(*)$), the probability difference between each word and the $target\ word$ is calculated. For the word d in the dictionary, the probability difference between d and the $target\ word$ is denoted as $ProbDiff(d)$,

$$ProbDiff(d) = |p(d) - p(target\ word)|. \tag{4}$$

Then, we sort the dictionary in ascending order according to the $ProbDiff(d)$. When the size of the candidate pool is k, we take the first k sorted words to construct the candidate pool. In this case, the probabilities of candidate words are relatively close to the target probability, and the candidate pool (CP) is represented as:

$$CP = \{target\ word, c_2, \dots, c_k\} \tag{5}$$

where $0 = ProbDiff(target\ word) < ProbDiff(c_2) < ProbDiff(c_3) < \cdots$

By approximating the real sampling probability of the language model, we get a candidate pool with less probability distortion.

3.3 The PD-based coding algorithm

After constructing the candidate pool, previous methods usually perform entropy coding on the words in the candidate pool according to their conditional probability. However, this method still guarantees the priority of high probability words in the candidate pool, rather than maintaining the priority of actual sampling, which further makes the probability distribution of steganographic text incline to high probability sequences. As shown in **Fig. 3 (a)**, the darker the color, the higher the sampling probability.

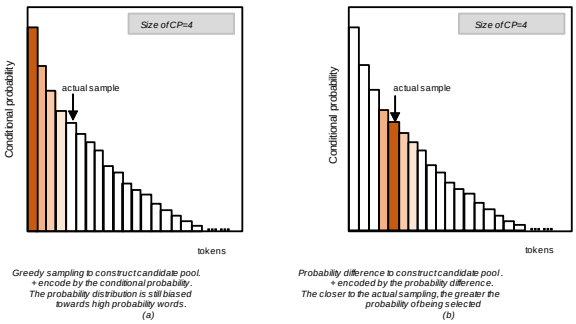

Fig. 3. Sampling probability under different candidate pool construction methods and coding methods.

In order to keep the sampling probability of the language model as much as possible even in the candidate pool, we calculate the probability similarity between each candidate word c_i and the target word, as calculated in (6).

$$ProbSim(c_i, target\ word) = 1 - ProbDiff(c_i), c_i \in CP. \tag{6}$$

After that, we can encode the candidate words according to the calculated value of the probability similarity. In this way, we can further eliminate the preference for high probability words in the candidate pool, but follow the actual sampling of the language model. As shown in **Fig. 3 (b)**, the closer the candidate word is to the actual sampling of the language model, the higher the probability of being selected. The usual steganography algorithm encodes candidate words according to the conditional probability. Compared with it, the proposed method is more consistent with the sampling process of the normal text generation process.

After normalizing the probability similarity, the coding algorithm is implemented, and each word in the candidate pool can be mapped to a bit sequence using an embedding algorithm. In this paper, we adopt Huffman coding[14] and arithmetic coding[17] to embed the secret information (denoted as PD-HC and PD-AC respectively). The corresponding coding processes are shown in **Fig. 4**, respectively.

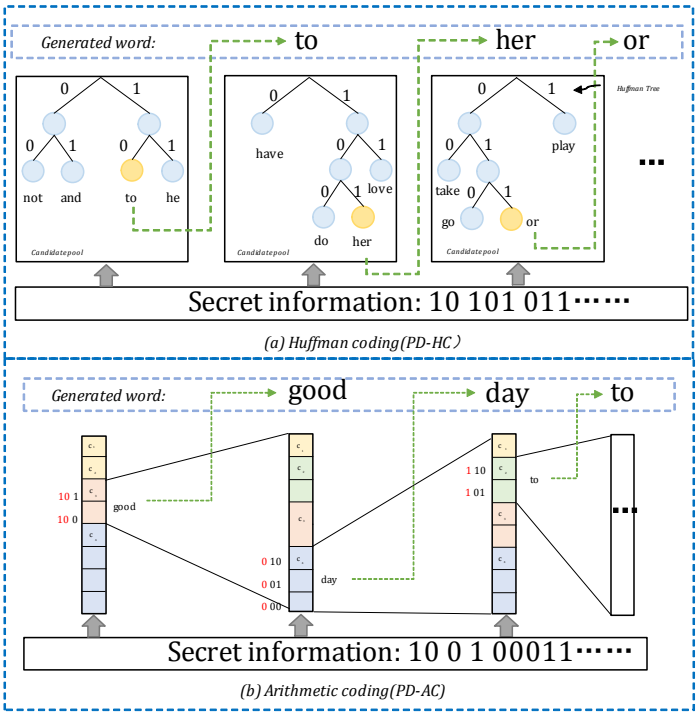

Fig. 4. The embedding process with two embedding algorithms.

When Huffman coding is used, a Huffman tree is constructed for the words in candidate pool according to their probability similarity calculated in (5), and each leaf node corresponds to a fixed sequence. We read in the secret information bits in turn, and then search from the root node. If the current bit is "0", we turn to the left subtree, otherwise turn to the right subtree until we reach a leaf node. At this time, the leaf node is added to the prefix as the current output, and the bit string corresponding to the retrieval path is the secret information carried by the word. The above process is repeated until the end character is generated, or the prefix length reaches the preset length. When using arithmetic coding, we first normalize the probability similarity of words in candidate pool at the current moment, then find the appropriate probability interval according to the read secret information bits, and output the corresponding word. Through this algorithm, the secret information is embedded into the text for transmission, and the statistical imperceptibility of the steganographic text is guaranteed. The specific algorithm program is shown in **Algorithm 1**.

Algorithm 1: Secret Information Embedding

Input:

secret bit string B = {1, 0, 0, ...1}

size of candidate pool: n

coding algorithm: G

start pairs list: T = {$start_1$, $start_2$, $start_3$...}

Output:

output generated steganographic sentences list: S

```
Data preprocess and train an LSTM as the generator
Generate two words randomly according to the language model as a prefix
while not the end of B:
  if there is no terminator in the current prefix:
    Feed the current prefix into the generator;
    Get the actual sample of language model as the target word according to the key;
    Calculate the conditional probability difference between each word in the dictionary and the target word;
    Descending the dictionary according to the probability difference, and select the top n sorted words to build a candidate pool (CP);
    Calculate the probability similarity of candidate words, and encode them according to their normalized probability similarity with coding algorithm G;
    Read in the secret bit in B and add the corresponding words in the candidate pool into prefix;
  else:
    Add the current prefix as a sentence into S, and select a start pairs from T as prefix randomly;
if there is no terminator in the current prefix:
  Construct a Candidate Pool and output a word outside the Candidate Pool;
Return Generated sentences list S.
```

4. Secret information extraction algorithm

At the receiving end, the receiver first obtains the delivered prefix, and then uses the same trained language model to execute the generating program. The receiver constructs the candidate pool by the same strategy, and extracts the secret information from the received text through the same sampling algorithm. This is a realizable operation, because the random sampling performed by the computer is based on a pseudo-random sequence. As long as the sender and receiver share the same random seed, the construction of candidate pool can be completely reproduced. The specific algorithm program is shown in **Algorithm 2**.

Algorithm 2: Secret Information Extraction

```
Input:
  steganographic sentences: S = {S1, S2…, Sn}
  steganographic generator: LM
  size of candidate pool: n
  coding algorithm: G
Output:
  Secret bits string: B
for s in S do:
  Feed the first two words of the s into the generator;
  While not the end of s do:
    Calculate the conditional probability of the next word according to the previously words using the LM;
    Select a word as the target word according to the key;
    Calculate the probability difference between each word in the dictionary and the target word;
```

```
        Descending the dictionary according to the probability difference and select
        the top n sorted words to build a candidate pool (CP);
        Calculate the probability similarity of candidate words, and encode them
        according to their normalized probability similarity with coding algorithm G;
        if current word of s in CP do:
            add the corresponding bit string into B;
        else:
            End the process and return.
    Return secret string B
```

5. Experimental Results

In this section, we compare our method with two related studies of generation-based text steganography, RNN-Stega[14] and AC-Stega[17]. Both of the two methods build candidate pools by greedy strategy, and they apply Huffman coding and arithmetic coding respectively.

5.1 Experimental Setup

The experiments are conducted on three large text datasets: Twitter[18], Movie reviews [19], and News[20]. These three corpora contain texts of different lengths and grammatical complexity, which can ensure the effectiveness of the experiments. All the cover texts are preprocessed, including word segmentation, special symbol replacement and low-frequency word removal. Then we randomly selected 100,000 sentences from Twitter and News, 150,000 sentences from Movie reviews to form multiple training datasets. The details of the final training datasets are shown in **Table 1**.

Table 1. The details of the training datasets.

Dataset	Twitter[18]	Movie review[19]	News[20]
Average Length	6.73	14.56	16.24
Words Num	7592	16125	24116

We use a 3-layer LSTM network to learn from the corpora and generate sentences, in which the size of the hidden layer is 800. The LSTM is performed on Python and TensorFlow framework. Moreover, we set the dimension of the word vector to 300 and train the model by using Adam[22] optimizer with a learning rate of 0.001. The well-trained LSTM is used to implement different steganography algorithms and generate several steganographic sentences.

In order to verify the performance of the algorithm, we analyze the generated steganographic text from the perspective of text quality and statistics. We also mix the steganographic sentences with the sentences extracted from the corpus (which do not intersect with the training dataset). Then we use the steganalysis model to identify whether the sentences are steganographic sentences, so as to evaluate the security of steganography.

5.2 Steganographic Text Quality

In this experiment, we aim to evaluate whether the steganographic texts generated by the proposed method are similar to the normal texts.

As proved in [15], natural language is produced in different knowledge backgrounds and language environments, and the probability of natural language sentences has a large variance. For different corpora, the probability distribution of sentences is different. We calculated

$perplexity$ (ppl)[23] of three corpora ($perplexity$, a common metric to measure the probability of a sentence in the field of natural language processing). The ppl of Twitter, Movie, and News are 101.88, 134.98, and 219.25 respectively. It can be seen that the probability distribution of different types of the corpus is very different.

In order to measure the similarity of steganographic texts and normal texts more comprehensively, we refer to Yang's[15] practice and calculate the difference between their mean ppl values. The difference is denoted as ΔMP:

$$\Delta MP = \left|mean\left(ppl_{Stega}\right) - mean(ppl_{Normal})\right| \tag{6}$$

While $mean\left(ppl_{Stega}\right)$ and $mean(ppl_{Normal})$ are the average perplexity values of 100,000 steganographic sentences and normal sentences, respectively. Obviously, the smaller the ΔMP is, the more similar the steganographic text is to the normal text. We calculate ΔMP of generated steganographic sentences by each experiment, and the results are shown in **Table 2**.

Table 2. The ΔMP values of steganographic texts generated by different algorithms under different embedding rate.

	Size of CP	**2**	**4**	**8**	**16**
Twitter[18]	RNN-Stega(HC)	77.56	76.39	70.13	62.78
	PD-HC	**19.31**	**7.91**	**9.00**	**37.12**
	AC-Stega(AC)	83.51	81.17	76.88	71.04
	PD-AC	**40.84**	**34.00**	**18.79**	**5.16**
Movie review[19]	RNN-Stega(HC)	114.33	110.53	105.11	96.64
	PD-HC	**14.19**	**1.77**	**30.57**	**84.13**
	AC-Stega(AC)	117.40	114.85	110.47	104.17
	PD-AC	**52.77**	**43.71**	**19.24**	**30.19**
News[20]	RNN-Stega(HC)	188.63	184.22	177.03	167.21
	PD-HC	**58.09**	**38.28**	**6.77**	**58.03**
	AC-Stega(AC)	197.81	193.43	187.05	179.15
	PD-AC	**109.76**	**94.76**	**68.63**	**14.54**

From the table, we can draw the following conclusions: When using the same embedding algorithm (such as RNN-Stega and PD-HC, AC-Stega and PD-AC), the ΔMP of steganographic sentences generated by the proposed method is smaller. This is because our algorithm builds candidate pool based on the actual sampling of the language model and encodes them according to probability similarity, making the steganographic sampling as consistent as possible with the sampling process of normal text generation. Therefore, the generated text is more similar to the normal text in the training set. However, both RNN-Stega and AC-Stega use greedy strategy to construct candidate pools, so the generator always outputs words with high probability at each time step of the steganographic generation process. The sampling range of the language model is reduced, leading to large deviation.

5.3 Consistency of Statistical Distribution

In this section, we will verify the statistical security of the algorithms.

5.3.1 The consistency of probability distributions

Kullback-Leibler divergence (D_{KL})[21] is a metric to measure the similarity of two probability distributions, which quantifies the similarity between texts statistically. We calculate the D_{KL}

between overall distributions of steganographic texts p and that of normal texts q. Kullback-Leibler divergence (D_{KL}) is defined as:

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i)\big(logp(x_i) - logq(x_i)\big), \tag{7}$$

the smaller the D_{KL} is, the closer the two probability distributions are.

When calculating the probability distribution of the text set, we use the statistical frequency of each word to represent its probability. We have carried out experiments on three corpora. By changing the size of candidate pool, we generate steganographic texts with different embedding rates. The D_{KL} values of different steganographic texts are shown in **Table 3.** In particular, in order to better present the above numerical results, for twitter corpus, the trend of D_{KL} is shown in **Fig. 5**. For a certain size of candidate pool, the amount of secret information carried by each generated word is not fixed, so in the **Fig. 5**, we calculate the average number of bits embedded in the generated text, which is indicated by bpw.

Table 3. The D_{KL} values of various generated steganographic texts.

	Size of CP	**2**	**4**	**8**	**16**
Twitter[18]	RNN-Stega(HC)	0.7596	0.6722	0.4676	0.3181
	PD-HC	**0.2488**	**0.2418**	**0.2501**	**0.2494**
	AC-Stega(AC)	0.9750	0.7802	0.6575	0.5271
	PD-AC	**0.2899**	**0.2974**	**0.2947**	**0.2837**
Movie review[19]	RNN-Stega(HC)	1.0115	0.8217	0.4921	0.3158
	PD-HC	**0.1800**	**0.1885**	**0.1926**	**0.1946**
	AC-Stega(AC)	1.274	0.9416	0.7417	0.5771
	PD-AC	**0.2340**	**0.2239**	**0.2263**	**0.2251**
News[20]	RNN-Stega(HC)	1.1814	0.9662	0.8329	0.6683
	PD-HC	**0.2301**	**0.2297**	**0.2331**	**0.2406**
	AC-Stega(AC)	1.4496	1.1566	0.9364	0.7470
	PD-AC	**0.2930**	**0.2773**	**0.2801**	**0.2766**

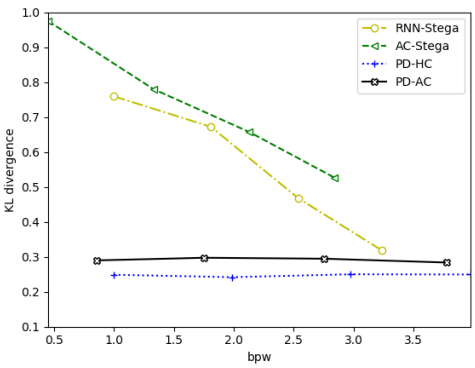

Fig. 5. D_{KL} of steganographic texts generated by different methods on Twitter.

It can be seen from the table and the figure that when using the same coding algorithm, the D_{KL} values of the texts generated by the proposed method are lower. What's more, as the number of secret bits in the generated words increases, the D_{KL} values change little. This demonstrates that changing the construction method of candidate pool and the coding mode can effectively maintain the consistency of probability distribution.

5.3.2 Diversity of Steganographic texts

To verify whether the algorithm can generate diverse text, we count the non-duplicate words in the steganographic text set and denote the total number as $unique$, which can reflect the lexical diversity of the text. We also calculate the $self\text{-}BLEU$ [24]value of steganographic text set, which is used to measure the similarity between each sentence with other sentences in the set. The average precisions of bi-grams are denoted as $self\text{-}BLEU2$, and those of tri-grams are denoted as $self\text{-}BLEU3$.

We count and calculate 10,000 steganographic sentences, and the results are shown in **Table 5**.

Table 5. Statistical results of different steganographic texts at lexical level.

	Size of CP	**2**	**4**	**8**	**16**
Twitter[18]					
RNN-Stega(HC)	self-BLEU2	0.9581	0.9535	0.9451	0.9345
	self-BLEU3	0.8848	0.8606	0.8212	0.7688
	unique	1085	1309	1385	1434
PD-HC	self-BLEU2	0.8831	0.8736	0.8691	0.8610
	self-BLEU3	0.6083	0.5780	0.5433	0.4982
	unique	**2170**	**2074**	**2069**	**2024**
AC-Stega(AC)	self-BLEU2	0.9720	0.9661	0.9607	0.9510
	self-BLEU3	0.9107	0.8902	0.8623	0.8189
	unique	962	1283	1500	1615
PD-AC	self-BLEU2	0.9017	0.9006	0.8928	0.8853
	self-BLEU3	0.6658	0.6510	0.6103	0.5596
	unique	**2027**	**2029**	**2014**	**1870**
Movie review[19]					
RNN-Stega(HC)	self-BLEU2	0.9717	0.9649	0.9553	0.9445
	self-BLEU3	0.9388	0.9120	0.8775	0.8322
	unique	1587	1882	2074	2352
PD-HC	self-BLEU2	0.8568	0.8549	0.8503	0.8424
	self-BLEU3	0.5722	0.5527	0.5184	0.4615
	unique	**4004**	**3841**	**3793**	**3695**
AC-Stega(AC)	self-BLEU2	0.9769	0.9724	0.9644	0.9533
	self-BLEU3	0.9545	0.9351	0.9041	0.8636
	unique	1388	1629	1945	2202
PD-AC	self-BLEU2	0.8857	0.8854	0.8812	0.8700
	self-BLEU3	0.6526	0.6407	0.6044	0.5359
	unique	**3715**	**3547**	**3531**	**3493**
News[20]					
RNN-Stega(HC)	self-BLEU2	0.9668	0.9622	0.9531	0.9418
	self-BLEU3	0.9253	0.8980	0.8651	0.8143
	unique	2073	2413	2697	2938
PD-HC	self-BLEU2	0.8482	0.8452	0.8416	0.8286
	self-BLEU3	0.5421	0.5204	0.4902	0.4335

	unique	**4697**	**4649**	**4545**	**4524**
AC-Stega(AC)	self-BLEU2	0.9762	0.9699	0.9617	0.9512
	self-BLEU3	0.9462	0.9243	0.8906	0.8493
	unique	1619	2102	2489	2837
PD-AC	self-BLEU2	0.8777	0.8760	0.8700	0.8595
	self-BLEU3	0.6283	0.6031	0.5716	0.5102
	unique	**4489**	**4451**	**4291**	**4248**

It can be concluded from the table that: (1) Texts generated by PD-HC and PD-AC have the maximum number of non-duplicate words, and lower value of $self\text{-}BLEU$, which means that more new sequences are generated, rather than always generating repetitive and predictable ones. This is because the proposed method does not adopt greedy strategy to build candidate pool, so the output words are more diverse. (2) For PD-HC and PD-AC, with the increase of the embedding rate, the number of non-duplicate words and value of $self\text{-}BLEU$ change little, and can always ensure the diversity of text content.

5.4 Anti-steganalysis Ability

In order to further measure the security of different steganographic algorithms, we use two latest steganalysis methods: LS-CNN[25] and TS-BiRNN[26] to detect the steganographic texts generated by RNN-Stega[14], AC-Stega[17], PD-HC, and PD-AC respectively. Firstly, classified sentences are generated from different corpora under different embedding rates. For each group of experiments, we select 10,000 steganographic texts, and extract the same number of sentences from normal texts, and mix them as training samples. Another 1,000 pairs of sentences are selected as test samples. After that, the trained steganalyzers are used to model and analyze the mixed texts. The steganalysis model judges whether a sentence contains secret information, and the performance is represented by the average detection accuracy. The lower the detection accuracy, the better the security of steganography.

Table 6 records the detection accuracy of two steganalysis methods. $Accuracy$ refers to the ratio of the total number of correctly classified texts to the total number of test samples.

Table 6. The accuracy of staganalyzers when detecting various steganographic texts.

	Size of CP	**2**	**4**	**8**	**16**
Twitter[18]					
RNN-Stega(HC)	TS-BiRNN	0.9292	0.9048	0.8665	0.8562
	LS-CNN	0.9405	0.9237	0.8907	0.8655
PD-HC	TS-BiRNN	**0.7768**	**0.7815**	**0.7953**	**0.8038**
	LS-CNN	**0.7828**	**0.7945**	**0.8088**	**0.8368**
AC-Stega(AC)	TS-BiRNN	0.9473	0.9237	0.8912	0.8712
	LS-CNN	0.954	0.9285	0.9022	0.883
PD-AC	TS-BiRNN	**0.8132**	**0.8012**	**0.8038**	**0.8142**
	LS-CNN	**0.8173**	**0.8072**	**0.8167**	**0.8263**
Movie review[19]					
RNN-Stega(HC)	TS-BiRNN	0.9677	0.9582	0.9345	0.9283
	LS-CNN	0.9728	0.962	0.951	0.9413
PD-HC	TS-BiRNN	**0.7739**	**0.7896**	**0.8339**	**0.8577**
	LS-CNN	**0.7936**	**0.8253**	**0.8472**	**0.8797**
AC-Stega(AC)	TS-BiRNN	0.9742	0.9572	0.9497	0.915

	LS-CNN	0.9793	0.9617	0.9555	0.9328
PD-AC	TS-BiRNN	**0.815**	**0.8038**	**0.8192**	**0.857**
	LS-CNN	**0.8135**	**0.8378**	**0.8587**	**0.866**
News[20]					
RNN-Stega(HC)	TS-BiRNN	0.9725	0.9663	0.9438	0.9353
	LS-CNN	0.9775	0.9696	0.9512	0.9400
PD-HC	TS-BiRNN	**0.8517**	**0.8435**	**0.8411**	**0.8566**
	LS-CNN	**0.8332**	**0.8492**	**0.8802**	**0.8837**
AC-Stega(AC)	TS-BiRNN	0.9817	0.9733	0.9560	0.9428
	LS-CNN	0.9843	0.9808	0.9620	0.9513
PD-AC	TS-BiRNN	**0.8665**	**0.8688**	**0.8535**	**0.8473**
	LS-CNN	**0.8788**	**0.8383**	**0.8783**	**0.8907**

From **Table 6**, we can see that the detection rate of the proposed method is lower than that of other methods. The main reason is that our method makes the steganographic output follow the actual sampling as much as possible. Therefore, the generated steganographic sentences are more consistent with the language model, resulting in a significant reduction in detection accuracy. Experimental results show that the proposed method has strong anti-steganalysis ability no matter what embedding algorithm is adopted.

5.5 Time efficiency of the algorithm

In order to evaluate the time efficiency of the proposed algorithm, we calculated the average generation time of different steganography algorithms under the same experimental environment.

Specifically, for different algorithms (RNN-Stega[14], AC-Stega[17], PD-HC, PD-AC) and different candidate pool sizes, we respectively generate 1000 steganographic sentences and calculate the average running time. We have conducted experiments on Twitter and Movie review datasets. The results are shown in **Table 7**.

It can be seen from the table that when Huffman coding is used, the time taken by the proposed algorithm is significantly reduced. This is because the time efficiency of the Huffman coding will be affected by the distribution of tokens in the candidate pool. The proposed algorithm constructs a candidate pool with the smallest difference in probability. Therefore, the constructed Huffman tree is the closest to a balanced binary tree, and the average search length of the leaf nodes is the shortest, that is, the coding time is the shortest. On the other hand, as the candidate pool expands, the Huffman tree becomes deeper, resulting in an increase in the average time to search for leaf nodes, and therefore an increase in text generation time. For arithmetic coding, the time efficiency of the proposed algorithm is comparable to that of AC-Stega[17]. This is because the variance of the probability value does not affect the coding efficiency.

Table 7. Average time to generate a steganographic sentence(seconds).

Size of CP	**2**	**4**	**8**	**16**
Twitter[18]				
RNN-Stega(HC)	0.5020	0.8294	1.5081	2.8344
PD-HC	0.2693	0.2758	0.2855	0.2908
AC-Stega(AC)	0.2202	0.2202	0.2175	0.2195

PD-AC	0.2613	0.2617	0.2639	0.2660
Movie review[19]				
RNN-Stega(HC)	1.0772	1.7378	3.0410	5.7191
PD-HC	0.4661	0.4676	0.4740	0.4832
AC-Stega(AC)	0.3897	0.3846	0.3874	0.3872
PD-AC	0.4613	0.4647	0.4624	0.4608

5.6 Overall performance of steganography

In the previous experiments, we verified that the change of the candidate pool's construction can improve the security of the steganography algorithm. In order to evaluate the proposed algorithm more comprehensively, in this section, we test and compare representative generative steganography systems.

The steganography systems we have tested are as follows: LSTM-based block coding algorithm[13] (denoted as LSTM-BLOCK), LSTM-based Huffman coding algorithm[14] (denoted as LSTM-HC), and the proposed LSTM-based PD-HC algorithm (denoted as LSTM-PD-HC), the Huffman coding algorithm based on the more advanced language model VAE[15] (denoted as VAE-HC). In addition, VAE-HC is composed of BERT[] and LSTM modules. We have implemented these algorithms on Twitter, and the experimental results are shown in **Table 8**.

We use a variety of indicators to evaluate their security performance, including text quality (smaller ΔMP represents better text quality), consistency of statistical attributes (smaller D_{KL} represents higher consistency), and anti-steganalysis ability (the lower the detection accuracy rate, the higher the security). In addition, for variable-length coding algorithms, we select steganographic texts with similar average embedding rates for comparison.

From the table, we can see that our algorithm can obtain better performance by changing the method of constructing candidate pools, including improved text quality, more consistent statistical attributes, and stronger anti-steganalysis abilities. The high security of the proposed algorithm is proved.

Table 8. Security performance of multiple steganography systems on Twitter.

Twitter[18]					
LSTM-BLOCK [13]	bpw	1.0	2.0	3.0	4.0
	ΔMP	52.4776	29.4782	10.8375	**24.5312**
	D_{KL}	0.3232	0.3779	0.5018	0.5784
	TS-BiRNN	0.8083	0.8625	0.9265	0.9653
	LS-CNN	0.8178	0.8785	0.9553	0.9715
LSTM-HC[14]	bpw	1.0	1.81	2.54	3.24
	ΔMP	77.555	76.3850	70.1250	62.7750
	D_{KL}	0.7596	0.6722	0.4676	0.3181
	TS-BiRNN	0.9292	0.9048	0.8665	0.8562
	LS-CNN	0.9405	0.9237	0.8907	0.8655
VAE-HC[15]	bpw	1.0	1.9319	2.7373	4.16
	ΔMP	82.9007	78.6959	73.8152	57.3863
	D_{KL}	0.8460	0.6768	0.5485	0.3591
	TS-BiRNN	0.9370	0.8837	0.844	0.8007

	LS-CNN	0.9502	0.9192	0.881	**0.8053**
LSTM-PD-HC (the proposed)	bpw	1.0000	1.9867	2.9757	3.9778
	ΔMP	**19.3108**	**7.9064**	**9.0023**	37.1229
	D_{KL}	**0.2488**	**0.2418**	**0.2501**	**0.2494**
	TS-BiRNN	**0.7768**	**0.7815**	**0.7953**	**0.8038**
	LS-CNN	**0.7828**	**0.7945**	**0.8088**	0.8368

6. Conclusion

Generation-based text steganography mainly embeds secret information by controlling the sampling of text generation model. But in this process, greedy strategy to build candidate pool will bring statistical distortion. To solve the problem, in this paper, we design a candidate pool construction method which conforms to the real sampling of language model. In addition, we change the coding mode and embed secret bits according to the probability similarity to minimize the coding distortion in the candidate pool. The function of secret information hiding and extracting is realized. Experimental results show that this method improves the consistency of probability distribution between the steganographic text and the normal texts, guarantees the diversity of generated texts, and greatly improves the anti-steganography ability and statistical security of steganography.

The proposed method guarantees the statistical characteristics of the steganographic text set, but has no restrictions on the semantics of the generated text, and it is difficult to generate coherent paragraphs with specific semantics. In future work, we hope to design a steganography system that can meet certain semantic constraints.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (Grant No. 61872368 and No. 61802410).

References

[1] B. Li, S. Tan, M. Wang, J. Huang, “Investigation on cost assignment in spatial image steganography,” *IEEE Transactions on Information Forensics and Security*, 9(8), pp. 1264–1277, 2014. Article (CrossRef Link)

[2] X. Liao, J. Yin, M. Chen and Z. Qin, “Adaptive Payload Distribution in Multiple Images Steganography Based on Image Texture Features,” *IEEE Transactions on Dependable and Secure Computing*, 2020. Article (CrossRef Link)

[3] T Y. Tew and K. Wong, "An Overview of Information Hiding in H.264/AVC Compressed Video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 2, pp. 305-319, Feb. 2014, Article (CrossRef Link).

[4] Y. Xue, J. Zhou, H. Zeng, P. Zhong, J. Wen, “An adaptive steganographic scheme for h.264/avc video with distortion optimization,” *Signal Processing: Image Communication*, Vol. 76, pp. 22–30, 2019. Article (CrossRef Link)

[5] Y. Luo and Y. Huang, “Text steganography with high embedding rate: Using recurrent neural networks to generate Chinese classic poetry,” in *Proc. of ACM Workshop on Information Hiding and Multimedia Security*, pp. 99–104, Jun. 2017. Article (CrossRef Link)

[6] J. Wen, X. Zhou, M. Li, P. Zhong, Y. Xue, “A novel natural language steganographic framework based on image description neural network,” *Journal of Visual Communication and Image Representation*, Vol. 61, pp. 157-169, May. 2019. Article (CrossRef Link)

[7] K. Bennett, “Linguistic steganography: Survey, analysis, and robustness concerns for hiding information in text,” 2004.
[8] Z. Chen, L. Huang, Z. Yu, W. Yang, L. Li, X. Zheng, X. Zhao, “Linguistic steganography detection using statistical characteristics of correlations between words,” in *Proc. of International Workshop on Information Hiding*, pp. 224-235, 2008. Article (CrossRef Link)
[9] P. Wayner, “Mimic functions,” *Cryptologia*, vol. 16, no. 3, pp. 193–214,1992.
[10] M. Chapman and G. Davida, “Hiding the hidden: A software system for concealing ciphertext as innocuous text,” *Information and Communications Security*, pp. 335-345, 1997. Article (CrossRef Link)
[11] H. H. Moraldo, “An approach for text steganography based on markov chains,” *arXiv preprint arXiv:1409.0915*, 2014. Article (CrossRef Link)
[12] Z. Yang, S. Jin, Y. Huang, Y. Zhang, and H. Li, “Automatically generate steganographic text based on markov model and huffman coding,” *arXiv preprint arXiv:1811.04720*, 2018. Article (CrossRef Link)
[13] T. Fang, M. Jaggi, and K. Argyraki, “Generating steganographic text with lstms,” in *Proc. of the 55th Annual Meeting of the Association for Computational Linguistics - Student Research Workshop*, pp. 100-106, Jul. 2017. Article (CrossRef Link)
[14] Z. Yang, X. Guo, Z. Chen, Y. Huang and Y. Zhang, "RNN-Stega: Linguistic Steganography Based on Recurrent Neural Networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1280-1295, May 2019. Article (CrossRef Link)
[15] Z. Yang, S. Zhang, Y. Hu, Z. Hu and Y. Huang, "VAE-Stega: Linguistic Steganography Based on Variational Auto-Encoder," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 880-895, 2021. Article (CrossRef Link)
[16] F. Dai and Z. Cai, “Towards Near-imperceptible Steganographic Text,” in *Proc. of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4303-4308, Jul. 2019. Article (CrossRef Link)
[17] Z. M. Ziegler, Y. Deng, and A. M. Rush, “Neural linguistic steganography,” in *Proc. of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1210−1215, Nov. 2019. Article (CrossRef Link)
[18] A. Go, R. Bhayani, and L. Huang, “Twitter sentiment classification using distant supervision,” *CS224N Project Report*, Stanford, vol. 1, no. 12, 2009. Article (CrossRef Link)
[19] 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. of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1. Association for Computational Linguistics*, pp. 142–150, Jun. 2011.
[20] “kaggle,” 2017. [Online]. Available: https://www.kaggle.com/snapcrack/all-the-news/data
[21] S. Kullback. R. A. Leibler, "On Information and Sufficiency," *Ann. Math. Statist*, 22 (1), pp. 79 - 86, Mar. 1951. Article (CrossRef Link)
[22] D.P. Kingma, J. Ba, “Adam: A Method for Stochastic Optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
[23] P. Meng, L. Hang, W. Yang, Z. Chen and H. Zheng, "Linguistic Steganography Detection Algorithm Using Statistical Language Model," in *Proc. of 2009 International Conference on Information Technology and Computer Science*, pp. 540-543, 2009. Article (CrossRef Link).
[24] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, Y. Yu, “Texygen: A benchmarking platform for text generation models,” in *Proc. of The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 1097–1100, 2018. Article (CrossRef Link)
[25] J. Wen, X. Zhou, P. Zhong, and Y. Xue, “Convolutional neural network-based text steganalysis,” *IEEE Signal Processing Letters*, vol. 26, no. 3, pp. 460–464, 2019. Article (CrossRef Link)
[26] Z. Yang, K. Wang, J. Li, Y. Huang and Y. Zhang, "TS-RNN: Text Steganalysis Based on Recurrent Neural Networks," *IEEE Signal Processing Letters*, vol. 26, no. 12, pp. 1743-1747, Dec. 2019. Article (CrossRef Link).

[27] Devlin, J. et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. of NAACL-HLT*, pp. 4171-4186, June. 2019. Article (CrossRef Link)
[28] S. Zhang, Z. Yang, J. Yang and Y. Huang, "Provably Secure Generative Linguistic Steganography," in *Proc. of Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3046-3055, 2021. Article (CrossRef Link)
[29] J. Shen, H. Ji and J. Han, "Near-imperceptible Neural Linguistic Steganography via Self-Adjusting," in *Proc. of EMNLP*, pp. 303-313, 2020. Article (CrossRef Link)

Boya Yang received her B.E. degree from Department of Computer Science and technology, Beijing Jiaotong University, Beijing in 2018. She is now pursuing the M.E. degree in Department of Information and Electrical Engineering, China Agricultural University, Beijing. Her current research interests mainly focus on information hiding and Natural Language Processing. (Email: yangboya@cau.edu.cn).

Wanli Peng received the B.E. degree in Collage of Physics and Electronic Engineering in 2018 from Harbin Normal University, China. He is currently pursuing the Ph.D. degree with the College of Information and Electrical Engineering, China Agricultural University, Beijing, China. His research interests include information hiding and natural language processing. (Email: hunanpwl@cau.edu.cn).

Yiming Xue is currently a professor with the College of Information and Electrical Engineering, China Agricultural University. His research interests include multimedia processing, multimedia security, VLSI design. (Email: xueym@cau.edu.cn).

Ping Zhong is a professor and Ph. D. supervisor in College of Science, China Agricultural University, Beijing, China. She has published many papers. Her research interests include machine learning, support vector machines, steganalysis. (Email: zping@cau.edu.cn).