Addressing Segmentation Ambiguity in Neural Linguistic Steganography

Jumon Nozaki **Yugo Murawaki**
Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
nozaki@sap.ist.i.kyoto-u.ac.jp murawaki@i.kyoto-u.ac.jp

Abstract

Previous studies on neural linguistic steganography, except Ueoka et al. (2021), overlook the fact that the sender must detokenize cover texts to avoid arousing the eavesdropper's suspicion. In this paper, we demonstrate that segmentation ambiguity indeed causes occasional decoding failures at the receiver's side. With the near-ubiquity of subwords, this problem now affects any language. We propose simple tricks to overcome this problem, which are even applicable to languages without explicit word boundaries.

1 Introduction

Lying at the intersection of information security and natural language processing, linguistic steganography is the practice of hiding information in cover texts (Simmons, 1984; Anderson and Petitcolas, 1998; Bennett, 2004). Formally, the sender *Alice* encodes a secret message, usually in the form of a bit sequence, into a cover text, while the receiver *Bob* decodes the message. The most important requirement is *security*: The cover text must be so natural that even if transmitted in a public channel, it does not arouse the suspicion of the eavesdropper *Eve*. In fact, steganography engages in an arms race with *steganalysis*, the practice of detecting the presence of secret messages (Fridrich, 2009). With the security requirement fulfilled, we also want to increase *payload capacity*, the size of the secret message relative to the size of the cover text (Chang and Clark, 2014).

Compared with dominant cover media in steganography, such as images, videos, and audio (Fridrich, 2009), texts are characterized by a low degree of redundancy. This makes it particularly challenging to enumerate natural variations of text into which bit chunks are encoded (Chang and Clark, 2014). Nevertheless, this difficulty is surmounted to some degree by powerful neural language models (LMs) for their ability to suggest probable next tokens in a context-aware manner (Fang et al., 2017), and the research focus has shifted towards increasing payload capacity (Dai and Cai, 2019; Ziegler et al., 2019; Shen et al., 2020; Zhang et al., 2021).

Previous studies, however, overlook the fact that Alice must detokenize texts before sending them to a public channel; Otherwise they arouse Eve's suspicion. Ueoka et al. (2021) were the first to point out that Bob may fail to recover the original tokens from detokenized texts, leading to decoding failures. While segmentation ambiguity has been a vexing problem for *scriptio continua*, or writing systems without explicit word boundaries (e.g., Chinese and Japanese), the near-ubiquitous use of subwords implies that it now affects any language. For example, suppose that Alice generates the English sequence "*un ##us ##able*". Detokenized into "*unusable*", it is unfortunately re-tokenized into "*un ##usable*" by Bob (Figure 1 (top)).

While recent proposals are flawed, the fact that the problem went unnoticed till Ueoka et al. (2021) suggests that the errors occur only infrequently. This leads us to the following question: How often do decoding failures occur? We expect that they affect morphologically rich languages and *scriptio continua* more severely than English. We report our experimental results using Russian and Japanese in addition to English.

Although Ueoka et al. (2021) proposed a simple solution for their edit-based method, it is not applicable to LM-based (generation-based) methods. This motivates us to address the second question: How can generation-based methods overcome segmentation ambiguity?

In this paper, we propose a combination of simple tricks to ensure that Bob recovers the same tokens as Alice (Figure 1 (bottom)). The proposed method can be applied not only to subword-based LMs but also to *scriptio continua*, as we demonstrate for Japanese. Our code is available at https://github.com/jumon/himitsu.

Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 109–116
November 20–23, 2022.

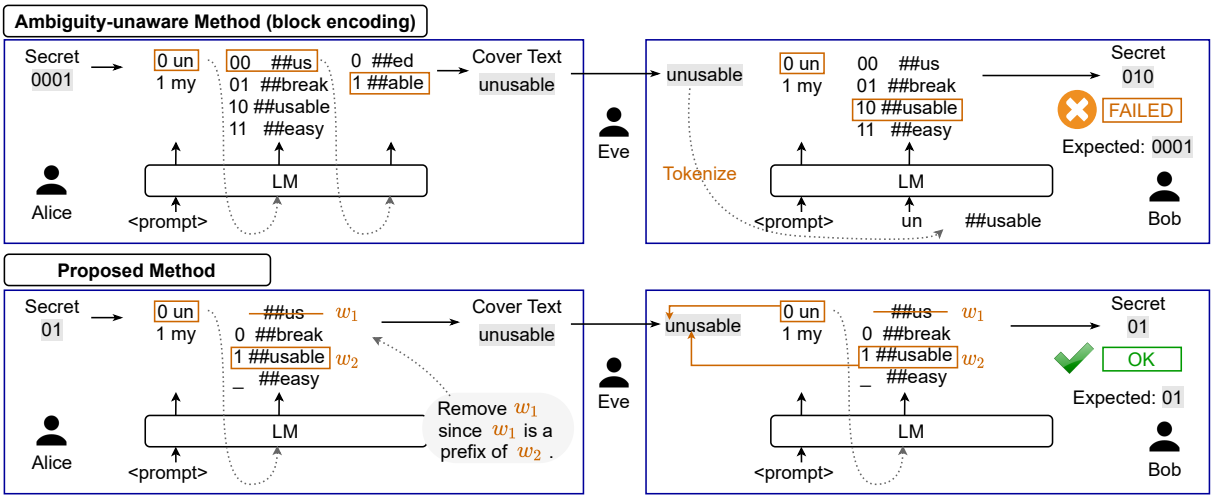

Figure 1: Overview of neural linguistic steganography based on an ambiguity-unaware method (top) and the proposed method (bottom). Starting with some introductory context (prompt), the sender Alice iteratively uses a language model (LM) to propose probable next tokens, assigns bit chunks to them, and selects a token corresponding to the secret message. The receiver Bob tries to decode the secret message but may fail with the ambiguity-unaware method because the original tokens are not always recovered from the detokenized cover text. The proposed method guarantees correct decoding by performing stepwise tokenization at Bob's side and by resolving ambiguities.

2 Related Work

2.1 Finite Word-level Vocabularies

Before the widespread adoption of subwords, which coincided with the invention of the Transformer architecture (Vaswani et al., 2017), recurrent neural network-based (RNN-based) LMs were accompanied by a finite word-level vocabulary (Bengio et al., 2003). Vocabulary selection was usually based on frequencies in the training data, and low-frequency words were replaced with the special token `UNK`. Applying this technique to linguistic steganography (Zhang et al., 2021) is impractical because `UNK` is a clear signal of automatic generation and hence is subject to steganalysis.

Oddly enough, previous studies exploring RNN LMs for linguistic steganography (Fang et al., 2017; Yang et al., 2018, 2019, 2020; Kang et al., 2020; Yang et al., 2021; Li et al., 2021; Zhou et al., 2021) make no mention of or obscure the vocabulary selection step. At any rate, a finite word-level vocabulary should be seen as a security vulnerability. The complete absence of rare words can be exploited by steganalysis.

2.2 Subwords in Linguistic Steganography

In their experiments, Dai and Cai (2019), Ziegler et al. (2019), and Shen et al. (2020) built their steganographic models on top of GPT-2 (Radford et al., 2019), which used subwords. Dai and Cai (2019) and Shen et al. (2020) make explicit claims about the applicability of their methods to subword-level LMs. As we discussed in Section 1, however, they do not guarantee 100% recovery of the original subword tokens at Bob's side if Alice detokenizes subwords in order not to arouse Eve's suspicion.

Ueoka et al. (2021) point out that segmentation ambiguity may lead to decoding failures in linguistic steganography. Their solution is to simply skip subwords. This is possible because they edit human-generated texts by masking a small portion of tokens (Devlin et al., 2019), meaning that the resultant texts still contain rare words as before. If a similar technique is applied to a generation-based method, it falls back into the same problem as LMs with finite word-level vocabularies: the complete absence of rare words. Note that Ueoka et al. (2021) do not overcome segmentation ambiguity stemming from *scriptio continua* as we do for generation-based steganography in this paper.

Unfortunately, publications that postdate Ueoka et al. (2021) remain silent on segmentation ambiguity. Yang et al. (2022) do not detokenize cover texts at all. Yi et al. (2022), Zheng and Wu (2022), and Cao et al. (2022) make no single mention of subwords even though they used subword-baed models in their experiments. A faithful implementation of their methods would lead to decoding failures if detokenization is applied. For example, Yi et al. (2022) generate a cover text by interleaving a text-based secret message with dummy words. While

Bob is supposed to be informed of word positions of a secret message in the cover text, subwords do distort word-level positions.

We urge the community to take detokenization and retokenization as necessary steps for linguistic steganography. Clarification on the use of subwords is also needed.

3 Segmentation Ambiguity

The basic idea underlying generation-based neural linguistic steganography is to let a powerful neural LM, like GPT-2, enumerate natural variations of text into which bit chunks are encoded (Figure 1 (top)). We assume that Alice and Bob share the LM and an encoding strategy in advance. Following Ziegler et al. (2019), we also assume that Alice uses some introductory context (prompt) in a way such that Bob can use the same prompt during decoding. This helps diversify cover texts.

Now we consider an *ambiguity-unaware* method of generation-based steganography. For simplicity, we use block encoding (Fang et al., 2017) as the encoding strategy. At Alice's side, the LM is given a prompt and proposes probable next tokens at each time step. Alice sorts tokens in descending order of probability and performs a two-step filtering to select the top 2^n tokens. She first selects c tokens with probabilities greater than or equal to p and then chooses n such that it is the largest integer that satisfies $2^n \leq c$. Each of the tokens is given a unique bit chunk of length n, and Alice chooses the one that corresponds to the next n bits of the secret message. Alice repeats this until she finishes encoding the message. In the end, she detokenizes the text and sends it to Bob via a public channel.

Receiving the cover text, Bob first tokenizes it and then feeds the resultant tokens to the LM. He associates tokens with bit chunks in the same way as Alice. He decodes the secret message by repeatedly selecting a bit chunk corresponding to the next input token.

Unfortunately, this method is flawed because detokenization triggers segmentation ambiguity. Even if Alice generates the tokens "*un ##us ##able*", Bob obtains "*un ##usable*", which results in a wrong secret message. One might be tempted to use an error correcting code for the secret message, but it is of little help because one segmentation error affects all subsequent tokens.

4 Proposed Method

Figure 1 (bottom) shows an overview of the proposed method. To overcome the segmentation ambiguity problem in generation-based neural linguistic steganography, we combine two simple tricks: *stepwise tokenization* and *token disambiguation*.

Stepwise tokenization The first trick is to resist the temptation to use an off-the-shelf tokenizer at Bob's side. Bob is to imitate Alice's autoregressive generation process instead. At each time step, Bob selects a token that is a prefix of the remaining part of the detokenized cover text. For example, suppose that Bob receives the cover text "*unusable*". He first selects "*un*", which is a prefix of "*unusable*". Given the remaining part of the cover text, "*##usable*", he next selects a prefix of it. He repeats this until he finishes reading the cover text.

Token disambiguation Stepwise tokenization alone does not resolve segmentation ambiguity. At the second step of the aforementioned example, Bob faces an indeterminacy problem, as both "*##us*" and "*##usable*" are prefixes of "*##usable*". We resolve ambiguity by introducing a simple trick at the filtering step of both sides: If there are two candidate tokens w_1 and w_2 such that w_1 is a prefix of w_2, w_1 is removed from the candidate list. For the example above, Alice drops "*##us*" because it is a prefix of another candidate "*##usable*". Bob follows the same procedure as Alice to ensure that he can uniquely and correctly identify tokens.

5 Experiments

We compared the proposed method with the above-mentioned ambiguity-unaware method. For each method, we generated 10,000 cover texts following different prompts. Our primary focus was on decoding error rates, or the percentages of decoding failures among the 10,000 trials. A trial was deemed a failure if Bob re-tokenized the cover text differently from Alice. The proposed method is guaranteed to have a 0% decoding error rate, and we intended to experimentally confirm this. We also evaluated these methods in terms of payload capacity and security.

5.1 Datasets and Models

Datasets We chose three languages, Japanese, Russian, and English, for which GPT-2 models were available. For each language, 10,000 lines

Method	Japanese Error Rate (%)↓	Japanese Bits/Token ↑	Russian Error Rate (%)↓	Russian Bits/Token ↑	English Error Rate (%)↓	English Bits/Token ↑
Ambiguity-unaware	6.25	**2.47**	3.89	**2.52**	1.18	**2.70**
Proposed	**0.00**	2.28	**0.00**	2.41	**0.00**	2.59

Table 1: Decoding error rates and payload capacity (bits/token) in three different languages.

Japanese	
Alice	... ǀ を ǀ 成功 ǀ させる ǀ ...
Bob	... ǀ を成功させ ǀ る ǀ ...
Russian	
Alice	... ǀ переи ǀ ##да ǀ ##валось ǀ ...
Bob	... ǀ переи ǀ ##дав ǀ ##алось ǀ ...
English	
Alice	... ǀ med ǀ ##iation ǀ ...
Bob	... ǀ mediation ǀ ...

Table 2: Examples of cover texts for which the ambiguity-unaware method caused decoding failures. A vertical bar marks a token boundary.

of text were extracted from the CC-100 web corpus (Wenzek et al., 2020) and used as prompts of the LM. The length of a prompt was 30 characters for Japanese and 10 words for Russian and English. We used 64 random bits as a secret message.

Models We used medium-sized GPT-2 models taken from Hugging Face's `transformers` package[1] (Wolf et al., 2020). While the Japanese model used SentencePiece (Kudo and Richardson, 2018) for its vocabulary, the Russian and English models used a byte-level version of BPE (Radford et al., 2019). Accordingly, the prefixes in the proposed method were determined at the byte level. The probability threshold, p, was set to 0.01.

5.2 Automatic Detection (Steganalysis)

To measure the security of each method, we trained a discriminator to distinguish real texts from generated texts and evaluated the detection accuracy (the lower, the better). Specifically, we fine-tuned a BERT model on the binary classification task. As a simple baseline, we also evaluated texts randomly generated by GPT-2, without encoding any secret message. See Appendix A for details.

[1]Publicly available at `https://huggingface.co/` (Japanese: rinna/japanese-gpt2-medium, Russian: sberbank-ai/rugpt3medium_based_on_gpt2, and English: gpt2-medium). Each model had about 350M parameters.

Method	Accuracy (%)↓ ja	ru	en
Ambiguity-unaware	**86.6**	**85.4**	**88.2**
Proposed	88.6	86.5	91.5
(GPT-2 Random)	79.0	77.8	82.8

Table 3: Results of automatic detection. The last row shows a baseline that did not encode any secret message.

5.3 Results

Table 1 compares the two methods in terms of decoding error rate and payload capacity. The error rates for the ambiguity-unaware method were small but non-negligible. Note that in real situations, secret messages can be longer than 64 bits and consequently can push the decoding error rate upward. While not strictly comparable because of differences in hyperparameters and datasets, the three languages exhibit an interesting inclination: Japanese, the language without explicit word boundary markers, was the most susceptible to segmentation ambiguity, which was followed firstly by morphologically rich Russian and lastly by analytic English. Some examples of segmentation ambiguity of the ambiguity-unaware method are shown in Table 2 (see Appendix B for more examples).

The proposed method featured 100% correct decoding. It was at the expense of payload capacity, but no language showed more than a 10% drop.

Table 3 shows the result of automatic detection. The proposed method was slightly more prone to automatic detection than the ambiguity-unaware method. We suspect that the token disambiguation trick worsened the statistical deviation from human-written texts. The drop in performance is, however, not a prime cause of concern given that even the GPT-2 random baseline was easily detected. Switching to a more powerful LM would mitigate the risk. Finally, Appendix C shows some examples of generated texts.

6 Discussion

Although recent studies on generation-based neural linguistic steganography (Dai and Cai, 2019; Ziegler et al., 2019; Shen et al., 2020; Zhang et al., 2021) exploit the entire vocabulary distributions proposed by an LM, we turn back to naïve block encoding (Fang et al., 2017), which only uses the most probable 2^n tokens. In fact, our solution in its current form is not compatible with the use of the entire vocabulary because with $p = 0$, the token disambiguation trick always drops a fixed portion of the vocabulary. The present study should be seen as a proof-of-concept demonstration focusing on segmentation ambiguity. We hope that it sets out a future research direction.

7 Conclusions

Linguistic steganography is an interdisciplinary research area that combines information security and natural language processing (NLP). In this paper, we investigated its unexpected connection to the decades-old NLP task of word segmentation. Specifically, we shed light on segmentation ambiguity in generation-based neural linguistic steganography. Previously proposed methods are flawed if combined with a subword-level LM.

We proposed a combination of simple tricks to guarantee the recovery of the original tokens and thus the correct decoding of a secret message. Our solution is language-agnostic and is applicable even if no word boundaries are marked.

With powerful neural LMs, linguistic steganography is approaching the level of practical utility. Now is the time to face up to the fact that without detokenization, linguistic steganography is useless.

Ethical Considerations

Linguistic steganography conceals a secret message into a text, without a sign that secret communication is taking place. With the advance in neural language models, it is becoming possible to generate more natural texts while encoding a good amount of secret data. The proposed method is language-agnostic and guarantees the correct decoding of a secret message, thus making a step toward real-life applications. Intended applications of steganography are embedding copyright information, countering censorship, and just for fun, among others. However, it can also be used to transfer malicious contents, which makes steganography a dual-use technology. Therefore, along with steganography, steganalysis, the study of detecting the presence of hidden messages, would also be an encouraging research direction to safeguard against malicious use.

References

Ross J Anderson and Fabien AP Petitcolas. 1998. On the limits of steganography. *IEEE Journal on Selected Areas in Communications*, 16(4):474–481.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137—1155.

Krista Bennett. 2004. Linguistic steganography: Survey, analysis, and robustness concerns for hiding information in text. Technical report, Center for Education and Research in Information Assurance and Security, Purdue University.

Yi Cao, Zhili Zhou, Chinmay Chakraborty, Meimin Wang, Q. M. Jonathan Wu, Xingming Sun, and Keping Yu. 2022. Generative steganography based on long readable text generation. *IEEE Transactions on Computational Social Systems*, pages 1–11.

Ching-Yun Chang and Stephen Clark. 2014. Practical linguistic steganography using contextual synonym substitution and a novel vertex coding method. *Computational Linguistics*, 40(2):403–448.

Falcon Dai and Zheng Cai. 2019. Towards near-imperceptible steganographic text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4303–4308, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Tina Fang, Martin Jaggi, and Katerina Argyraki. 2017. Generating steganographic text with LSTMs. In *Proceedings of ACL 2017, Student Research Workshop*, pages 100–106, Vancouver, Canada. Association for Computational Linguistics.

Jessica Fridrich. 2009. *Steganography in digital media: principles, algorithms, and applications*. Cambridge University Press.

Huixian Kang, Hanzhou Wu, and Xinpeng Zhang. 2020. Generative text steganography based on LSTM network and attention mechanism with keywords. *Electronic Imaging*, 2020(4):291–1–291–8.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Yamin Li, Jun Zhang, Zhongliang Yang, and Ru Zhang. 2021. Topic-aware neural linguistic steganography based on knowledge graphs. *ACM/IMS Transactions on Data Science*, 2(2).

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jiaming Shen, Heng Ji, and Jiawei Han. 2020. Near-imperceptible neural linguistic steganography via self-adjusting arithmetic coding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 303–313, Online. Association for Computational Linguistics.

Gustavus J Simmons. 1984. The prisoners' problem and the subliminal channel. In *Advances in Cryptology*, pages 51–67. Springer.

Honai Ueoka, Yugo Murawaki, and Sadao Kurohashi. 2021. Frustratingly easy edit-based linguistic steganography with a masked language model. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5486–5492, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Tianyu Yang, Hanzhou Wu, Biao Yi, Guorui Feng, and Xinpeng Zhang. 2022. Semantic-preserving linguistic steganography by pivot translation and semantic-aware bins coding.

Zhong-Liang Yang, Xiao-Qing Guo, Zi-Ming Chen, Yong-Feng Huang, and Yu-Jin Zhang. 2019. RNN-Stega: Linguistic steganography based on recurrent neural networks. *IEEE Transactions on Information Forensics and Security*, 14(5):1280–1295.

Zhong-Liang Yang, Si-Yu Zhang, Yu-Ting Hu, Zhi-Wen Hu, and Yong-Feng Huang. 2021. VAE-Stega: Linguistic steganography based on variational auto-encoder. *IEEE Transactions on Information Forensics and Security*, 16:880–895.

Zhongliang Yang, Nan Wei, Qinghe Liu, Yongfeng Huang, and Yujin Zhang. 2020. GAN-TStega: Text steganography based on generative adversarial networks. In Hongxia Wang, Xianfeng Zhao, Yunqing Shi, Hyoung Joong Kim, and Alessandro Piva, editors, *Digital Forensics and Watermarking*, pages 18–31. Springer International Publishing.

Zhongliang Yang, Pengyu Zhang, Minyu Jiang, Yongfeng Huang, and Yu-Jin Zhang. 2018. RITS: Real-time interactive text steganography based on automatic dialogue model. In *Cloud Computing and Security*, pages 253–264, Cham. Springer International Publishing.

Biao Yi, Hanzhou Wu, Guorui Feng, and Xinpeng Zhang. 2022. ALiSa: Acrostic linguistic steganography based on BERT and Gibbs sampling. *IEEE Signal Processing Letters*, 29:687–691.

Siyu Zhang, Zhongliang Yang, Jinshuai Yang, and Yongfeng Huang. 2021. Provably secure generative linguistic steganography. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3046–3055, Online. Association for Computational Linguistics.

Xiaoyan Zheng and Hanzhou Wu. 2022. Autoregressive linguistic steganography based on BERT and consistency coding. *Security and Communication Networks*, 2022.

Xuejing Zhou, Wanli Peng, Boya Yang, Juan Wen, Yiming Xue, and Ping Zhong. 2021. Linguistic steganography based on adaptive probability distribution. *IEEE Transactions on Dependable and Secure Computing*. (early access article).

Zachary Ziegler, Yuntian Deng, and Alexander Rush. 2019. Neural linguistic steganography. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1210–1215, Hong

Kong, China. Association for Computational Linguistics.

A Details of Automatic Detection

The 10,000 texts generated by each method were split in an 8:1:1 ratio to create the training, development, and test sets. For the GPT-2 random baseline, we fed the same prompts to GPT-2 and performed random sampling according to the probabilities of the next tokens. The real texts and the texts generated by the GPT-2 random baseline were truncated so that they had comparable lengths with texts generated by steganographic methods. As a discriminator for each language, we used a `base`-sized BERT model taken from Hugging Face's `transformers` package (Japanese: cl-tohoku/bert-base-japanese-whole-word-masking, Russian: DeepPavlov/rubert-base-cased, and English: bert-base-cased). The numbers of parameters of the Japanese, Russian, and English BERT models were about 111M, 178M, and 108M, respectively.

To fine-tune a BERT model, we used generated texts following the prompts as inputs. Adam (Kingma and Ba, 2015) was used as the optimizer with a learning rate of 10^{-5}. The batch size was set to 32. We did not conduct any hyperparameter search and we report the experimental results of single runs. We trained each model for 10 epochs and used the checkpoint with the best validation accuracy as the final model. Throughout training, we used a single Quadro P6000 GPU. It took about 15 minutes to train a model.

B Examples of segmentation ambiguity

Table A.1 shows more examples of cover texts for which the ambiguity-unaware method caused decoding failures.

C Examples of texts generated by our proposed method

Table A.2 shows examples of texts generated by the proposed method following prompts. Each generated text carries a 64-bit random secret message.

Japanese	
Alice	... | 新しい | カ | ラム | を | 作成した | ら | どう | します | か | ? | ...
Bob	... | 新しい | カ | ラム | を作成し | たら | どう | します | か | ? | ...
Alice	... | 各 | 会場 | で | 撮影した | 写真を | 1 | 枚の | アルバム | にして | 配布 | される | ...
Bob	... | 各 | 会場 | で | 撮影した | 写真を | 1 | 枚のアルバム | にして | 配布 | される | ...
Russian	
Alice	... | Он | достаточно | лак | ##о | ##нич | ##ен | и | в | тоже | время | очень | ...
Bob	... | Он | достаточно | лак | ##они | ##чен | и | в | тоже | время | очень | ...
Alice	... | при | любом | исходе | дела | он | стремится | получить | максим | ##альную | ...
Bob	... | при | любом | исходе | дела | он | стремится | получить | максимальную | ...
English	
Alice	... | decreased | more | in | dry | land | environments | compared | to | wet | ##lands | ...
Bob	... | decreased | more | in | dry | land | environments | compared | to | wetlands | ...
Alice	... | while | C | ##CA | ##A | works | directly | with | local | schools | ...
Bob	... | while | CC | ##AA | works | directly | with | local | schools | ...

Table A.1: More examples of cover texts for which the ambiguity-unaware method caused decoding failures. A vertical bar marks a token boundary, and tokens colored red indicate where tokenization differs.

Japanese	
Prompt Generated text	お店ではたくさんの商品が所狭しと並んでいます。それだけ置いて おいて欲しい商品だからでしょう。 そんな風に商品が置いてあります ので、商品を購入するにはお目当てのお店に行くことになるのが通常
Prompt Generated text	私の場合は、仕事の帰宅時間がとても不規則だったので睡眠時間が 不規則な状態でしたが、今となってはその理由も分かる気もします(;'∀')」 これは私も同じ経験なのですが、「おやすみ
Russian	
Prompt Generated text	Обоих не было в игре, но ответят, как зайдут. Других тоже. Но, судя по всему, у вас с собой будет оружие и запасные патроны к ним. Это я уже от себя
Prompt Generated text	Это меня еще больше встревожило. Несколько часов я провел без сна. Но ничего. Я еще буду в порядке, когда у нас появятся свои дома... Но тут зазвонил мобильный
English	
Prompt Generated text	She hugged me then, burying her face into my chest. It hurt me too much and I was getting hot and sweaty, and I had a terrible stomach bug. It didn
Prompt Generated text	I have read many articles on the subject and have tried not to comment on this as it has become the focus of an intense debate amongst fans in my time with this

Table A.2: Examples of texts generated by the proposed method following prompts. Each generated text carries a 64-bit random secret message. Following Ziegler et al. (2019), we stop generation when the proposed method finishes embedding the message.