Neural Linguistic Steganography

Zachary M. Ziegler* **Yuntian Deng*** **Alexander M. Rush**
School of Engineering and Applied Sciences
Harvard University, Cambridge, MA
{zziegler@g, dengyuntian@g, srush@seas}.harvard.edu

Abstract

Whereas traditional cryptography encrypts a secret message into an unintelligible form, steganography conceals that communication is taking place by encoding a secret message into a cover signal. Language is a particularly pragmatic cover signal due to its benign occurrence and independence from any one medium. Traditionally, linguistic steganography systems encode secret messages in existing text via synonym substitution or word order rearrangements. Advances in neural language models enable previously impractical generation-based techniques. We propose a steganography technique based on arithmetic coding with large-scale neural language models. We find that our approach can generate realistic looking cover sentences as evaluated by humans, while at the same time preserving security by matching the cover message distribution with the language model distribution.

1 Introduction

Cryptography is central to modern communication, but while it effectively conceals the content of a message it reveals that meaningful communication is taking place. Steganography answers an alternative question: how to conceal a message in some cover signal (an image, text etc.) such that an eavesdropper is not even aware any meaningful communication has taken place (Amin et al., 2003; Shirali-Shahreza and Shirali-Shahreza, 2007; Kumar and Pooja, 2010)? Different from cryptography, in steganography security is derived from the inability to detect that a message exists within the cover signal, rather than the inability of an eavesdropper to determine the content of the message (Westfeld and Pfitzmann, 1999).

Natural language is an especially useful cover signal for steganography because it is prevalent and innocuous in everyday life. Furthermore, unlike images or audio in which encoded information depends on exact pixel values, linguistic steganography (hiding information in choices of words) is untethered from any one medium. For example, the sender could encode a message on a computer, read off the cover text in person, and then the receiver could separately enter the cover text into her decoder and retrieve the message.

Linguistic steganography methods can be classified as either edit-based or generation-based (Bennett, 2004). In edit-based methods, such as synonym substitution, the content of the cover text is selected by a human and slightly modified to encode information (Xiang et al., 2017; Hefei, 2009; Xiang et al., 2018). In generative methods an entire block of text is generated while encoding the message reversibly in the choice of tokens. Traditionally, most practical stenography systems are edit-based. These methods only encode a small amount of information (for example, 2 bits per tweet (Wilson and Ker, 2016)), whereas generation-based systems can encode an order of magnitude more information (Fang et al., 2017).

Generation-based steganography has a well-established foundation in information theory (Cox et al., 2005) with the aim of provably fooling any machine steganalysis. Due to weak language models practical performance at the scale of modern neural network-based models has not been considered or developed. One recent exception is a neural network-based approach using heuristic-based methods with the goal of fooling human eavesdroppers. These methods are theoretically sub-optimal, however, and while they demonstrate some fluency in generations the underlying language models are much worse than current state-of-the-art large models (Fang et al., 2017).

In this paper we aim to get the best of both approaches with a linguistic steganography

* Equal contribution.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 1210–1215,
Hong Kong, China, November 3–7, 2019.

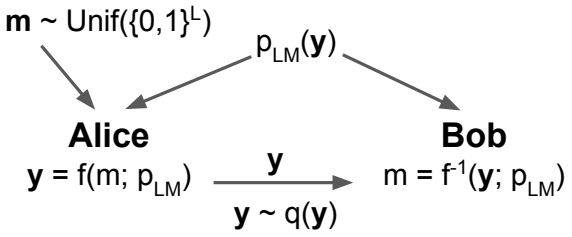

Figure 1: Problem setup. $\boldsymbol{m} \sim \text{Unif}(\{0,1\}^L)$ is the secret message, $\boldsymbol{y}$ is the cover text, $p(\boldsymbol{y})$ is the language model and f is a deterministic invertible function. f and the distribution of m implicitly defines q.

method based on arithmetic coding and modern language models. Our contribution is two-fold: 1) we show that a linguistic steganography approach combining arithmetic coding with state-of-the-art language models can achieve near-optimal statistical security; 2) human evaluations show that the cover text from our approach is able to fool humans even in the presence of context. Our code is available at https://github.com/harvardnlp/NeuralSteganography. A demo is available at http://steganography.live/

2 Background & Related work

Linguistic steganography We briefly give an overview of steganography as outlined in (Cox et al., 2005) and diagrammed in Figure 1: Alice wants to communicate a hidden message $\boldsymbol{m} \sim \text{Unif}(\{0,1\}^L)$ with Bob by encoding it into a choice of natural language cover text $\boldsymbol{y}$. The uniform distribution is chosen for $\boldsymbol{m}$ without loss of generality: if $\boldsymbol{m}$ has additional structure it can be further compressed to a uniformly distributed random variable (Han, 2005). Alice and Bob have both agreed on an invertible mapping f which performs the steganography. Alice and Bob also both have access to the exact same language model, $p_{LM}(\boldsymbol{y})$, which f can use during encoding and decoding. The steganography mapping f and the language model $p_{LM}(\boldsymbol{y})$ form the key.

The combination of the distribution of $\boldsymbol{m}$ and deterministic function $\boldsymbol{y} = f(\boldsymbol{m})$ implicitly defines a distribution for $\boldsymbol{y}$ which we denote q. This is the cover distribution of natural language that an eavesdropper would observe. As described in (Cox et al., 2005), the security of the system is determined by $D_{KL}(q||P_{true})$ where P_{true} is the true distribution of natural language. For example, if $D_{KL} = 0$ then the system is perfectly secure

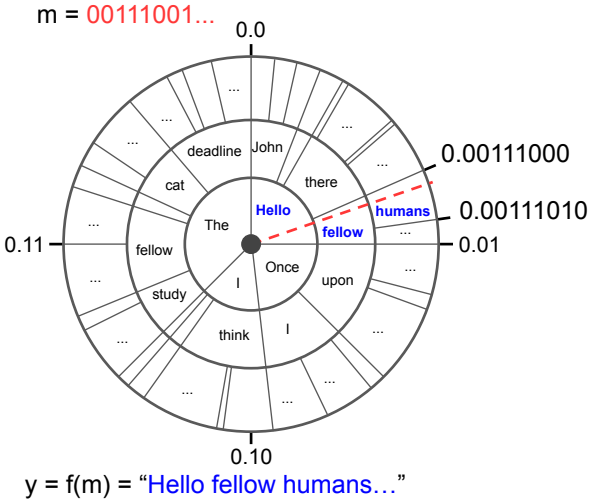

Figure 2: Diagram of arithmetic coding for steganography. See Section 3 for details.

because an eavesdropper could not distinguish the encoded messages from normal language. From an information theoretic perspective the security objective of steganography is therefore to ensure a small D_{KL}. At the same time, because f is invertible the average number of bits that can be encoded is given by the entropy $H(q)$. Thus, the compression objective is to maximize $H(q)$.

Generative coding techniques Recent work studying generative approaches for linguistic steganography have considered heuristic approaches to map uniform messages into similar distributions as natural language. Fang et al. (2017) considers a block-based approach (**Block**), where the vocabulary is randomly split into $2^{|B|}$ bins and each bin is indexed by a $|B|$-bit string. Encoding is performed by splitting the message up into chunks B_t and taking the maximum likelihood token as determined by the language model $p(y_t|y_{<t})$ that falls in bin B_t. Yang et al. (2019) proposes a related method based on constructing a Huffman coding (**Huffman**) at each step t based on $p(y_t|y_{<t})$ and encoding the message in variable length chunks via the Huffman tree. Both objectives provide a "quality" parameter that trades off quality at the expense of encoding fewer bits.

These approaches target a different security objective than the information-theoretic view in Cox et al. (2005). Instead of aiming to minimize the KL at maximal compression they aim to maximize generated quality at maximal compression (Yang et al., 2019; Fang et al., 2017). The issue with these approaches is that they aim to fool human eavesdroppers for whom generation quality

Context	kim jong il was the enigmatic leader of the most enigmatic country on earth. much about kim's life was, and is, shrouded in mystery. even the year of his birth – 1942 – has been contested by knowledgeable sources in south korea and russia, who claim kim was born a few years earlier but that his official birth date was pushed back to make him exactly 30 years younger than his father, north korea's founding leader, kim il sung.
Message 1	Bob, I have something very important to tell you. Eve is not who she says she is. I know that you think she's your friend, but really she's a creepy stalker who wants to read all of your messages for some unknown reason.
Cover text 1	That kim's birthdate was not even known to the people of south korea is even more baffling. however, there is another contradiction. bak tae-jung, the ruler of north korea at the time, claimed to have inherited kim's kim ju-ha-teh, his so-called golden heritage, from his father, the last kim Yun-jong. and he also
Message 2	The secret code is X3SJ83F. Once you open the door you will see a staircase on your left and a hallway to your right. Go down the hallway and enter the code in the safe at the end of the hallway.
Cover text 2	kim il sang, by the way, was not the first leader of north korea, but the first for nearly a thousand years. his leader was kim jong il, the same person the people voted in to power to start with, without their knowledge or consent. though it's been claimed by others that kim il sang fled north korea in 1940 to escape the reprisals, even that is disputed by k

Table 1: Steganography example. Two different encoded messages are produced given the same introductory context. The messages are first converted into bit strings and then mapped to cover text using the arithmetic steganography approach described in Section 4.4.

is most important; but are susceptible to machine-based eavesdroppers which can in principle identify statistical patterns in generations.

Concurrent with this work, Dai and Cai (2019) conduct a related analysis in terms of KL with a modified Huffman algorithm. Experiments consider the distribution of KL values, although no human evaluation is performed.

3 Arithmetic coding

Arithmetic coding is a data compression method designed specifically to code strings of elements with a known probability distribution (Rissanen and Langdon, 1979; Zoph et al., 2015). For long strings the coding is optimal; it compresses information to its entropy (Rissanen and Langdon, 1979). In practice, it is often more efficient than Huffman coding because it does not require blocking. Arithmetic coding traditionally maps a string of elements to a uniformly distributed binary string. To use such a coding for steganography we reverse the order: first a (uniformly sampled) message is selected, then the message is mapped to a sequence (words).

The coding scheme is demonstrated in Figure 2. In this work the probability distribution comes from the conditional distributions of a pretrained language model, but for illustration purposes a hand-crafted example distribution is used in the diagram. Concentric circles represent timesteps; the innermost represents $t = 1$, the middle $t = 2$, and the outer $t = 3$. Each circle represents the conditional distribution $p(y_t|y_{<t})$. For example, given that y_1 = "Once", $p(y_2|y_1)$ has "upon" and "I" as the only possible tokens with equal probability. The circle diagram spans [0,1) from 0 at the top, clockwise around to 1.

To encode the message into text, the secret message m is viewed as a binary representation of a fraction in the range $[0, 1)$. This fraction uniquely marks a point on the edge of the circle, as well as a line from the origin to the point. Encoding is performed by simply reading off the tokens corresponding to the bins. Encoding stops when the list of tokens unambiguously defines the message.

Decoding is performed via the reverse operation: the sequence of natural language tokens progressively narrows the range of possible messages. Assuming that the original message is encoded with a predetermined end token, decoding terminates once both sides of the range of possible messages includes the end token.

Sallee (2004) show that just as arithmetic coding is optimal for data compression, it is also optimal for steganography. Specifically, given any goal distribution $p_s(\boldsymbol{y})$ that one wants to sample from, starting with a uniform $\boldsymbol{m}$ and applying the deterministic steganography procedures yields a distribution $q = p_s$ or equivalently $D_{KL}(q||p_s) = 0$ for long sequences. This further ensures that $H(q) = H(p_s)$ and therefore the number of bits encoded on average is equal to the entropy of p_s. Intuitively, this works because higher probability sequences map to larger "chunks" of the pie and therefore require fewer bits to uniquely determine.

In this work we apply arithmetic coding to lin-

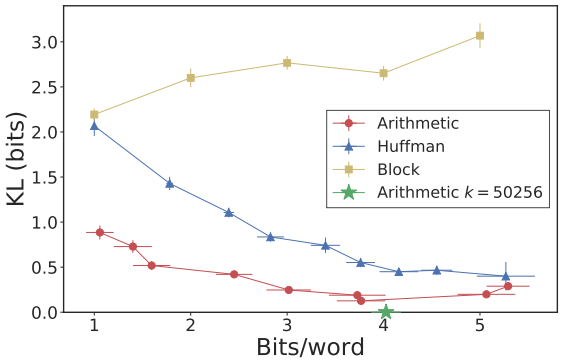

Figure 3: Information theoretic evaluation. KL and bits/word are evaluated for each algorithm at different values of the tuning parameter; each datapoint gives the mean and standard error for repeated samples with a fixed value of the tuning parameter. We show the standard error bars for both bits/word and KL.

guistic steganography. Given a pretrained language model, this approach can be directly applied to produce cover texts given some preliminary context. To ensure that these texts seem more natural to human eavesdroppers we introduce two parameters to trade off quality for compression: we modulate the LM distribution by a temperature τ, and we truncate the distribution to a top-k tokens at each position to ensure generations do not include tokens from the long tail.

In practice, a naive implementation of arithmetic coding quickly runs into precision limitations. We use a common equivalent variant based on fixed precision binary fractions (Rubin, 1979).

4 Experiments and results

4.1 Experimental Setup

We use the 345M parameter GPT-2 model as our language model p_{LM} (Radford et al., 2019). We run our experiments on the CNN/Dailymail (CNNDM) dataset (Hermann et al., 2015; Nallapati et al., 2016), which is used to provide context for generation[1]. We take the first three sentences of each news article as the context on which to condition our language model, and use the indicated steganography algorithm to generate an entire sentence for estimation or evaluation given a uniform random message. We compare the proposed arithmetic coding-based algorithm with the Block (Fang et al., 2017) and Huffman (Yang et al., 2019) methods as baselines. The baselines are re-implemented to allow for a fair comparison using the strong GPT-2 language model. Each method has a parameter that controls the various tradeoffs: for arithmetic encoding we vary the temperature from 0.4 to 1.2 with $k = 300$, for Huffman encoding we vary the truncation length from 2^1 to 2^8, for block encoding we vary the block size from 2^1 to 2^5. For the KL analysis we also evaluate arithmetic coding with $\tau = 1$ and $k = 50256$, i.e. the unmodulated language model.

[1]Conditional generation makes it more challenging to fool humans and is thus better for comparisons. CNNDM presents an additional challenge as poor generations may have clear factual errors.

4.2 Metrics

We consider several different metrics to assess performance. In all cases, it is important that the algorithms encode information efficiently. The metric **Bits/word** is the ratio of message bits encoded to sentence length.

Another key aspect of the system is being able to fool a statistical adversary in a *distributional* sense, i.e. when the adversary gets multiple samples from q. This can be quantified by the **KL** divergence between q and P_{true}. Since we do not have access to P_{true} we resort to $D_{KL}(q||p_{LM})$. p_{LM} might not match the distribution of natural language, but it allows for relative comparisons of steganography algorithms. The remaining gap is a function of language model performance, which is not the focus of this work. For each model the KL is estimated via MC sampling on the news data.

Finally it is important to be able to convince a **human adversary** that the text is natural. This differs from statistical considerations because a human will only see a single sample from q at a time. To this end, we conduct a human study aligned with the information theoretic evaluation above, using Amazon Mechanical Turk. For each of the three methods we generate sentences with randomly sampled messages conditioned on a news article context, and ask participants to indicate if the sentence is a likely next sentence in context. We repeat this experiment with the true next sentence as well. See the Supplementary Materials for more details.

4.3 Quantitative results

Figure 3 shows the information theoretic evaluation. For any bits/word in the range of (1, 5), arithmetic coding achieves the lowest KL, i.e. the distribution of the encoded texts is the most similar to the language model distribution, providing the best defensive against algorithmic detections. Most striking, arithmetic coding with the unmodu-

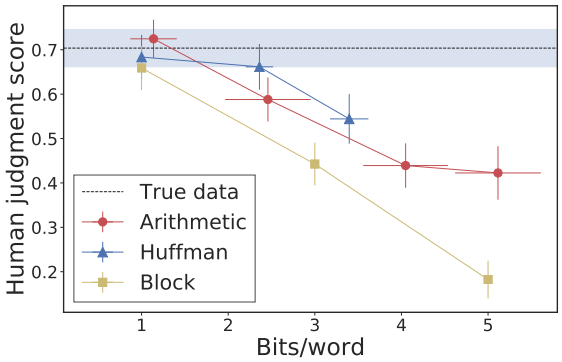

Figure 4: Human evaluation. Y axis shows the percentage of humans considering the generation to be a proper continuation of the context. Huffman coding achieves the best generation quality as judged by humans, while arithmetic coding is comparable (the difference is not statistically significant).

lated language model induces a q distribution with a KL of 4e-8 nats. This indicates that, consistent with theory (Sallee, 2004), arithmetic coding enables generative steganography matching the *exact* distribution of the language model used. The small but positive KL value for arithmetic coding with $k = 300$ comes from the slight distributional difference when the long tail is truncated.

Figure 4 shows the human evaluation results. Within 3 bits/word both Huffman and arithmetic coding give plausible next sentences over 50% of the time, and at 1 bit/word both methods lead to cover texts statistically indistinguishable from human-written sentences. Although Huffman coding reaches slightly better performance than arithmetic coding, the transferred cover text distribution is statistically different from natural text (see Figure 3), which can be potentially discovered by automatic systems. Compared to the KL analysis which focuses on the relative performance of the algorithms, the human evaluation highlights the realizable generation quality enabled specifically by large pretrained language models. Still, there is room for improvement in terms of language modeling capabilities: whereas the arithmetic approach reaches minimum KL around 4 bits/word with $\tau = 1.0$, only at 1 bit/word with $\tau = 0.4$ do the generations fool humans.

4.4 Qualitative results

Steganography fundamentally deals with hiding uniformly random message bits in cover text. In many cases, however, the message may itself be natural language. In these cases we have an additional choice of how the text is converted to bits. One option is simply the unicode encoding (i.e. UTF-8), though this is highly inefficient. Instead, we can use the same arithmetic coding algorithm in reverse, with an empty context, to compress the message text into near-uniformly distributed bits (Han, 2005). The full procedure is as follows:

1. Alice decodes message text into bits via arithmetic coding with an empty context.
2. Alice encodes bits into cover text via arithmetic coding with a predetermined context.
3. Alice sends cover text over public channel.
4. Bob decodes cover text into bits via arithmetic coding with the predetermined context.
5. Bob encodes bits into message text via arithmetic coding with empty context.

Because the same strong LM is used with different contexts for the two applications of arithmetic coding, the message text length and cover text length will be comparable (modulo a small overhead) but the content of the text can be completely different. This is demonstrated in Table 1, which shows two examples of natural language messages (Step 1) encoded in unrelated and innocuous cover text (Step 3). In both cases the message is encoded efficiently into cover text which is largely fluent and coherent, and follows from the given context.

5 Conclusion

We demonstrate how the combination of large pretrained language models and an arithmetic coding-based steganography algorithm allows for a steganography system that can fool both statistical adversaries who analyze the full generative distribution, and human adversaries who analyze a single sample. While our work shows the potential for high-quality efficient steganography and the realizable optimality of arithmetic coding, future advancements in language modeling can push steganographic performance even further. With current state-of-the-art language models the steganography algorithms studied generate most convincing cover texts at low compression, where the KL is moderate. As language models continue to improve, they can be directly plugged into our arithmetic approach to maximize steganographic performance.

Acknowledgments

This work was supported by NSF 1845664. YD was supported by a Baidu AI Fellowship.

References

Muhalim Mohamed Amin, Mazleena Salleh, Subariah Ibrahim, Mohd Rozi Katmin, and MZI Shamsuddin. 2003. Information hiding using steganography. In *4th National Conference of Telecommunication Technology, 2003. NCTT 2003 Proceedings.*, pages 21–25. IEEE.

Krista Bennett. 2004. Linguistic steganography: Survey, analysis, and robustness concerns for hiding information in text.

Ingemar J Cox, Ton Kalker, Georg Pakura, and Mathias Scheel. 2005. Information transmission and steganography. In *International Workshop on Digital Watermarking*, pages 15–29. Springer.

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.

Tina Fang, Martin Jaggi, and Katerina Argyraki. 2017. Generating steganographic text with lstms. *arXiv preprint arXiv:1705.10742*.

Te Sun Han. 2005. Folklore in source coding: Information-spectrum approach. *IEEE Trans. Inf. Theor.*, 51(2):747–753.

PR Hefei. 2009. Steganalysis of synonym-substitution based natural language watermarking. *International Journal of Multimedia and Ubiquitous Engineering*, 4(2).

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Arvind Kumar and Km Pooja. 2010. Steganography-a data hiding technique. *International Journal of Computer Applications*, 9(7):19–23.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

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

Jorma Rissanen and Glen G Langdon. 1979. Arithmetic coding. *IBM Journal of research and development*, 23(2):149–162.

Frank Rubin. 1979. Arithmetic stream coding using fixed precision registers. *IEEE Transactions on Information Theory*, 25(6):672–675.

P. Sallee. 2004. Model based steganography. *Int. Workshop on Digital Watermarking*.

Mohammad Shirali-Shahreza and M Hassan Shirali-Shahreza. 2007. Text steganography in sms. In *2007 International Conference on Convergence Information Technology (ICCIT 2007)*, pages 2260–2265. IEEE.

Andreas Westfeld and Andreas Pfitzmann. 1999. Attacks on steganographic systems. In *International workshop on information hiding*, pages 61–76. Springer.

Alex Wilson and Andrew D Ker. 2016. Avoiding detection on twitter: embedding strategies for linguistic steganography. *Electronic Imaging*, 2016(8):1–9.

Lingyun Xiang, Yan Li, Wei Hao, Peng Yang, and Xiaobo Shen. 2018. Reversible natural language watermarking using synonym substitution and arithmetic coding. *Comput., Mater. Continua*, 55(3):541–559.

Lingyun Xiang, Xinhui Wang, Chunfang Yang, and Peng Liu. 2017. A novel linguistic steganography based on synonym run-length encoding. *IEICE transactions on Information and Systems*, 100(2):313–322.

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.

Barret Zoph, Marjan Ghazvininejad, and Kevin Knight. 2015. How much information does a human translator add to the original? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 889–898.