Text Steganography Based on Ci-poetry Generation Using Markov Chain Model

Yubo Luo[1], Yongfeng Huang[1], Fufang Li[2] and Chinchen Chang[3]
[1] Department of Electronic Engineering, Tsinghua University
Beijing, 100084 - P. R. China
[e-mail: yfhuang@tsinghua.edu.cn]
[2] School of Computer Science and Educational Software, GuangZhou University
Guangzhou, 510006 - P. R. China
[3] Department of Information Engineering and Computer Science, Feng Chia University
Taichung 427, Taiwan
*Corresponding author: Yongfeng Huang

Received April 19, 2016; revised June 12, 2016; accepted June 29, 2016; published September 30, 2016

Abstract

Steganography based on text generation has become a hot research topic in recent years. However, current text-generation methods which generate texts of normal style have either semantic or syntactic flaws. Note that texts of special genre, such as poem, have much simpler language model, less grammar rules, and lower demand for naturalness. Motivated by this observation, in this paper, we propose a text steganography that utilizes Markov chain model to generate Ci-poetry, a classic Chinese poem style. Since all Ci poems have fixed tone patterns, the generation process is to select proper words based on a chosen tone pattern. Markov chain model can obtain a state transfer matrix which simulates the language model of Ci-poetry by learning from a given corpus. To begin with an initial word, we can hide secret message when we use the state transfer matrix to choose a next word, and iterating until the end of the whole Ci poem. Extensive experiments are conducted and both machine and human evaluation results show that our method can generate Ci-poetry with higher naturalness than former researches and achieve competitive embedding rate.

Keywords: Steganography, Ci-poetry Generation, Markov Chain, Mutual Information

This work is supported by the National Science Foundation of China (Grant Nos. 61472092 and U1405254).

1. Introduction

Recently, steganography which hides secret information in various media, such as images, videos, audio, texts, etc., has become a hot research topic. It has been widely applied in multiple areas, especially in covert communication and watermark. How to select a cover-media is the key technology of steganography. Numerous approaches have been proposed and their differences mainly rely on their tactics on cover media.

Since text is the most widely used media type, information hiding based on text is of great value. Existing textual techniques are mainly divided into two groups: one is based on text format and the other is based on text content. Algorithms based on text format include spacing [1], word-shifting [2], character-coding [3], etc. The main disadvantage of format-modifying methods is the lack of sufficient robustness. A little change to the format may cause irreversibility of the hidden information.

As the development of natural language process, researchers design systems which modify cover texts on lexical, syntactic or semantic level. Some popular methods are based on word-replacement [6], translation [8] and text generation [4-5] [9-13]. These linguistic approaches, except for those based on generation, heavily rely on the private or restricted access to the original unchanged texts, which means that the potential attacks based on comparison pose a major threat to the concealment. TEXTO [4], NICETEXT [5] are two famous examples based on text generation, but their major vulnerabilities rely on the compilation of word lists or grammar rules, which are impossible to be compiled perfectly in all cases. Method in [15] uses Markov chain model to generate English texts, but due to the complexity of normal language model, even high-order Markov model cannot do this job well. Therefore, the generated cover texts are extremely unnatural. However, schemes that generate texts of special genre, such as notes [9], jokes [10], Internet protocol [14, 24], etc., have a better performance. Motivated by this observation, we look for proper solution in traditional Chinese literature and find Ci-poetry is an ideal cover candidate.

We develop Ci-Based Steganography Methodology (Cistega) in this paper, which uses Markov model to generate Ci-poetry, a traditional Chinese literature style. As each Ci poem has fixed syllables rhythm and sentence length, and no complex grammar rules, the generation process can be simplified into the following steps. First, it determines parameters including tone pattern, *StackList* capacity (a list constructed to deposit potential next-state words for the sake of information hiding, see details in section 3.3) and an initial word. Second, choose words which meet rhythm rules from the markov transfer matrix and put them into *StackList*. Third, encoding *StackList* and choosing a word whose code is matched to the bitstream of secret message. Last, repeat the previous two steps until all data are fully concealed.

The main advantages of Cistega are as follows. Generating a Ci poem is actually choosing proper words according to a specific tone pattern, which Markov chain model is very good at, and no complex grammar rules need to be considered. Furthermore, Ci-poetry is the work of traditional Chinese literature, so that it is difficult to verify whether a Ci poem is good or not for common readers. Thus, certain naturalness is enough to defy nonexpert identification. Finally, its high embedding rate is competitive with other generation-based approaches.

In the remainder of this paper, Section 2 introduces related work. A framework of the Cistega and algorithm details are introduced in Section 3. Followed are the experimental results and evaluations in Section 4. Finally, conclusions are drawn in Section 5.

2. Related Work

This section first presents poem-generation related works which mainly focus on generating poems and have nothing to do with steganography, and then discusses text-generating steganography methods. All methods can be categorized into three parts.

2.1 Poem generation

If steganography is not taken into consideration, there are several methods that focus mainly on classic Chinese literature generation. A couplet system [18] was developed by Microsoft Research Asia. A couplet usually contains two lines, and this system, by virtue of machine translation, can offer many options for the second line based on the first line given by the user. He and Zhou [19] adapted this couplet-generating method to a man-machine cooperation approach of writing metrical poetry, also called quatrain, which is similar to couplet system to some extent. He and Zhou use statistical machine translation models (SMT) to generate quatrain. Zhou and You [20] applied genetic algorithm to generating Ci-poetry which satisfy all constraints of a given tone pattern and have fair fluency, coherence and meaningfulness.

However, all of these methods cannot hide information. The algorithm architecture of these approaches determines the inconvenience of directly incorporating steganography into them.

2.2 Steganography generating normal-genre texts

Most of previous text-generation steganography concentrate on producing normal-genre texts, or non-literature texts. Wayner's algorithm [16] mimics statistical characteristics of a normal file, then generates character sequences having similar statistic profile with the original file, and one of its applications is Spammimic [17] which mimics spam. By this method, it is resilient against statistical attacks but the fabricated texts are nonsense. TEXTO [4] and NICETEXT [5] are two synonym-based methods, and generate readable texts by using sentence patterns and organized word bank. There are many pre-reserved positions in these sentence patterns, so that the sender can embed message by manipulating synonym-substitution in these positions. But, their output texts are absurd and nonsense. The work in [15] uses Markov chain model to generate English texts, by taking each word as a transfer state in first-order Markov model and then outputting English words sequence based on embedding message. This approach can be applied to many other languages but cannot resist human examination, as its content can easily draw suspicions.

Non-literature text generation is a notorious task all the time, and no previous works are competent to produce readable texts and hide information at the same time. This motivates us to find a solution from the aspect of text genre.

2.3 Steganography generating special-genre texts

Generally speaking, steganography generating special-genre texts have better performance in diverting suspicions. Desoky exploited many special text forms, such as notes [9], jokes [10], chess [11], etc., to hide information. The commonality of Desoky's methods lays on their utilization of special genres which usually have simpler rules (e.g., grammar, semantic rules). There is one example [7] that combines information hiding with Ci-poetry generation. As we all know, all Ci poems have fixed tone patterns, and each pattern has fixed segmentation rules, so the work in [7] segments Ci-poetry which share the same tone pattern into many parts (see **Table 1**). The generation process is all about choosing a word for each part. For example, assuming that there are N Ci poems who all belong to the same tone pattern, we divide each of them into n parts according to their tone pattern. So, we can get n word lists and each list

contains *N* words. The generation process is to choose a word from each list, then combine these chosen words into one Ci poem. Apparently, embedding capacity is as high as $n*\log_2 N$, but it chooses words from lists totally randomly. That sharply weakens its resistance to third-parts who check word frequency or collocation.

From this point, we propose Cistega which generates special-genre texts. Cistega takes advantages of Markov chain model and chooses words according to the state transfer matrix, greatly improving the naturalness of generated Ci-poetry.

3. Cistega Methodology

Cistega achieves legitimacy by using Markov chain model to generate Ci-poetry. As stated earlier, Ci-poetry has fixed tone and rhythm rules, which greatly facilitates the generation process. This section introduces how Cistega works in detail.

3.1 The Framework of Cistega

Concisely, Cistega has two important missions, generating readable Ci-poetry and hiding secret messages. Based on different functions of each part, the architecture of Cistega can be divided into four separate modules as shown in **Fig. 1**.

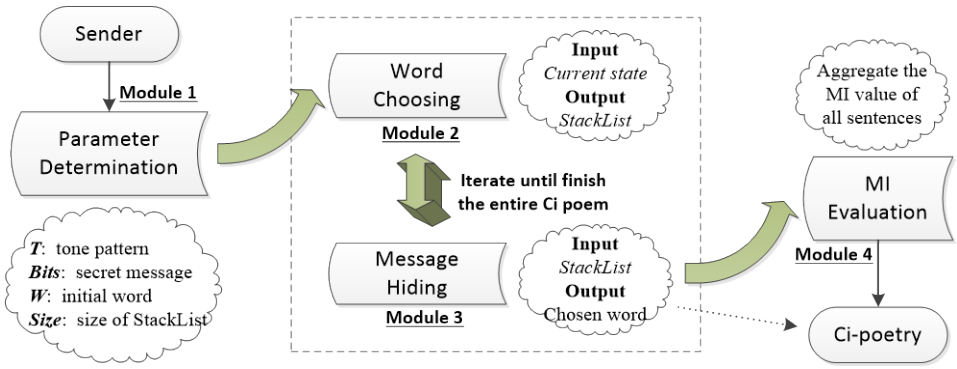

Fig. 1. Illustrates the architecture of Cistega

1. **Parameter determination** (Module 1): Determines four important parameters, including *tone pattern* which is responsible for sentence length, rhythm and tones; *secret message* that we want to hide; *initial word* which is as a seed to begin the generation process; *size of StackList* that is strongly related to the embedding rate of generated Ci-poetry.
2. **Word choosing** (Module 2): Picks out all potential words who meet the requirement of tone pattern and put these words into *StackList*. *StackList* is the output of this module. Concisely, it uses Markov state transfer matrix to choose the potential words according to a *current-state* word. At the first time, initial word input from Module 1 is the current state, but afterward, the current states are all from the output of message hiding module.
3. **Message hiding** (Module 3): utilizes Huffman coding to assign an unique code to each word of *StackList*, then choose one word whose code is matched to the binary stream of secret message, finally output this choose word. This chosen word will be input backward into Module 2, and Cistega iterates between this two modules till the entire Ci poem is finished.

4. **MI evaluation** (Module 4): evaluates the Ci-poetry generated by Cistega by counting the mutual information of each adjacent words. It is not a necessary step for the sender, only an alternative option.

3.2 Parameter Determination (Module 1)

There are several parameters that play important roles in generation process. Firstly, the sender must choose a **tone pattern**. As this tone pattern is the ground work, we need to give a brief introduction to the background knowledge of Ci-poetry.

Ci-poetry emerged around Tang dynasty in response to the popularity of foreign musical tunes. As Ci-poetry has many tone patterns (cipai 詞牌), composers usually first choose one tone pattern, and then write Ci lyric that meets the requirement in sentence length, syllable tone, rhythm, etc.

Each tone pattern has fixed tones, rhythm and sentence length. There are two types of tone: "平" and "仄". The former one "平" means level tone and "仄" means deflected tone, while "中" means either tone is ok for this syllable. Rhythm here means "韵", which requires specific ending words to have the same rhythm. It is these rules which are originally made for the sake of singing, that greatly remove ambiguity of language model of Ci-poetry.

Table 1. Segmentation by Tune Pattern

Tune Pattern: Butterflies kiss flowers
中仄/中平/平仄仄（韵）。 中仄/平平，中仄/平平仄（韵）。 中仄/中平/平仄仄(韵)，中平/中仄/平平仄(韵)。
临水/人家/深宅院（韵）。 阶下/残花，门外/斜阳岸（韵）。 柳舞/面尘/千万线(韵)，青楼/百尺/临天半(韵)。

Another benefit of tone pattern is that it can help us to segment Ci-poetry. Different from English texts, which have space between words, Chinese texts have no such space within one sentence. The performance of current Chinese word segmentation methods are not perfect even for normal Chinese texts, so we cannot apply existing segmentation tools to Ci-poetry directly. But note that once the tone pattern is determined, the tune and rhythm of syllables, and sentence length are all fixed, we can do segmentation according to tone patterns. An example is shown in **Table 1**.

Secondly, a **beginning word** should be determined before the generation. As we all know, Markov state transfer matrix discusses the transition probability between states. So, an initial state is supposed to be predetermined.

The third parameter is the **size of *StackList*** which is used to deposit potential *next words*. The larger its size is, the more information Cistega can hide. Details of *StackList* is discussed in Section 3.3.

Lastly, we need to convert the secret message into **binary stream** directly. No encryption method is used here, as the steganography only hides the existence of secret message. Whether the hiding information is plain text or encrypted message is beyond the scope of our discussion.

In conclusion, **tone pattern** and the **beginning word** are responsible for the style of Ci-poetry, the **size of *StackList*** affects the embedding rate, and the **binary stream** is the message we want to transmit. In order to simplify the expression of these parameters in latter sections, we denote tone pattern as ***T***, beginning word as ***W***, binary stream as ***Bits*** and the size of *StackList* as ***Size***.

3.3 Word Choosing (Module 2)

As stated earlier, the generation process is all about to select proper words based on a chosen tone pattern. Because information hiding requires redundancy, we actually do not choose only one word for each position. Instead, we choose a set of potential words and put them into *StackList*. Thus, the task of this module is to output a *StackList* which contains a specified set of words and this list is the core part of Cistega for it is strongly related to the embedding rate.

Words of *StackList* are selected from the state transfer matrix, denoted as ***M***, which is derived from a given corpus by using Markov model. Next, we briefly introduce Markov model and change it slightly, in order to make it more suitable for Ci-poetry generation.

Markov model is an important model in stochastic process that describes a random sequence discrete both in time and in state, among which each variable value depends only on previous states. Suppose that there is a system with limited states $S=\{s_1, s_2, ..., s_N\}$, and as time goes by, it changes from one state to another. $Q=(q_1, q_2, ..., q_T)$ is a random sequence, whose values are sampled from *S*. Denote the state of the system at time *t* as q_t, then the probability of $q_t=s_j$ is depended on the states of moments from *1* to *t-1*. If we suppose, system state at *t* is only to do with the state at *t-1*, so we have

$$P(q_t = s_j \mid q_{t-1} = s_i, q_{t-2} = s_k, ...) = P(q_t = s_j \mid q_{t-1} = s_i) \tag{1}$$

Then, this system is called a discrete first-order Markov chain.

Therefore, if Eq. (1) is independent of t, this stochastic process can be expressed as:

$$P(q_i = s_j \mid q_{t-1} = s_i) = a_{ij},\ 1 \le j,\ j \le N \tag{2}$$

This process is called Markov model, and transition probabilities satisfy the following constraints:

$$\sum_{j=1}^{N} a_{ij} = 1,\ where\ \ a_{ij} \ge 0. \tag{3}$$

Obviously, the size of the state transition matrix for first-order Markov process is *N***N*, where *N* is the total states.

In order to make the matrix more suitable for the generation algorithm, a_{ij} here represents frequency instead of probability, where a_{ij} means how many times $word_j$ appears right after $word_i$. For example, "天上 35" (see **Table 2**) means that "天上" appears 35 times right after "人间". Considering that the corpus of Ci-poetry is strictly limited, which may cause that there are a great amount of zero elements in the matrix, we need to convert ***M*** into a chain table, like $M=\{m_i \mid i=1,2,...,n\}$, where m_i is a chain that contains all the words (*ArcNodes*) appearing right after $word_i$ (*VexNode*). An example of the state transfer chain is shown in **Table 2**.

Table 2. Example of State Transfer Chain

M	VexNode	ArcNodes
m_i	人间	天上 35 今古 11 春意浓 2 …
m_j	东风	吹散 5 依旧 3 吹柳带 1 …
m_k	四海	无人 3 有知音 2 …
…	…	…

Now, we have gathered all necessary elements, ***T***, ***W***, ***Size*** and ***M***. Then, this word choosing module can be expressed as:

$$Module\ 2\,(T, W, Size, M) \rightarrow StackList,$$

where ***T*** is the tone pattern, ***W*** is the initial word, ***Size*** is the maximum word number of *StackList*, and ***M*** is Markov state transfer matrix. Details are shown in **Algorithm 1**.

Algorithm 1. Word Choosing Module

Step 1: If *current state* is the ending word of previous sentence, go to Step 6, otherwise to next step;
Step 2: Find the chain m_i whose *VexNode* is *current state* and pick words that completely* meet requirements of *T* from m_i into *StackList*, and the maximum word number must be less then *Size*;
Step 3: If *StackList* < 2, go to next step, otherwise to Step 9;
Step 4: Pick words that roughly* meet requirements of *T* from m_i into *StackList*, and the maximum word number must be less then *Size*;
Step 5: If *StackList* < 2, go to next step, otherwise to Step 9;
Step 6: Pick words that completely meet requirements of T from chains whose VexNodes are in *UsedWordList** into *StackList* ;
Step 7: If *StackList* < 2, go to next step, otherwise to Step 9;
Step 8: Pick words from random chains until *StackList* is full;
Step 9: Output *StackList*.

**Completely* means it should meet all requirements of *T*, including word length, syllable tune and rhythm, while **Roughly* means syllable tune is exempted from these requirements. This is intended to reduce chain breaks.
**UsedWordList* is where the initial word of each sentence comes from, and its details are shown in the *solution 2* of Section 3.4.

Finally, Module 2 outputs *StackList* which contains a set of words, but not more than the number of ***Size***. *StackList* is used as input in the next module.

3.4 Message Hiding (Module 3)

Implementing the message hiding follows a two-steps process. First, according to the number of words of *StackList*, we use Huffman coding to code each word, assuming all words have equal transfer probability. Second, based on ***Bits***, the binary stream of secret message, we pick out a word from *StackList* whose Huffman code is the same as the beginning digits of ***Bits***. This chosen word will be input as *next state* into Module 2 in the next embedding position.

Algorithm 2. Message hiding Module

Step 1: Input *StackList* and apply Huffman coding by supposing that all words have equal transfer probability;
Step 2: Assign one Huffman code to each word, for the sender, go to next step; for the recipient, go to Step 4;
Step 3: Choose a word whose Huffman code is matched with the binary stream of ***Bits***;
Step 4: Choose a Huffman code whose word is matched with *current state*;
Step 3: Push the chosen word into *UsedWordList* and output this word as current state;

Algorithm details of Module 3 are shown as above. From it, we know that Huffman coding is an indispensable step in **Algorithm 2**. However, Huffman coding has a minimum requirement for the word number of *StackList*, that *StackList* must contain no less than 2 words.

Different from the normal language model which has boundless corpus materials, literature works that we can utilize to build the Markov transfer state matrix are very limited, which leads the matrix very sparse. Many words may only have several following words, and moreover, the requirements of rhythm and tone in ***T*** (tone pattern) will reduce potential words further. An example is given to explain this phenomena.

- assume that the *current state* is "人间";
- assume that the following words of "人间" are: "天 上", "今 古", "方 事", "作 寿 星", "比 梦 间";
- if ***T*** requires the tone of next word to be "平仄", then there are only two words qualified; if it is "仄仄平", then only one word is suitable;

So, there is a high probability for the situation that only one word or no words selected into *StackList*. Since Huffman coding requires at least two nodes, the number of words in *StackList* should be no less than 2, and if not, this generation chain has to stop here.

To cope with the sparsity of state transfer matrix, we propose three solutions:

Solution 1: Loosen the requirement of *Template*

Just as shown in Step 2&4 of **Algorithm 1**, there are two standards for requirement: *completely* and *roughly*. If qualified words are no less than 2 after filtered by ***T*** *(tone pattern)*, we use the high standard of *completely*; otherwise, low standard of *roughly* is adopted.

This solution may cause certain words to violate tone requirement, but considering that this is not a frequent event, we ignore the consequence.

Solution 2: Shorten the generation chain

At the beginning, we thought that we could generate an entire Ci poem with ***W*** (the beginning word), but even *solution 1* has been applied to the system, chain breaks still happen. An entire Ci poem usually contains more than 20 words, it is highly possible for the chain to break between somewhere.

Given that there is actually no strong relations between sentences, we had better start a new generation chain for each sentence. Because each sentence only contains two or three words, this can help decrease chain breaks further.

However, this solution has drawbacks. Assigning separate generation chains to each sentence makes them independent of each other, and the entire poem has difficulty conveying a centralized meaning that logically corresponds with the user's intent. It is necessary to maintain connections between sentences, so that a list called *UsedWordList* is built to deposits all used words that have already been chosen in previous word-choosing rounds. All the initial words (*start state*) of each later sentence come from this list, which would ensure a rough relation between sentences. Experiments in next section proves the efficiency of *UsedWordList*.

Solution 3: Order words in *StackList* by *wordnum*

Even the generation chain now is shortened to one sentence, chain breaks still happen. In order to avoid chain breaks further, we need notice the sort rule of words in *StackList*. Words in *StackList* can be sorted either by word frequency (*wordfreq*) or by the number of following words in state transfer chain table (*wordnum*).

Table 3. Example of sort rule of ArcNodes

VexNode	ArcNodes sorted by wordfreq	wordfreq	ArcNodes sorted by wordnum	wordnum
人间	天上	31	多少	367
	今古	11	天上	259
	尘土	6	今古	135

	多少	2	尘土	58

As shown in **Table 3,** if words are sorted by *wordfreq*, "多少" is on the bottom of the list, but if by *wordnum*, "多少" has the top priority, because it has 367 descendants. Generally speaking, we should sort words by *wordfreq*, for it gives priority to choosing words with higher frequency. However, through the analysis above, we know that our Markov transfer matrix is very sparse, if words are sorted by *wordnum*, words that have more following words are more likely to be chosen, which can reduce the probability of chain break in next word-choosing round. The effectiveness of solution 3 is validated in the experiment of Section 4.2 (see **Table 5**).

These three solutions work together to reduce the breaks of generation process and it is an essential part of Cistega.

In conclusion, we input *StackList* into Module 3, get the output of one chosen word, and embed several beginning digits of ***Bits*** (binary stream of secret message). This is an accomplishment of one single embedding position.

This chosen word will be used as the *current state* in next embedding position, and be input into Module 2 backward. The times of iteration depends on how many embedding positions this Ci poem has. After all secret message has been embedded, an end flag which contains a few unique binary bits is necessary to be added, in order to facilitate the recipient to recover the message.

3.5 MI Evaluation (Module 4)

Module 4 evaluates the quality of generated poems. When a Ci poem is finished, we need to know whether this poem is good or not, so an evaluation method is proposed based on MI (mutual information).

Mutual information is the amount of information one random variable contains about another. It has been widely used in natural language process, especially in the early field of speech recognition [21-22]. Zhou [20] utilizes MI to help implement songci generation and even in the latest neural conversation generation system [23], we can track the trace of MI. Thus, given that MI not only has effectiveness, but the simplicity to implement, it is a practical tool to evaluate the ci-poetry generated by Ci-stega. A benchmark based on MI is proposed to grade Ci poems, and the definition of MI is shown as Formula (4).

$$MI(x,y)=\log\frac{p(x,y)}{p(x)*p(y)}. \tag{4}$$

Informally, mutual information compares the probability of observing *x* and *y* together with the probabilities of observing *x* and *y* independently. In our application, word probabilities *p(x)* and *p(y)* are estimated by counting the number of observations of *x* and *y* in corpus, and normalized by *N*, the total sentence number in corpus. Joint probability, *p(x,y)* is estimated by counting the number of times where *x* is adjacently followed by *y*, and normalized by *N*, too. MI value for one Ci poem is the sum of all sentences' MI value and for one sentence it is the sum of all contiguous word pairs, just as shown in Formule (5) and (6).

$$MI(Ci)=MI(s_1)+MI(s_2)+...+MI(s_n). \tag{5}$$

$$MI(s=w_1/w_2/w_3)=\log_2\frac{p(w_1,w_2)}{p(w_1)P(w_2)}+\log_2\frac{p(w_2,w_3)}{p(w_2)P(w_3)}. \tag{6}$$

To investigate the effectiveness of MI evaluation, experiment are conducted in next section, which proves that this evaluation method works well.

4. Experiments and Evaluation

In this section, validated experiments are conducted to explore the effects of variables and we use MI evaluation method mentioned in previous section to facilitate our analysis. Finally, comparison and expert testing are made to prove the high performance of our method.

4.1 Cistega Example

This section shows how Cistega hides information in the process of generation. For the recipient, it is virtually a reverse process with slight changes, which will be demonstrated too.

First, Module 1 determines the initial parameters as follows.

- ***T*** (tone pattern): *Song of the Country Norm (菩萨蛮)*
  First sentence is "中平/中仄/平平仄 (韵), 中平/中仄/平平仄 (韵)"
- ***W*** (beginning word): 扁舟
- ***Size*** (size of *StackList*): 16
- ***Bits*** (binary stream of "*stega*"): 01010011011110100011001O1...

Then, we come to Module 2, the input of which is ***W***, "扁舟", and its tone meets "中平". Based on Step 2 (**Algorithm 1**), we choose words that meet "中仄" from chain m_i whose *VexNode* is "扁舟", and then sort these words by *wordnum*, put the top 16 words into *StackList* as shown in column 1 (**Table 4**). When word-choosing module finishes its work, input *StackList* into message-hiding module.

Table 4. StackList in the example

Current word 扁舟		*Current word* 明月		*Current word* 南堂宴	
StackList	Huffman	StackList	Huffman	StackList	Huffman
一叶	0000	南堂宴	00	东风	0000
归去	0001	粉香残	01	西风	0001
…	…	隔云山	100	…	…
明月	0101	起青翰	101	落花	1101
…	…	复山川	110	…	…
摇荡	1111	木兰船	111	又还	1111

Module 3 assumes all these words have the same transfer probability, and gives each word a Huffman code. Then, "明月" is chosen here because its code, 0101, is matched to the beginning bits of ***Bits***. The embedding process of the first position is finished by now.

Next, as stated in Section 3.4, the choosing word "明月" will be input into Module 2 backward, as *current state*. Likewise, continuing to find words whose tone meets "平平仄", from chain m_j whose VexNode is "明月". But this time, there is only one word in chain m_j who meets the requirement. So, we need to move to Step 4 (**Algorithm 1**) where 6 words are picked out as shown in column 2 (**Table 4**). Then we choose "南堂宴" based on Step 3 (**Algorithm 2**). Now, the first sentence is accomplished as "扁舟明月南堂宴".

Next, according to Step 1 (**Algorithm 1**), we go to Step 6 (**Algorithm 1**) directly. Column 3 (**Table 4**) shows the selected words and "落花" is chosen to be the next word. Then follow the way of previous sentence, we can get the second sentence "落花寂寂水潺潺". Next, repeat what has been mentioned above until the bitstream is all finished.

In order to recover the hidden message correctly, flag bits should be added to the end of the message's bitstream. When we apply Cistega to practical usage, the hiding capacity of one Ci poem must be larger than the binary stream of secret message. This means that ***binary stream***

is supposed to be finished earlier than the poem. Therefore, once the binary stream is completed, we do not need to choose words from *StackList* by matching Huffman code as described in Step 2 (**Algorithm 2**), we can directly select the one who has the highest *wordfreq*, which can increase the naturalness as possible as it can.

As for the recipient, they follow the same schedule as the sender. The only difference relies on Step 2 (**Algorithm 2**). For the sender, they choose a word according to ***Bits*** in each word-choosing round, while for the recipient, they extract out the binary stream according to received Ci-poetry. Then, the position of flag bits should be located and discard the latter part. Finally, recover the hidden message according the binary stream extracted.

4.2 Experimental Results

- **Data and Preprocess**

We collated a corpus of Ci-poetry from *Complete Collection of Song Period Ci-poetry*, which is also called *Quansongci* and was compiled by the modern scholar Tang. *Quansongci* contains about 21,000 Ci poems in total, composed by more than 1300 authors.

However, as time goes by, for a small part of Chinese words, their pronunciations have changed. In order to simplify the work of corpus processing, we determine the tone of each character based on modern Chinese phonetic.

- **Effectiveness of MI evaluation**

In order to investigate MI evaluation's effectiveness, some experiments are carried out, where subjects are chosen from three parts: (1) *Quansongci*; (2) the ones generated by the algorithm in [7]; (3) the ones by our proposed method. Each part contains 100 samples.

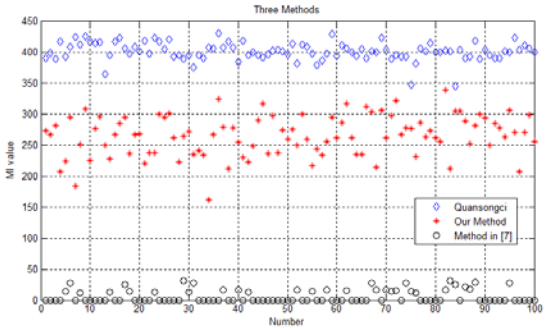

Fig. 2. Test on the effectiveness of MI evaluation

Fig. 2 shows that Ci-poetry from *Quansongci* has the highest MI value and smallest fluctuation; in our method, MI value is a little lower and has a bigger fluctuation; method in [7] has the lowest score, which stems from its randomly choosing words.

These experiments are strongly consistent with the naturalness of tested Ci-poetry and prove that MI evaluation is a useful tool to reflect the quality of Ci-poetry.

- **Property of StackList**

As discussed earlier, *StackList* plays a very important role in Cistega. First, the size of *StackList* affects the hiding efficiency. If *StackList* can hold 32 words mostly at one time, it can hide 5 bits at most. Apparently, the bigger *StackList* size is, the more data it can hide. Second, the sort rule of words in *StackList*, either by *wordfreq* or by *wordnum*, affects the generation chain breaks. Experiments are conducted to investigate this two factors.

Table 5 shows that with the increase of *StackList* size, hiding capacity goes up until it reaches at a summit. If words are sorted by *wordnum*, the chain has a lower probability of

being broken, especially in the condition of small size of *StackList*, which improves MI value greatly. So, being sorted by *wordnum* is obviously a better solution when the size of *StackList* is small.

Table 5. Tune : *Charm of a Singer*

Sort rule	Size of StackList	Embedding capacity/bit	MI value	Generation Chain breaks
wordfreq	8	126	261	7
	16	153	248	8
	32	173	228	9
	64	187	218	9
	128	190	216	10
wordnum	8	133	370	2
	16	163	336	3
	32	181	295	6
	64	182	266	7
	128	184	233	9

In order to extend the scope of experiment, in next part, other 4 famous tone patterns are added to investigate the difference between tone patterns.

◆ **Extension to more tone patterns**

To expand the comparison, four famous tone patterns are added to our experiment. In **Fig. 3 (a)**, we can see that the maximum of embedding rate is around 10%, and the curves of different tone patterns are faintly layered, which is caused by different sentence lengths in different patterns. For example, sentences with five syllables are mainly segmented in the form of *ww/www*, whose *embedding number/character number ratio* is 0.33 (2/6, 2 embedding positions with 6 characters totally, including five syllables and one punctuation), and for sentences with four, six or seven syllables, their ratios are 0.4, 0.43 and 0.375, respectively. Apparently, higher ratio outputs higher embedding rate. **Table 6** is obtained by adding ratios of all sentences and normalized by sentence number, which corresponds strongly with layers in **Fig. 3 (a)**.

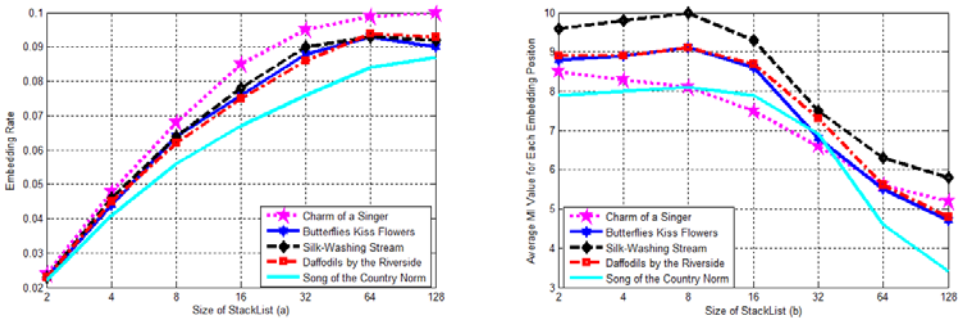

Fig. 3. Effects of different tone patterns **(a)** on embedding rate; **(b)** on MI value;

In **Table 6**, we can easily see that the tone pattern of *Charm of a Singer* has the highest *embedding number/character number* ratio, because of its high percentage of 4-word and 6-word sentences whose ratios are higher than the rest two sentence lengths. While, the tone pattern of *Song of the Country Norm* has the lowest ratio, as most of its content are 5-word sentences whose ratio is only 0.33, the lowest one among these four sentence lengths. Thus, by this principle, we can classify tone patterns with regard to embedding rate, especially when we

need high efficiency, we can choose who has higher *embedding number/character number* ratio as the generation template.

Table 6. Ratio of different tone patterns

Tone pattern	4-word sentence (0.40)	5-word sentence (0.33)	6-word sentence (0.43)	7-word sentence (0.38)	Average *embedding number/character number* ratio
Charm of a Singer	10	4	4	1	0.390
Butterflies Kiss Flowers	2	2	0	6	0.371
Silk-Washing Stream	0	0	0	6	0.375
Daffodils by the Riverside	0	4	2	4	0.368
Song of the Country Norm	0	6	0	2	0.341

In **Fig. 2 (b)**, curves are not distinctly layered but there is an obvious downward slope when the size of *StackList* increases from 8 to 128. High MI value and embedding rate are both what we want, because high MI value means better concealment and security, while high embedding rate means greater efficiency. But tradeoff must be made between security and embedding rate when deciding the size of *StackList*, as they are inversely related to one another. We can define a hierarchy system in which users can set the size of *StackList* according to the security level they want.

Since, in **Fig. 2 (a)**, there is no increasement in embedding rate when *StackList* size changes from 32 to 128, and in **Fig. 2(b)**, the slope does not go down until the size is bigger than 8, we can set three security levels as shown in **Table 7**.

Table 7. Hierarchy of security level

Size of *StackList*	8	16	32
Security level	High	Moderate	low

This hierarchy system grants users access to controlling the quality of generated Ci poems according to their own needs.

- **Embedding rate**

On the basis of the hierarchy system above, the embedding rate of presented Ci-stega fluctuates roughly between 7%~10%. Compared to other steganography approaches, Ci-stega is competitive among the current methods.

Table 8. Embedding rate of different systems

Models	Embedding rate (%)	Comment
NICETEXT	0.29	Noted by the authors in [13]
Steganography based on English-text generation [15]	5 ~ 10	Noted by the authors in [15]
High embedding ratio text steganography based on ci-poetry generation [7]	10 ~ 16	Noted by the authors in [7]
Ci-stega	7 ~ 10	Based on the corpus of *Quansongci*

As shown in **Table 8**, the early method of NICETEXT has the lowest embedding rate, mainly caused by the redundancy of English text. The work in [15] also uses Markov chain model to generate English text, but it outputs text letter by letter, which improves the embedding rate tremendously due to the increase of embedding positions. The approach in [7] has the highest embedding rate, while its high embedding efficiency is traded by the quality of

generated Ci-poetry, as it chooses words randomly. Theoretically, it can improve efficiency infinitely by expanding the word bank, but this kind of high embedding rate is no point in practical use.

It should be mentioned that this highest embedding rate, 10% of Cistega, is only based on the corpus of *Quansongci*. If we expand the scope of corpus and use more data to build a larger and denser state transfer matrix, the highest embedding rate is bound to rise. Theoretically, if the state transfer matrix is dense enough that in each word-choosing module, the words selected into *StackList* reach the maximum number, the expected embedding rate should be as Formula (7).

$$Embedding\ \ rate = \frac{ratio * \log_2 size}{16} \tag{7}$$

where **ratio** is the *embedding number/character number ratio* (see **Table 6**), **size** is the capacity of *StackList*, and **16** means each Chinese character equals 16 binary bits.

For instance, we choose *Charm of a Singer* as the template and the size of *StackList* is 128, then the theoretical embedding rate is 17%, even higher than [7]. Thus, Ci-stega makes a perfect tradeoff between embedding rate and the quality of output Ci-poetry, as it produces high-quality Ci-poetry without sacrificing embedding efficiency.

◆ **Generated example of Cistega**

Table 9 shows one example of Ci poem produced by our model, whose tone pattern is *Charm of a Singer*. This Ci poem meets the requirement of all syllable tunes and rhyme, and its meaning is consistent in a sad theme.

Table 9. Generation Exapmle *Charm of a Singer*

不知多少， 洞天谁道在， 一笑樽前。 底事今年春事早， 回首当日三贤。 几度春风， 西风明月， 千里倍潸然。 断肠风月， 少年无限当年。	So complicated that I can hardly tell where the paradise is, just holding my wine cup, smiling. The spring comes especially early this year, reflecting back on the three most talented people in the good old days. Spring breeze has repeatedly come and gone the same with west wind and the bright moon, dropping my tears thousands miles away. Accompanied by wind and the moon, I was in extreme sorrow, thinking of my salad days.

By careful scrutiny, there does exist a few shortcomings. The logic relation between sentences is not very strong and it might be better if sentence order is changed a little. That is actually what we should focus on in further research. However, in general, this is a qualified Ci poem.

4.3 Human Evaluation

Expert testing is conducted to confirm our method's performance in confidentiality. As our Ci-poetry generation system is designed for steganography, our prime objective is to make the poems legitimate and avert suspicion. Thus, we do not take regular Ci-poetry evaluation

methods in which poems are usually graded with regard to their fluency, coherence, meaningfulness, etc., such as the method used in [20]. We adopt a so-called ABX blind test to evaluate the performance of Cistega in averting suspicion.

A blind test is a method of testing in which the people being experimented on have no idea about what they are receiving. This test method prevents results from being influenced by any priori information. In our testing, ***A*** represents the Ci-poetry written by humans, while ***B*** represents Ci-poetry generated by machine. The comparison test data are the Ci-poetry from *Quansongci* and generated by the method in [7].

Participants are 20 Master candidates from Department of Chinese Language and Literature. They need to classify the received poem ***X***, into either ***A*** or ***B***. Each participant has to deal with Ci-poetry from five tone patterns, mentioned in **Fig. 3** and each tone pattern generates 10 samples. Overall, each participant has to classify for (3×5×10) samples.□

Table 10. Human evaluation results

Models	Generation template labels*					Average percentage
	T1	**T2**	**T3**	**T4**	**T5**	
Paper [7]	12.5%	25.0%	28.3%	10.6%	18.6%	19.0%
Ci-stega	50.8%	67.5%	64.0.%	51.0%	65.0%	59.7%
Quansongci	66.2%	75.6%	76.0%	72.6%	75.3%	73.1%

*template labels T1~T5 represent *Charm of a Singer*, *Daffodils by the Riverside*, *Silk-Washing Stream*, *Song of Country Norm* and *Butterflies Kiss Flowers*, respectively. The number in the table represents the percentage of being classified to **A**.

The results of our human evaluation are summarized in **Table 10**. We compare our Cistega against high embedding rate method in [7], and *Quansongci* is our reference system. Each column in the **Table 10** reports the percentage of Ci-poetry that is considered by the participants as written by humans, which means that the higher this number is, the better the Ci-poetry is. Obviously, the data set of *Quansongci* has the comprehensive advantage, and Cistega is slightly lower than *Quansongci*, while the method in [7] lags far behind the previous two methods. This strongly demonstrates that Cistega performs well at camouflage, much better than previous researches.

In detail, the percentage of being classified to human-written category, to certain extent, varies with the different templates. According to our experiments, the tone pattern of *Charm of a Singer* has the lowest percentage. This may stem from its long length, which has almost double volume of other patterns, as the lengthier the Ci-poetry is, the more flaws participants can find in collocation, coherence, etc.

5. Conclusion

In this paper, Markov model is used to generate classic Chinese literature with steganography for the first time. The algorithm can not only guarantee high embedding rate, but also generate Ci-poetry that meets requirements of tone patterns and has certain coherence thematically, a great improvement compared to previous researches. The main challenge confronted in the research is the generation chain breaks caused by the sparsity of Markov transfer matrix, more precisely by the inadequacy of corpus. We figure out three solutions to alleviate this problem, and minimize the possibility of chain breaks. Meanwhile, based on mutual information, we propose a statistical evaluation method, which can well evaluate machine-generated Ci-poetry. Finally, effects from different tone patterns on Ci-poetry generated by our Cistega have been investigated and we build a hierarchy system corresponding to several security levels. Challenges and further research should focus on

adding style and emotion elements to words, and give more attention to promoting thematic unity and consistency of style and emotion.

Reference

[1] C. Nopporn, "Electronic Document Data Hiding Technique using Inter-character Space," in *Proc. of IEEE Asia-Pacific Conference on Circuits and System*, pp. 419-122, 1998. Article (CrossRef Link)

[2] MH. Shirali-Shahreza and M. Shirali-Shahreza, "A New Approach to Persian/Arabic Text Steganography," in *Proc. of 5th IEEE/ACIS International Conference on Computer and Information Science*, pp. 310-315, 2006. Article (CrossRef Link)

[3] S. H. Low, N. F. Maxemchuk and A. M. Lapone, "Document Identification for Copyright Protection Using Centroid Detection," *IEEE Transactions Communications*, vol. 46(3) , pp. 372-383, 1998. Article (CrossRef Link)

[4] K. Maher, "Texto", available at URL: ftp://ftp.funet.fi/pub/crypt/steganography/texto.tar.gz.

[5] M. Chapman, "Hiding the hidden: A software system for concealing ciphertext as innocuous text," *Master Thesis*, University of Wisconsin-Milwaukee, 1998.

[6] M. Shirali-Shahreza and M. H. Shirali-Shahreza, "Text steganography in SMS," *International Conference on Convergence Information Technology*, pp. 2260-2265, 2007. Article (CrossRef Link)

[7] Z. S. Yu and L. S. Huang, "High Embedding Ratio Text Steganography by Ci-poetry of the Song Dynasty," *Journal of Chinese Information Processing*, vol. 23(4) , pp. 55-62, 2009. Article (CrossRef Link)

[8] C. Grothoff, K. Grothoff, L Alkhutova, R Stutsman and M Atallah, "Translation-based Steganography," in *Proc. of 7th International Workshop on Information Hiding*, Lecture Notes in Computer Science, pp. 219-233, 2005. Article (CrossRef Link)

[9] A. Desoky, "Notestega: Notes-Based Steganography Methodology," *Information Security Journal: A Global Perspective*, vol. 18(4) , pp. 178-193, 2009. Article (CrossRef Link)

[10] A. Desoky, "Jokestega: Automatic Joke Generation–based Steganography Methodology," *International Journal of Security and Networks*, vol. 7(3) , pp. 148-160, 2012. Article (CrossRef Link)

[11] A. Desoky, "Chestega: Chess Steganography Methodology," *Security and Communication Networks*, vol. 2(6), pp. 555-566, 2009. Article (CrossRef Link)

[12] A. Desoky, M. Younis and H. EI-Sayed, "Auto-Summarization-Based Steganography," in *Proc. of the 5th IEEE International Conference on Innovations in Information Technology*, pp. 608-612, 2008. Article (CrossRef Link)

[13] A. Desoky, "Listega: List-Based Steganography Methodology," *International Journal of Information Security*, vol. 8(4), pp. 247-261, 2009. Article (CrossRef Link)

[14] Y. F. Huang, C. H. Liu, S. Y. Tang and S. Bai, "Steganography Integration into a Low-Bit Rate Speech Codec," *Information Forensics & Security, IEEE Transactions on*, vol. 7(6), pp. 1865-1875, 2012. Article (CrossRef Link)

[15] S. F. Wu, "Research on Information Hiding," *Master Thesis*, University of Science and Technology of China, 2003.

[16] P. Wayner, "Mimic Functions," *Cryptologia* 16(3), pp. 192-213, 1992. Article (CrossRef Link)

[17] Spammimic. www.spammimic.com.

[18] G. Chen, L. Cui and M. Zhou, "Chinese Couplet Generator", couplet.msra.cn/app/couplet.aspx.

[19] J. He and M. Zhou, "Generating Chinese Metrical Poetry by a Statistical MT Approach," *Journal of Chinese Information Processing*, vol. 24(2), pp. 96-103, 2010. Article (CrossRef Link)

[20] C. L. Zhou and W. You, "Genetic Algorithm and Its Implementation of Automatic Generation of Chinese SongCi," *Journal of Software*, vol. 21(3), pp. 427-437, 2010. Article (CrossRef Link)

[21] L. Bahl, P. Brown, P. de Souza and R. Mercer, "Maximum mutual information estimation of hidden Markov model parameters for speech recognition," in *Proc. of Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP*, pp. 49-52, 1986. Article (CrossRef Link)
[22] P. Brown, "The Acoustic-modeling Problem in Automatic Speech Recognition," *Ph.D. thesis*, Carnegie Mellon University, 1987.
[23] J.W. Li, M. Gallry, C. Brockett, J. Gao and B. Dolan, "A diversity-promoting objective function for neural conversation models," *arXiv preprint arXiv:1510.03055*, 2015.
[24] Y. F. Huang, S. Tang and J. Yuan, "Steganography in Inactive Frames of VoIP Streams Encoded by Source Codec," *Information Forensics & Security IEEE Transactions on*, vol. 6(2), pp. 296-306, 2011. Article (CrossRef Link)

Yu-Bo Luo(罗榆博) received his B.S. degree in Telecommunication Engineering from Central South University, in 2014. He is currently pursuing the M.S. degree in the Department of Electronic Engineering, Tsinghua University, Beijing. His research interests include information hiding and natural language process.

Yong-Feng Huang (黄永峰) received his Ph.D. degree in computer science and engineering from Huazhong University of Science and Technology in 2000. He is a professor in the Department of Electronic Engineering, Tsinghua University. His research interests include multimedia network security, steganography and steganlysis, big data mining and next-generation Internet. He has published six books and over 100 research papers on computer network, multimedia communications and security. He holds the prestigious grade of Senior Member in the IEEE.

Fu-Fang Li (李福芳) received his Ph.D. degree in Computer science from South China University of Technology in 2008, MS degree in Computer science from Central South University in 2002, and the BS degree in Physics Science from East China Normal University in 1991. He is currently an associate professor at the School of Computer Science and Educational Software, Guangzhou University, P.R. China. He is a senior member of CCF (China Computer Federation), IEEE and IEEE CS. His research interests include: computer network, network and information security, information hiding, distributed system, grid computing, cloud computing, WSN, etc.

Chin-Chen Chang (張真誠) received his Ph.D. degree in computer engineering from National Chiao Tung University. His first degree is Bachelor of Science in Applied Mathematics and master degree is Master of Science in computer and decision sciences. Both were awarded in National Tsing Hua University. Dr. Chang served in National Chung Cheng University from 1989 to 2005. His current title is Chair Professor in Department of Information Engineering and Computer Science, Feng Chia University, from Feb. 2005. Professor Chang's specialties include, but not limited to, data engineering, database systems, computer cryptography and information security.