EXPLOITING LANGUAGE MODEL FOR EFFICIENT LINGUISTIC STEGANALYSIS

Biao Yi, Hanzhou Wu, Guorui Feng and Xinpeng Zhang

Shanghai University, Shanghai 200444, China

ABSTRACT

Recent advances in linguistic steganalysis have successively applied CNN, RNN, GNN and other efficient deep models for detecting secret information in generative texts. These methods tend to seek stronger feature extractors to achieve higher steganalysis effects. However, we have found through experiments that there actually exists significant difference between automatically generated stego texts and carrier texts in terms of the conditional probability distribution of individual words. Such kind of difference can be naturally captured by the language model used for generating stego texts. Through further experiments, we conclude that this ability can be transplanted to a text classifier by pre-training and fine-tuning to improve the detection performance. Motivated by this insight, we propose two methods for efficient linguistic steganalysis. One is to pre-train a language model based on RNN, and the other is to pre-train a sequence autoencoder. The results indicate that the two methods have different degrees of performance gain compared to the randomly initialized RNN, and the convergence speed is significantly accelerated. Moreover, our methods achieved the best performance compared to related works, while providing a solution for real-world scenario where there are more cover texts than stego texts.

Index Terms— Linguistic steganalysis, language model, natural language processing, deep learning, security.

1. INTRODUCTION

Steganography [1] embeds secrets in public carriers without being easily noticed by the monitor. The carrier can be generally arbitrary media. As an important carrier for people to communicate with each other in daily life, natural language is actually quite suitable for steganography, which is referred to as *linguistic steganography (LS)*. The advantage is that LS can be easily concealed by the huge number of social activities.

Conventional LS mainly includes two categories: *modification based* and *generation based*. The former modifies a given text carrier to realize the embedding of secret information such as [2, 3, 4, 5]. Since a text is often highly coded, a significant disadvantage for modification based methods is that the maximum size of embeddable payload is small. The latter trains a language model (LM) on a corpus so that secret information can be embedded during text generation according to the LM. Compared to modification based methods, this category allows more data to be embedded. Moreover, texts generated by a well-trained LM are more semantically natural, implying that generation based methods tend to generate stego texts with higher quality. It motivates many scholars to propose generation based LS methods [6, 7, 8, 9, 10].

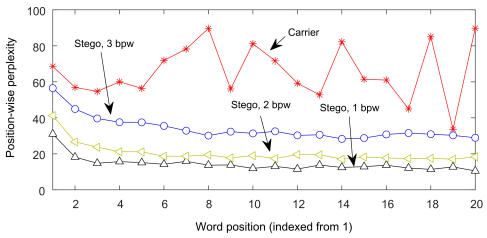

Fig. 1. Position-wise perplexities for carrier/stego texts.

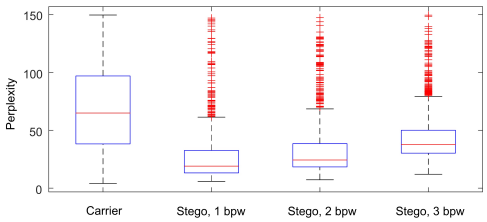

Fig. 2. Text-level perplexity distribution for carriers/stegos.

As the opposite of steganography, a goal of steganalysis is to detect whether there is secret data embedded in the media. We urgently need to develop a steganalysis system with efficient detection capabilities to deal with threats caused by generative LS. Early steganalysis methods [11, 12, 13] feed manually-crafted features into an ordinary classifier such as support vector machine for text classification. They are no longer sufficient to detect generative LS. Recently, increasing works use deep learning [14] for efficient linguistic steganalysis. It enables a deep neural network to automatically extract the discriminative features for classification by an end-to-end fashion. E.g., Yang *et al.* [15] map the words in a given text to a semantic space and extract the correlation features between words using a hidden layer for final classification. Wen *et al.*

It was supported by National Natural Science Foundation of China under Grant No. 61902235 and Shanghai "Chen Guang" Program under Grant No. 19CG46. Corresponding author: Hanzhou Wu (E-mail: h.wu.phd@ieee.org)

[16] use convolution kernels of different sizes to extract text features to achieve steganalysis. Yang *et al.* [17] use recurrent neural network (RNN) [18] to mine the distribution difference between stego texts and carrier texts for steganalysis. In recent, Wu *et al.* [19] successfully apply graph neural network (GNN) to linguistic steganalysis.

The above methods treat linguistic steganalysis as an ordinary text classification task, focusing on improving the model architecture and finding stronger feature extractors. The consequence of this research trend is that the development of linguistic steganalysis always relies on the development of deep feature representation technologies. However, unlike LS that directly modifies given unrelated carrier texts, generative LS should pre-train a LM and then use the LM for embedding and extraction, causing the generated texts to inevitably expose the statistical characteristics to the LM, implying that uniting LM may bring us a new direction to move forward.

In this paper, we propose two methods for efficient linguistic steganalysis. One is to pre-train a language model based on RNN, and the other is to pre-train a sequence autoencoder. Experiments show that both outperform the randomly initialized RNN and the best detection performance is achieved in strict data-balanced scenario. This work has verified the effectiveness of pre-training LM for linguistic steganalysis, while firstly providing a solution for real-world scenario where there are more cover texts than stego texts.

The rest will be organized as follows. We introduce the proposed method in Section 2. Experiments and analysis are provided in Section 3. We conclude this paper in Section 4.

2. PROPOSED METHOD

2.1. Motivation

A well-trained LM used for LS naturally exposes statistical characteristics of stego texts. To explain this, we train a LM on MOVIE [20] and use the method in [9] (with fixed-length coding) for generating stego texts with the trained LM. We randomly choose 1,000 stego texts generated by the trained LM for each data embedding rate and 1,000 carrier texts from the dataset. Notice that, the used LM architecture is the same as [9]. We use the well-trained LM to determine the perplexity [7] for a text $\mathbf{w} = (w_1, w_2, ..., w_n)$, where n is the number of words. The perplexity can be expressed as:

$$\text{Perp}(\mathbf{w}) = 2^{-\frac{1}{n}\cdot\log_2 \Pr(w_1,w_2,...,w_n)}, \tag{1}$$

where $\Pr(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} \Pr(w_i|w_{i-1}, w_{i-2}, ..., w_1)$ can be estimated by the LM. For each word position $i \in [1, n]$, we can also determine a perplexity as $\text{Perp}(\mathbf{w}_i) = 2^{-\log_2 \Pr(w_i|w_{i-1},w_{i-2},...,w_1)}$, which is defined as *position-wise perplexity*. Fig. 1 shows the (mean) position-wise perplexities. E.g., for the stego texts embedded with 1 bpw (bits per word) and a specified position, we compute the position-wise perplexity. The mean perplexity is then used as the result.

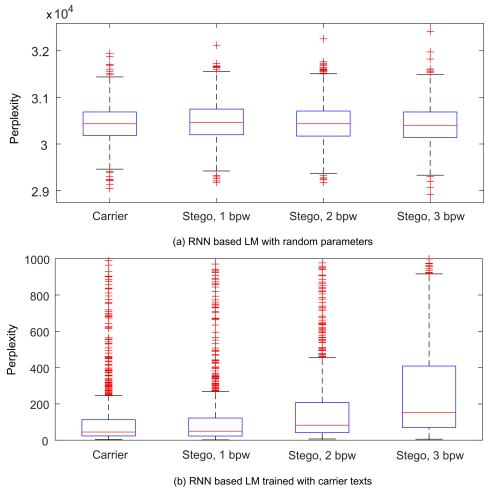

Fig. 3. Text-level perplexity distribution for two RNNs.

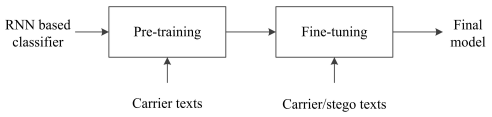

Fig. 4. RNN based pre-training strategy for steganalysis.

As shown in Fig. 1, there is clear difference between stego texts and carrier texts in terms of the conditional probability distribution of individual words, and the LM has the ability to characterize the difference. We show the perplexity distribution corresponding to Eq. (1) in Fig. 2. It could be inferred that we can distinguish most stego texts from normal ones by choosing a threshold (though the performance is not the best).

In mainstream steganalysis frameworks, the LM applied to LS is not available to the steganalyzer, who only holds some labeled carrier/stego texts for training a classifier. The practical reason is that a steganographic system is deemed secure (to a certain extent) if it manages to fool the steganalyzer even under such disadvantageous condition. It inspires us to exploit carrier texts for giving some prior knowledge to a steganalysis model by pre-training [21, 23, 24]. The pre-trained parameters are used for parameter initialization for the subsequent steganalysis task to improve the detection performance.

To explain that pre-training with carrier texts indeed has the ability to capture statistical characteristics of stego texts, we use two RNNs with different parameters to represent two LMs for analyzing the perplexities. One was randomly initialized. The other was trained with 10,000 carrier texts randomly chosen from the aforementioned dataset. The LM architecture refers to Ref. [9]. Fig. 3 shows the results tested on 1,000 carriers and 1,000 stegos mentioned above. There was no intersection between carrier texts used for training and carrier texts for testing. In Fig. 3, there is no distinguishable perplexity difference between carrier texts and stego texts for the randomly initialized RNN. However, there is remarkable

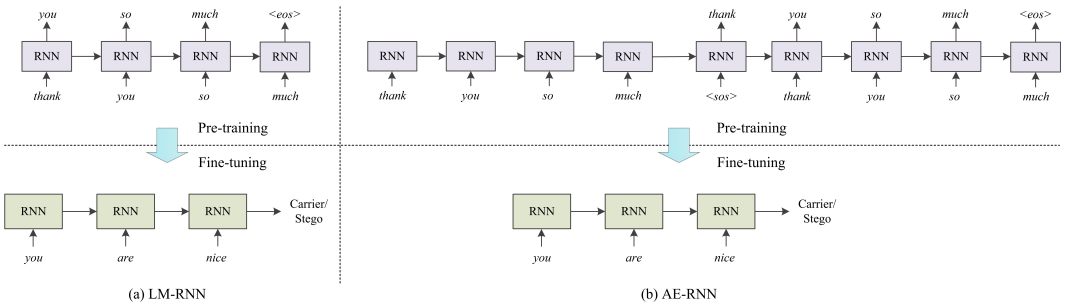

Fig. 5. Explanation for training (a) LM-RNN model and (b) AE-RNN model. The parameters are randomly initialized during the pre-training phase. The pre-trained parameters are then used to initialize the corresponding RNN based steganalysis model.

difference for the trained RNN, indicating that pre-training can learn prior knowledge for detecting stego texts, and thus has potential to enhance the steganalysis performance.

2.2. Recurrent Neural Network

We propose two pre-training methods to obtain superior steganalysis performance. One is to pre-train a traditional LM based on RNN (denoted by LM-RNN), and the other is to use a sequence autoencoder [25] to encode a text into a vector and then produce the prediction result (denoted by AE-RNN).

RNN is the most popular neural network structure in natural language processing since its recurrent structure is suitable for processing sequences of variable lengths. The original RNN has the problem with exploding or vanishing gradients, making it difficult to learn long-term dependency. To this end, we use a variant of RNN, i.e., long and short-term memory (LSTM) [18], that can effectively deal with the above problem. We refer the reader to [18, 7] for details about LSTM.

A common strategy of RNN is to project the hidden vector into the output space, and then use "softmax" [14] to convert it into the probabilistic space for classification or other purposes. For example, to train a LM, we project the hidden vector at each moment into the dictionary-sized output space to predict the next word. To perform steganalysis, we project the hidden state at the last moment to the output vector with a size of 2, thereby classifying a text as carrier or stego.

2.3. Pre-training and Linguistic Steganalysis

We introduce two RNN based methods for steganalysis. As shown in Fig. (4, 5), both have two steps. First, carrier texts in the training set are used for pre-training. Then, the pre-trained parameters are used to initialize the model to perform regular steganalysis. Referring to Fig. 5, there are two ways for pre-training. One way is to train a traditional LM. The other is to train a sequence autoencoder, i.e., using the encoder to encode the text into a vector (that is, the hidden vector at the last moment), and then using this vector as the initial hidden vector to reconstruct the input text with the decoder. One thing to note is that the encoder and the decoder use the same LSTM network, which also means that their weights are the same.

In this study, we use the pre-trained LSTM parameters with useful prior knowledge for parameter initialization of the steganalysis model, which can improve the steganalysis performance and the convergence efficiency compared to random initialization. The specific process is briefly described as follows. We input a text into the pre-trained LSTM network and project the hidden vector at the last moment to the output vector containing two elements using a fully-connected layer. The softmax function is then used to further transform the real vector into a probability vector. During training, we minimize the cross entropy between the prediction distribution and the ground-truth distribution. During testing, a text is classified as carrier or stego based on the prediction result.

3. EXPERIMENTAL RESULTS AND ANALYSIS

The tested LS methods include RNN-Steg [9] and Bins [6]. For RNN-Steg, two information encoding methods, i.e., fixed-length coding (FLC) and variable-length coding (VLC), were tested in the text generation stage. The details of FLC and VLC can be found in [9]. The two datasets MOVIE [20] and TWITTER [26] were used to train LMs. The mean lengths of sentences for MOVIE and TWITTER are around 20 and 10. The numbers of sentences for them approach 1.3×10^6 and 2.6×10^6. Both datasets have near 5×10^4 words. Each trained LM was used to produce 1×10^4 stego texts with the corresponding embedding rate. These stego texts together with carrier texts (randomly chosen from the original corpus) were used for steganalysis. For each dataset, 70% texts were used for training and 30% for testing. In addition, 10% training texts were used for validation.

For pre-training, the hyperparameters of two pre-training methods were the same. The tokenizer used in this paper was consistent with that in BERT. A word was mapped to a 128-D vector through an embedding layer. The dropout function after the embedding layer took a retention probability of 0.5. The number of layers of RNN was set to 2, the hidden vector was 256-D. The learning rate was 10^{-3}, and the Adam

Table 1. Detection results using different steganalysis models under different experimental conditions.

Steganalysis model ↦			FCN [15]		CNN [16]		GNN [19]		RNN [17]		LM-RNN		AE-RNN	
Dataset	LS method	bpw	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
TWITTER [26]	Bins [6]	1.000	0.8142	0.8162	0.9072	0.9083	0.9125	0.9134	0.9102	0.9119	0.9082	0.9074	**0.9175**	**0.9170**
		2.000	0.7833	0.7835	0.8953	0.8977	0.9007	0.9020	0.8998	0.9027	0.9017	0.9020	**0.9057**	**0.9068**
		3.000	0.7647	0.7735	0.8953	0.8955	0.9013	0.8998	0.9002	0.8998	0.8988	0.8980	**0.9078**	**0.9093**
	FLC [9]	1.000	0.7992	0.7908	0.8952	0.8966	0.9023	0.9039	0.9000	0.9015	0.9085	0.9066	**0.9112**	**0.9120**
		2.000	0.7657	0.7696	0.8937	0.8960	0.8973	0.8975	0.8820	0.8850	0.8995	0.9004	**0.9035**	**0.9040**
		3.000	0.7393	0.7514	0.8935	0.8937	0.8890	0.8862	0.8948	0.8988	**0.9013**	**0.9028**	0.8965	0.8996
	VLC [9]	1.000	0.7947	0.7902	0.8943	0.8961	0.9068	0.9056	0.9043	0.9049	**0.9082**	**0.9074**	0.9043	0.9058
		2.150	0.7685	0.7708	0.8808	0.8825	0.8907	0.8872	0.8813	0.8877	0.8980	0.8969	**0.9068**	**0.9070**
		3.147	0.7518	0.7509	0.8842	0.8857	0.8867	0.8857	0.8877	0.8915	0.8960	0.8949	**0.9027**	**0.9020**
MOVIE [20]	Bins [6]	1.000	0.8973	0.8935	0.9612	0.9616	0.9535	0.9541	0.9538	0.9541	**0.9633**	**0.9635**	0.9627	0.9630
		2.000	0.8575	0.8570	0.9390	0.9400	0.9428	0.9431	0.9412	0.9424	**0.9515**	**0.9515**	0.9488	0.9485
		3.000	0.8052	0.8032	0.9255	0.9278	0.9155	0.9148	0.9152	0.9177	0.9335	0.9339	**0.9352**	**0.9355**
	FLC [9]	1.000	0.8848	0.8825	0.9492	0.9503	0.9540	0.9543	0.9527	0.9535	0.9592	0.9590	**0.9618**	**0.9615**
		2.000	0.8327	0.8317	0.9357	0.9368	0.9358	0.9360	0.9232	0.9268	0.9452	0.9455	**0.9498**	**0.9504**
		3.000	0.7815	0.7850	0.9183	0.9208	0.9210	0.9200	0.9027	0.9080	0.9302	0.9294	**0.9315**	**0.9327**
	VLC [9]	1.000	0.8783	0.8739	0.9492	0.9498	0.9523	0.9524	0.9467	0.9479	0.9608	0.9605	**0.9643**	**0.9645**
		2.215	0.8358	0.8346	0.9307	0.9326	0.9387	0.9386	0.9330	0.9347	0.9433	0.9429	**0.9475**	**0.9475**
		3.260	0.8018	0.7957	0.9225	0.9242	0.9228	0.9216	0.9168	0.9201	**0.9397**	**0.9404**	0.9365	0.9365

Table 2. Mean accuracies due to different pre-training sizes.

Steganalysis model ↦		LM-RNN	AE-RNN
Dataset	Number of pre-training samples		
MOVIE	6.3×10^3	0.9474	0.9487
	1.2×10^4	0.9518	0.9510
	1.8×10^4	0.9535	0.9528
TWITTER	6.3×10^3	0.9022	0.9062
	1.2×10^4	0.9086	0.9124
	1.8×10^4	0.9120	0.9169

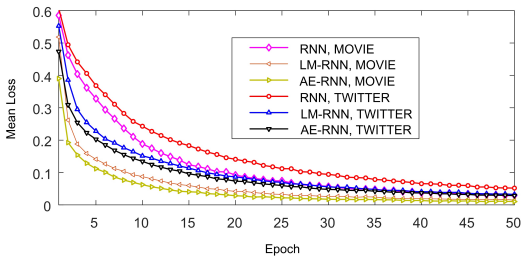

Fig. 6. Mean loss curves. For each point, we collect nine loss values (since there are three LS algorithms and three different embedding rates) and determine the mean value as the result.

[27] optimizer was used. For pre-training, the batch size was 128 and the number of epochs was 50. After pre-training, we directly used the pre-trained model parameters as the initial values for steganalysis training, so the hyperparameters were unchanged, but a fully connected layer was added after the last hidden vector to map it to a vector sized 2, from which we can get a probability vector by softmax. Two common indicators [16]: Accuracy (Acc) and F1 score (F1), were used.

We compare our work with GNN [19], fully connected network (FCN) [15], convolutional neural network (CNN) [16] and RNN [17]. To show the improvement brought by our models when compared to the randomly initialized RNN model, the hyperparameter settings for the RNN model used in [17] are consistent with ours. Table 1 shows the results, from which we conclude that: First, the proposed LM-RNN and AE-RNN have achieved the best detection results, which can verify the feasibility and superiority of proposed work. Second, by comparing with the randomly initialized RNN [17], it is inferred that the performance improvement is significant. It indicates that pre-training with carrier texts indeed improves the performance. In addition, AE-RNN is superior to LM-RNN. The reason may be that the traditional LM training focuses on predicting the next word, but AE-RNN not only does this, but also learns the entire sentence information and therefore has stronger modeling capabilities.

We have also tested the performance of randomly initialized RNNs and pre-trained RNNs in terms of loss convergence efficiency. As shown in Fig. 6, pre-training leads to significant improvement. In fact, since carrier texts can be shared before performing steganalysis on the same dataset, we only need to pre-train the RNN model once to speed up various steganalysis for different steganographic methods and different embedding rates, which greatly improves the efficiency.

In practice, we may be able to collect more texts for pre-training. We use more carrier texts for pre-training to evaluate its impact on the steganalysis performance. Table 2 shows the mean accuracy values due to different sizes of the pre-training set, e.g., 0.9474 is the mean value of nine accuracy values of LM-RNN (tested on the MOVIE dataset) shown in Table 1. There is no intersection between pre-training set and testing set. It can be inferred that the steganalysis performance can be further improved by using more carrier texts for pre-training, which inspires us to collect carrier texts as many as possible in practice so as to achieve superior steganalysis performance.

4. CONCLUSION

We found through experiments that the well-trained language model has the ability to model the distribution difference between carrier texts and steganographic texts. This ability can be passed to the classifier through pre-training and finetuning to improve the performance of subsequent steganalysis. Motivated by this important insight, we propose two methods for enhancing the performance of RNN-based steganalysis models. The experimental results have shown that the two methods have achieved the best performance compared to related works and can greatly improve the training efficiency.

5. REFERENCES

[1] J. Fridrich, "Steganography in digital media: principles, algorithms, and applications," *Cambridge Univ. Press*, 2009.

[2] L. Huo and Y. Xiao, "Synonym substitution-based steganographic algorithm with vector distance of two-gram dependency collocations," In: *Proc. IEEE Int. Conf. Computer Commun.*, pp. 2776-2780, 2016.

[3] H. Hu, X. Zuo, W. Zhang and N. Yu, "Adaptive text steganography by exploring statistical and linguistical distortion," In: *Proc. IEEE Int. Conf. Data Science in Cyberspace*, pp. 145-150, 2017.

[4] Y. Liu, X. Sun, C. Gan and H. Wang, "An efficient linguistic steganography for Chinese text," In: *Proc. IEEE Int. Conf. Multimed. Expo*, pp. 2094-2097, 2007.

[5] M. Topkara, U. Topkara and M. J. Atallah, "Information hiding through errors: a confusing approach," In: *Proc. SPIE, Security, Steganography, & Watermarking of Multimed. Cont. IX*, vol. 6505, pp. 321-332, 2007.

[6] T. Fang, M. Jaggi and K. Argyraki, "Generating steganographic text with LSTMs," *arXiv preprint arXiv:1705.10742*, 2017.

[7] H. Kang, H. Wu and X. Zhang, "Generative text steganography based on LSTM network and attention mechanism with keywords," In: *Proc. IS&T Electronic Imaging, Media Watermarking, Security & Forensics*, pp. 291-1-291-8(8), 2020.

[8] Z. Ziegler, Y. Deng and A. Rush, "Neural linguistic steganography," *arXiv:1909.01496*, 2019.

[9] Z. Yang, X. Guo, Z. Chen, Y. Huang and Y. Zhang, "RNN-Stega: linguistic steganography based on recurrent neural networks," *IEEE Tran. Inf. Forensics Security*, 14(5): 1280-1295, 2018.

[10] Z. Yang, S. Zhang, Y. Hu, Z. Hu and Y. Huang, "VAE-Stega: Linguistic steganography based on variational auto-encoder," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 880-895, 2021.

[11] Z. Chen, L. Huang, H. Miao, W. Yang and P. Meng, "Steganalysis against substitution-based linguistic steganography based on context clusters," *Computers & Electr. Engineering*, 37(6): 1071-1081, 2011.

[12] L. Xiang, X. Sun, G. Luo and B. Xia, "Linguistic steganalysis using the features derived from synonym frequency," *Multimed. Tools Appl.*, 71(3): 1893-1911, 2014.

[13] P. Meng, L. Hang, Z. Chen, Y. Hu and W. Yang, "STBS: A statistical algorithm for steganalysis of translation-based steganography," In: *Proc. Int. Workshop Inf. Hiding*, pp. 208-220, 2010.

[14] I. Goodfellow, Y. Bengio and A. Courville, "Deep learning," *The MIT Press*, 2016.

[15] Z. Yang, Y. Huang and Y. Zhang, "A fast and efficient text steganalysis method," *IEEE Signal Process. Lett.*, 26(4): 627-631, 2019.

[16] J. Wen, X. Zhou, P. Zhong and Y. Xue, "Convolutional neural network based text steganalysis," *IEEE Signal Process. Lett.*, 26(3): 460-464, 2019.

[17] Z. Yang, K. Wang, J. Li, Y. Huang and Y. Zhang, "TS-RNN: Text steganalysis based on recurrent neural networks," *IEEE Signal Process. Lett.*, 26(12): 1743-1747, 2019.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neur. Computation*, 9(8): 1735-1780, 1997.

[19] H. Wu, B. Yi, F. Ding, G. Feng and X. Zhang, "Linguistic steganalysis with graph neural networks," *IEEE Signal Process. Lett.*, vol. 28, pp. 558-562, 2021.

[20] A. L. Mass, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng and C. Potts, "Learning word vectors for sentiment analysis," In: *Proc. Annual Meeting of the Association for Computational Linguistics*, pp. 142-150, 2011.

[21] J. Devlin, M. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv:1810.04805*, 2018.

[22] W. Peng, J. Zhang, Y. Xue and Z. Yang, "Real-time text steganalysis based on multi-stage transfer learning," *IEEE Signal Process. Lett.*, vol. 28, pp. 1510-1514, 2021.

[23] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, 2019.

[24] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," *arXiv preprint arXiv:1511.01432*, 2015.

[25] I. Sutskever, O. Vinyals and Q. V. Le, "Sequence to sequence learning with neural networks," *arXiv:1409.3215*, 2014.

[26] A. Go, R. Bhayani and L. Huang, "Twitter sentiment classification using distant supervision," *Stanford CS224N Project Reports*, 6 pages, 2009.

[27] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," *arXiv:1412.6980*, 2014.