Linguistic Steganography by Sampling-based Language Generation

Rui Yang and Zhen-Hua Ling
National Engineering Laboratory for Speech and Language Information Processing,
University of Science and Technology of China, Hefei, China
E-mail: ruiyang@mail.ustc.edu.cn, zhling@ustc.edu.cn

***Abstract*—Linguistic steganography aims to hide secret messages within text carriers. In this paper, we propose a linguistic steganography method by means of sampling-based language generation. Comparing with deterministic text generation using beam-search, the sampling-based approach increases the redundancy of generated texts and benefits the hiding of information. The arithmetic coding (AC) algorithm is adopted to embed messages in our proposed method. Its performance is compared with fixed-length coding (FLC) and variable-length coding (VLC) which were designed for embedding messages during deterministic text generation. Besides, the KL divergence and temperature based strategies are designed to control the embedding rates of FLC, VLC and AC respectively. Experiments using a story generation model show that AC performed better than FLC and VLC when embedding messages during sampling-based text generation. With an embedding rate of 1.45 bits/word, our AC-based steganography method achieved ideal imperceptibility, and the subjective quality of its generated text is as good as the non-steganography one.**

Keywords: linguistic steganography, text generation, arithmetic coding, neural network

I. Introduction

Text is the most commonly used medium of information exchange in everyday life. People transmit a large amount of text data through the Internet every day, and using text as an information carrier has great research value and practical value. Steganography is a information hiding technique that hides secret messages in the cover carrier to ensure that the opponent does not recognize it. Linguistic steganography uses text as a cover carrier to hide secret messages. It is a quite challenging task due to the limitation of natural language process (NLP) techniques and the little redundancy of linguistic representations [1].

Previous studies have exploited different linguistic transformation methods for steganography, such as semantic transformation [2], phrase paraphrasing [3], and synonym substitution [4], [5]. The transformed text can achieve high naturalness and it is difficult to detect them by steganalysis. However, one obvious disadvantage of these methods is the low embedding rate [4].

The methods of linguistic steganography by text generation have also been proposed [6], [7], [8], [9], [10]. A generation-based text steganography method mainly includes two parts, a text generation model and an embedding algorithm. Text carriers are not given. Instead, they are automatically produced by text generation models and the secret messages are embedded during the generation process.

Regarding with generation algorithms, Markov models were used to generate text carriers [6], [10]. However, the quality of the texts generated by Markov models was dissatisfactory. The encoder-decoder model based on recurrent neural networks (RNNs) was used to generate quatrains [7], one genre of Chinese poetry, for linguistic steganography. Although it can generate high quality quatrains, the length of single quatrain was too short to hide much secret information. Language models based on long short-term memory (LSTM) neural networks have also been applied to generate sentences for steganography [8], [9].

Regarding with embedding algorithms, several methods including fixed-length coding (FLC) [9] based on balanced binary trees, variable-length coding (VLC) [6], [9] based on Huffman trees, and fixed-block coding (FBC) [8], have been proposed. During the generation process, the generation model first outputed probability distributions of word candidates at each step. The FLC and VLC algorithms encoded word candidates using balanced binary trees or Huffman trees, and then selected the output word according to the bit sequence of secret messages. For both algorithms, it was inevitable that some words with low probabilities may be selected according to the secret messages, which degraded the quality of generated texts. The FBC algorithm divided all word candidates into different blocks, and then encoded each block with FLC for message embedding. This method didn't consider the probability distributions given by the generation model at all.

The evaluation on linguistic steganography focuses on the security and the embedding rate of steganography. The measurements of steganography security include the quality of generated texts and the ability of resisting steganalysis detection. The embedding rate is the number of bits per word that can be embedded. For previous generation-based linguistic steganography methods, it is difficult to achieve satisfactory steganography security and embedding rate simultaneoulsy.

In this paper, we tackle the challenges of linguistic steganography by using sampling-based language generation. When producing a word sequence, the language generation model first calculates the probabilities of all word candidates at each word position. In contrast to deterministic text generation which finds the optimal word sequence by beam search, the sampling-based approach samples among the K most likely

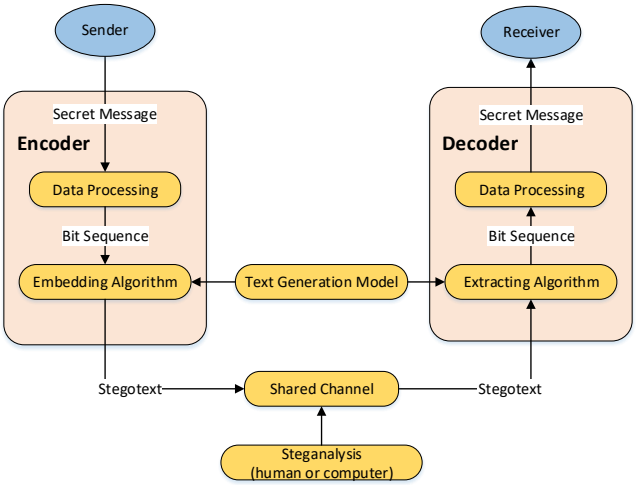

Fig. 1. The framework of linguistic steganography based on text generation.

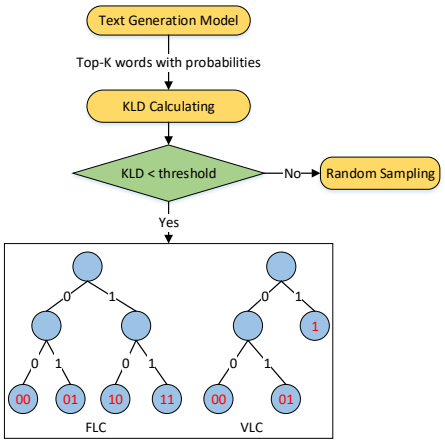

Fig. 2. FLC and VLC with KLD-based word filtering.

candidates according to their probabilities. This approach has demonstrated its effectiveness of generating high-quality texts in some studies [11]. Regarding with linguistic steganography, sampling-based generation increases the uncertainty and redundancy of the generated word sequences, which can help to hide secret messages imperceptibly. In recent study of provably secure steganography on generative media [12], arithmetic coding (AC) was adopted to embed secret messages according to the probability distributions given by generative models. Since this paper adopts similar sampling-based generation approach as this study [12], the AC algorithm is also adopted in our proposed method. Its performance is further compared with FLC and VLC which were initially proposed for embedding messages during deterministic text generation as introduced above. Besides, a KL divergence-based strategy is designed for FLC and VLC algorithms and a temperature-based strategy is designed for the AC algorithm respectively in order to control their embedding rates. Experiments using a story generation model [11] show that AC performed better than VLC and FLC. With an embedding rate of 1.45 bits/word, the AC-based steganography method achieved satisfactory imperceptibility, and the subjective quality of its generated text is as good as the non-steganography one.

The rest of this paper is organized as follows. Section II presents our proposed method. The results of experiments are presented in Section III. Finally, conclusions are drawn in Section IV.

II. Proposed Methods

Fig. 1 shows a general framework of linguistic steganography based on text generation. In the encoder, we first use an ASCII coding map to transform the secret messages into a bit sequence. then according to the bit sequence, stegotext is generated from a text generation model by applying an embedding algorithm. During the transmission of stegotext, steganalysis may be conducted by humans or computers to detect if there are secret messages hidden within the text. The decoder follows the reverse process of the encoder. After receiving the stegotext, we utilize an extracting algorithm combined with the text generation model to decode the corresponding bit sequence, and then convert the bit sequence into the secret messages.

A. Text Generation Model

In our proposed method, the story generation model proposed by [11] was employed as the text generation model. This model was a hierarchical neural network that can produce coherent and fluent text passages about prompts. First, a sequence-to-sequence (seq2seq) model was pre-trained with the prompt-story pairs in the train set. Then, a fusion mechanism [13] was applied by estimating a second model on top of the pre-trained seq2seq model to improve the dependency between prompts and stories. For modeling long documents efficiently, a novel gated self-attention mechanism was introduced which allowed the model to condition on its previous outputs at different time-scales.

At the generation stage, this model adopted a top-K random sampling scheme. At each time step, the model first calculated the probabilities of all word candidates, and then randomly sampled one word from the K most likely candidates according to their probabilities. This top-K random sampling scheme can generate longer and more diverse stories than beam search [11], [14], [15].

B. Embedding Algorithms

At the encoding stage of our proposed method, the text generation model first calculates the probabilities of all word candidates at each word position. Then, the embedding algorithm is applied instead of top-K random sampling to select the appropriate word from the candidates according to the bit sequence of secret messages.

1) Fixed-length coding (FLC) and variable-length-coding (VLC): The FLC and VLC algorithms which were initially proposed for embedding messages during deterministic text

generation [9] are also applied to the story generation model for linguistic steganography. FLC and VLC encodes the top-K word candidates at each word position using a balanced binary tree or a Huffman tree as shown in Fig. 2. Then, the appropriate leaf node is selected according to the bitstream of secret messages and the word candidate at this leaf node is the generated word at current position. Because the probabilities of word candidates are considered when building Huffman trees, VLC achieved better steganography performance than FLC despite of its higher complexity in previous study [9].

The original FLC and VLC algorithms use the value of *K*, i.e., the number of word candidates, to control their embedding rate. However, this number is usually fixed in sampling-based text generation. Thus, a word filtering strategy based on Kullback-Leibler divergence (KLD) is proposed to control the embedding rate of FLC and VLC in our proposed method. Specifically, the KLD between the probability distribution of word candidates and an ideal uniform distribution is calculated at each word position. A high KLD means that the candidate distribution is sharp and it is more likely to select the candidates with low probabilities by applying embedding algorithms instead of random sampling, which makes the steganalysis easier. Therefore, only the word positions whose KLDs are smaller than a threshold are utilized for hiding information. Otherwise, the random sampling strategy for generating non-steganography text is carried out as shown in Fig. 2. By modifying the threshold, the embedding rates of FLC and VLC can be controlled.

2) Arithmetic coding (AC): Recently, a method of provably secure steganography on generative media was proposed [12]. By considering message embedding as a dual problem of source coding, this method adopted arithmetic coding (AC) to embed secret messages according to the probability distributions given by generative models. Experiments on a generative model for audio signals were conducted to confirm the effectiveness of this method. In this paper, we apply the idea of provably secure steganography to linguistic steganography and employs the AC algorithm to generate stegotext.

Message Embedding The process of embedding secret messages corresponds to arithmetic decoding. At first, a bit sequence **B** = $b_1b_2b_3...b_L$ ($b_i \in \{0,1\}$) is converted into a binary decimal *q* by adding "0." as a prefix (e.g., 101010101 $\Rightarrow$ 0.101010101). The real value of *q* is

$$q = \sum_{i=1}^{L} b_i \cdot 2^{-i}. \tag{1}$$

Then, we start from the complete sampling interval [0, 1) and divide it into *K* subintervals in proportion to the probabilities of the top-*K* word candidates. The subinterval [b, e) which contains the value of *q* is selected. The candidate of this subinterval is selected as the output word. The subinterval [b, e) is further divided into *K* subintervals according to the candidate probabilities at the next word position for generating the next word. This process is repeated until a subinterval [b, e) satisfies the constraints of

$$\begin{cases} q + (0.5)^L \notin [b, e), \\ q - (0.5)^L \notin [b, e), \end{cases} \tag{2}$$

which means that the binary decimal *q* is the unique binary decimal of length L that can exist in the interval [b, e). In another word, the bit sequence **B** can be exactly recovered from the sequences of subintervals without any ambiguity.

Message Extracting The process of extracting messages corresponds to arithmetic encoding. The receiver owns the same text generation model as the sender. Given the received stegotext and the bit length *L*, the receiver determines a sequence of subintervals according to the words in the stegotext. The last subinterval leads to a unique binary decimal *q*. Finally, the bit sequence can be derived by a reverse process of Eq. (1).

It has been proved that the embedding rate of AC satisfies Eq. (3) on the premise of ensuring security [12].

$$H(C) \leq l_A < H(C) + \frac{2}{n}. \tag{3}$$

In our proposed linguistic steganography method, $H(C)$ is the average entropy of the distributions at all word positions, *n* is the length of the generated sequence and l_A is the average embedding rate (bit/word). According to this equation, when the generated sequence *n* grows, the embedding rate tends to be the average entropy at each word position. Therefore, a temperature-based strategy is designed to modify the probabilities of the top-K candidates at each word position and to control the embedding rate as

$$p_k = \frac{e^{o_k/T}}{\sum_{i=1}^{K} e^{o_i/T}}, k \in \{1, 2, ..., K\}, \tag{4}$$

where p_k is the calculated probability of the *k*-th word candidate, o_k represents the output of the generation model for the *k*-th word, and $T > 0$ is a temperature coefficient. We can see that a larger T leads to a higher entropy of distribution $\{p_1, ..., p_K\}$ and a higher embedding rate.

TABLE I
STATISTICS OF THE WRITINGPROMPTS DATASET [11].

Train Stories	272600
Test Stories	15138
Prompt Words	7.7M
Story Words	200M
Average Length of Prompts (words)	28.4
Average Length of Stories (words)	734.5

III. EXPERIMENTS

A. Experimental Conditions

The dataset for training the story generation model was collected from Reddit's WRITINGPROMPTS forum [11][1]. WRITINGPROMPTS is an online community where user are encouraged to write stories according to the submitted

[1] www.reddit.com/r/WritingPrompts.

TABLE II
EVALUATION RESULTS OF NON-STEGANOGRAPHY TEXTS AND ORIGINAL FLC/VLC ALGORITHMS, WHERE *Perplexity*, *fastText (%)* AND *Perplexity SVM (%)* MEASURE THE IMPERCEPTIBILITY OF STEGANOGRAPHY METHODS AS INTRODUCED IN SECTION III-C.

Embedding Algorithm	*Threshold*	*Embedding Rate (bit/word)*	*Perplexity*	*fastText (%)*	*Perplexity SVM (%)*
Non-steganography	-	0.00	18.40	-	-
FLC	$+\infty$	3.00	78.80	98.76	98.33
VLC	$+\infty$	1.98	27.67	78.78	75.50

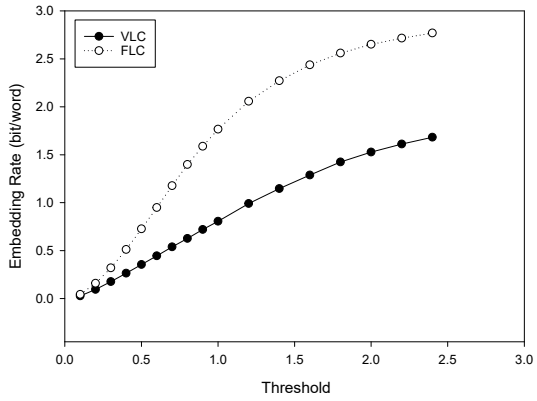

Fig. 3. The embedding rates of FLC and VLC with different KLD thresholds.

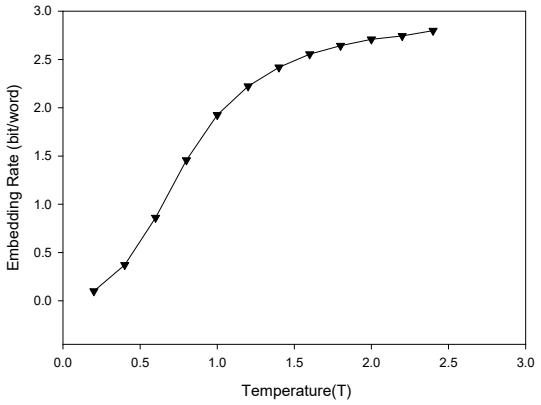

Fig. 4. The embedding rates of AC with various temperature coefficient.

story premise, or prompts. The prompts contain a number of different themes, lengths, and details. Each prompt can have multiple story responses and each story must have at least 30 words. The dataset included three years of prompts and their associated stories. More details about the dataset can be found in Table I. Its test set was adopted to generate the stegotext for evaluating different embedding algorithms in our experiments.

The trained model[2] was directly utilized to generate stories in our experiments. According to the default configurations [11], the maximum length of generated stories was set as 200 words and the temperature parameter T in Eq. (4) was set as 0.8 for generating non-steganography texts. The number of top candidates was set as $K = 8$ for random sampling and message embedding. A randomly generated binary sequence was adopted to represent secret messages in our experiments.

B. Evaluation on Hiding Capacity

Embedding Rate measures how much information we can hide in a stegotext. For linguistic steganography, the embedding rate is defined as the number of hidden bits per output word, i.e., bit/word. When the KLD threshold introduced in Section II-B1 was set as infinite, the embedding rates of the FLC and VLC with KLD-based word filtering were the same as the original FLC and VLC. Under this circumstances, the embedding rate of FLC was 3 bits/word since $K = 8$. The embedding rate of VLC was calculated as the average number of bits embedded in each word of the generated text and the result is shown in Table II.

Fig. 3 shows the influence of KLD threshold on the embedding rates of FLC and VLC. We can see that as the threshold

[2]www.github.com/pytorch/fairseq.

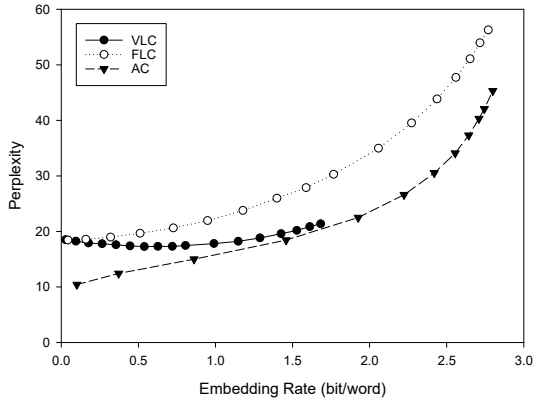

Fig. 5. The perplexity of FLC and VLC with KLD-based filtering and AC at different embedding rates.

increases, the embedding rate increases accordingly. Fig. 4 shows the influence of the temperature coefficient in Eq. (4) on the embedding rates of AC. We can see that that as the T value increases, the embedding rate also increases accordingly. As mentioned in Section III-A, the T value for generating non-steganography texts was set as 0.8 [11]. When $T = 0.8$ was adopted in the AC-based message embedding, the embedding rate was 1.45 bits/word as shown in Fig. 4.

C. Evaluation on Imperceptibility

First, we measured the perplexities [16] of the story texts generated by different embedding algorithms. Perplexity is commonly used to evaluate the quality of language models, and it reflects how correctly the model can produce current word given preceding ones. Lower perplexity indicates a better

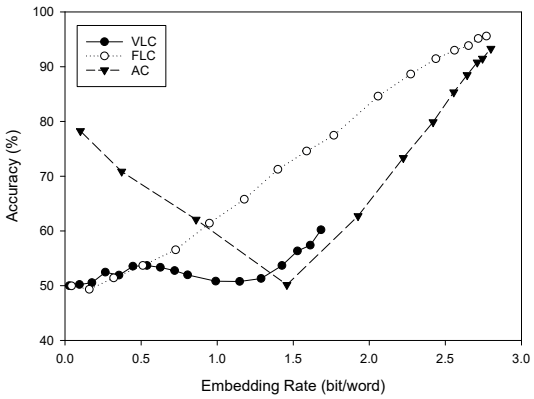

Fig. 6. Test set detection accuracies of perplexity-SVM steganalysis [17].

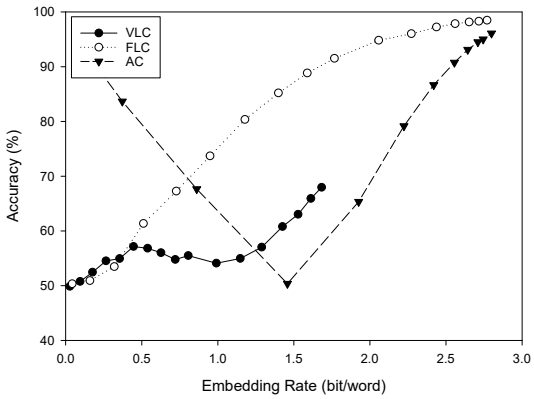

Fig. 7. Test set detection accuracies of fastText steganalysis [18].

matching between the text and the language model. Here, the text generation model was treated as the language model, and the perplexity of the texts with embedded messages was expected to be close to the perplexity of non-steganography texts which was shown in Table II.

Fig. 5 shows the relationship between the embedding rate (bit/word) and the perplexity for different embedding algorithms. We can see that the perplexity of FLC became much larger than that of non-steganography texts as the embedding rate increased. For VLC, the perplexity decreased slightly at first and then gradually increased. According to Fig. 3, when the KLD threshold was 0, the embedding rate of VLC was 0 and its perplexity in Fig. 5 was the same as the one of non-steganography texts. When the embedding rate was small, the KLD-filtered word positions for embedding messages had uniform candidate distributions and the VLC algorithm tended to select word with higher probabilities than random sampling. This explained the initial decrease of VLC's perplexity in Fig. 5. With the increase of embedding rate, the filtered word positions had less uniform candidate distributions and the VLC algorithm may select words with low probabilities. This led to the gradual increase of VLC's perplexity in Fig. 5. However, its increase was still smaller than the one of FLC. For AC, the perplexity also increased with the rise of embedding rate. As discussed above, larger temperature coefficient in Eq. (4) led to higher embedding rate and more uniform candidate distributions. Thus, the chances of the word candidates with originally low probabilities became larger and the perplexity of generated texts increased. We also observe in Fig. 5 that the perplexities of AC were always lower than the ones of FLC and VLC with similar embedding rates.

Furthermore, we evaluated the ability of resisting steganalysis for the three embedding algorithms. Two steganalysis methods were adopted here, including the perplexity-SVM method [17] and the fastText method [18]. We first utilized the story generation model to produce 5000 non-steganography stories based on the prompts in the test set of story generation. These stories were used as negative samples. 80% of them were adopted for training and the remaining 20% were used as test samples. For each embedding algorithm, 5000 steganography stories were generated as positive samples and they were further divided into a training set and a test set. In order to implement the perplexity-SVM method, an N-gram language model [19] was first estimated using the training set of the story generation model. Then, we use the trained N-gram model to calculate the perplexities of the positive and negative samples mentioned above. For each embedding algorithm, SVM [20] model was built using the positive and negative training samples to classify a given text into stegotext or non-steganography text according to its perplexity calculated using the N-gram language model. When implementing the fastText method, a fastText [18] classifier was built for each embedding algorithm using the training set of positive and negative samples and was further evaluated on the positive and negative test sample. The test set detection accuracies of these two steganalysis methods for different embedding algorithms and embedding rates are shown in Fig. 6 and Fig. 7. For both steganalysis methods, an accuracy closer to 50% indicates a better ability of resisting steganalysis.

Comparing Fig. 6 with Fig. 7, we can see than fastText achieved better staganalysis performance than perplexity-SVM. When the embedding rate was low, the detection accuracies of steganography text generated by FLC and VLC were quite similar. When the embedding rate was higher than 0.5 bit/word, VLC demonstrated better performance than FLC. When the embedding rate was lower than 1.5 bits/word, the detection accuracies of VLC was less than 60% for both steganalysis methods. Comparing Fig. 6 and Fig. 7 with Table II, we can see that the KLD-based word filtering improved the imperceptibility of FLC and VLC at the cost of reduced embedding rates.

As shown in Fig. 6 and Fig. 7, the accuracy curves of the AC embedding algorithm were both V-shaped. The lowest points were obtained when $T = 0.8$ in Eq. 4 and the embedding rate was 1.45 bits/word. The lowest detection accuracies were 50.15% and 50.36% respectively in these two figures, which were very close to the ideal value of 50%. Increasing or decreasing the temperature coefficient both enlarged the differ-

TABLE III
SUBJECTIVE PREFERENCE SCORES (%) BETWEEN NON-STEGANOGRAPHY STORIES AND AC-BASED STEGANOGRAPHY STORIES.

	Preference Score (%)
Non-steganography	40.88
No preference	12.75
AC-based steganography	46.37

ences between the AC-based steganography text and the non-steganography text, and caused higher detection accuracies. Comparing AC with FLC and VLC in Fig. 6 and Fig. 7, we can see that the AC algorithm achieved better ability of resisting steganalysis than FLC and VLC when the embedding rate was higher than 1.45 bits/word.

D. Subjective Evaluation

A subjective preference test was conducted to compare the quality of non-steganography stories and the steganography stories generated by AC at the embedding rate of 1.45 bits/word. We randomly selected 80 prompts from the test set of story generation, and then generated non-steganography stories and AC-based steganography stories respectively based on these prompts. The subjective evaluation was conducted on the crowdsourcing platform of Amazon Mechanical Turk[3]. These 80 pairs of stories were evaluated by 10 human evaluators. Each pair included a non-steganography story and a steganography story, together with their prompt. Human evaluators were asked to choose which one was better or they had no preference based on their overall judgement on the relevance to the prompt, the naturalness and the fluency of story text. The average preference scores of the 10 evaluators are shown in Table III. The result of a t-test ($p > 0.01$) further confirmed that the difference between the subjective quality of the non-steganography stories and the AC-based steganography stories at the embedding rate of 1.45 bits/word was insignificant.

IV. CONCLUSION

In this paper, we have presented a linguistic steganography method which utilizes sampling-based language generation. A story generation model was adopted in our implementation as the text generation model and several embedding algorithms, including fixed-length coding, variable-length coding and arithmetic coding, are introduced into our proposed method. Experimental results show that AC achieved better balance between embedding rate and imperceptibility than FLC and VLC. At the embedding rate of 1.45 bits/word, our AC-based steganography method achieved ideal imperceptibility, and the subjective quality of its generated text is as good as the non-steganography one. In the future, we will explore other text generation models for linguistic steganography and improve the imperceptibility of our proposed method at high embedding rate.

[3]https://www.mturk.com.

ACKNOWLEDGMENTS

This work was partially supported by the National Nature Science Foundation of China (Grant No. U1636201 and 61871358).

REFERENCES

[1] J. Fridrich, *Steganography in digital media: principles, algorithms, and applications*. Cambridge University Press, 2009.
[2] M. J. Atallah, V. Raskin, C. F. Hempelmann, M. Karahan, R. Sion, U. Topkara, and K. E. Triezenberg, "Natural language watermarking and tamperproofing," in *International workshop on information hiding*. Springer, 2002, pp. 196–212.
[3] C.-Y. Chang and S. Clark, "Linguistic steganography using automatically generated paraphrases," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 591–599.
[4] C.-Y. Chang and S. Clark, "Adjective deletion for linguistic steganography and secret sharing," *Proceedings of COLING 2012*, pp. 493–510, 2012.
[5] C.-Y. Chang and S. Clark, "Practical linguistic steganography using contextual synonym substitution and a novel vertex coding method," *Computational linguistics*, vol. 40, no. 2, pp. 403–448, 2014.
[6] Y. Luo, Y. Huang, F. Li, and C. Chang, "Text steganography based on ci-poetry generation using markov chain model," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 10, no. 9, pp. 4568–4584, 2016.
[7] Y. Luo and Y. Huang, "Text steganography with high embedding rate: Using recurrent neural networks to generate chinese classic poetry," in *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*. ACM, 2017, pp. 99–104.
[8] T. Fang, M. Jaggi, and K. Argyraki, "Generating steganographic text with lstms," *ACL 2017*, p. 100, 2017.
[9] Z.-L. Yang, X.-Q. Guo, Z.-M. Chen, Y.-F. Huang, and Y.-J. Zhang, "Rnn-stega: Linguistic steganography based on recurrent neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1280–1295, 2018.
[10] 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.
[11] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 889–898.
[12] K. Chen, H. Zhou, D. Hou, H. Zhao, W. Zhang, and N. Yu, "Provably secure steganography on generative media," *arXiv preprint arXiv:1811.03732*, 2018.
[13] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold fusion: Training seq2seq models together with language models," *Proc. Interspeech 2018*, pp. 387–391, 2018.
[14] A. K. Vijayakumar, M. Cogswell, R. R. Selvaraju, Q. Sun, S. Lee, D. Crandall, and D. Batra, "Diverse beam search: Decoding diverse solutions from neural sequence models," *arXiv preprint arXiv:1610.02424*, 2016.
[15] L. Shao, S. Gouws, D. Britz, A. Goldie, B. Strope, and R. Kurzweil, "Generating long and diverse responses with neural conversation models," *arXiv preprint arXiv:1701.03185*, 2017.
[16] J. H. Martin and D. Jurafsky, *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson/Prentice Hall Upper Saddle River, 2009.
[17] P. Meng, L. Huang, Z. Chen, W. Yang, and D. Li, "Linguistic steganography detection based on perplexity," in *2008 International Conference on MultiMedia and Information Technology*. IEEE, 2008, pp. 217–220.
[18] A. Joulin, E. Grave, and P. B. T. Mikolov, "Bag of tricks for efficient text classification," *EACL 2017*, p. 427, 2017.
[19] V. Siivola and B. L. Pellom, "Growing an n-gram language model," in *Ninth European Conference on Speech Communication and Technology*, 2005.
[20] T. Evgeniou and M. Pontil, "Support vector machines: Theory and applications," in *Advanced Course on Artificial Intelligence*. Springer, 1999, pp. 249–257.