Generating Steganographic Text with LSTMs

Tina Fang
University of Waterloo
tbfang@edu.uwaterloo.ca

Martin Jaggi
École polytechnique fédérale de Lausanne
martin.jaggi@epfl.ch

Katerina Argyraki
École polytechnique fédérale de Lausanne
katerina.argyraki@epfl.ch

Abstract

Motivated by concerns for user privacy, we design a steganographic system ("stegosystem") that enables two users to exchange encrypted messages without an adversary detecting that such an exchange is taking place. We propose a new linguistic stegosystem based on a Long Short-Term Memory (LSTM) neural network. We demonstrate our approach on the Twitter and Enron email datasets and show that it yields high-quality steganographic text while significantly improving capacity (encrypted bits per word) relative to the state-of-the-art.

1 Introduction

The business model behind modern communication systems (email services or messaging services provided by social networks) is incompatible with end-to-end message encryption. The providers of these services can afford to offer them free of charge because most of their users agree to receive "targeted ads" (ads that are especially chosen to appeal to each user, based on the needs the user has implied through their messages). This model works as long as users communicate mostly in the clear, which enables service providers to make informed guesses about user needs.

This situation does not prevent users from encrypting a few sensitive messages, but it does take away some of the benefits of confidentiality. For instance, imagine a scenario where two users want to exchange forbidden ideas or organize forbidden events under an authoritarian regime; in a world where most communication happens in the clear, encrypting a small fraction of messages automatically makes these messages—and the users who exchange them—suspicious.

With this motivation in mind, we want to design a system that enables two users to exchange encrypted messages, such that a passive adversary that reads the messages can determine neither the original content of the messages nor the fact that the messages are encrypted.

We build on linguistic steganography, i.e., the science of encoding a secret piece of information ("payload") into a piece of text that looks like natural language ("stegotext"). We propose a novel stegosystem, based on a neural network, and demonstrate that it combines high quality of output (i.e., the stegotext indeed looks like natural language) with the highest capacity (number of bits encrypted per word) published in literature.

In the rest of the paper, we describe existing linguistic stegosystems along with ours (§2), provide details on our system (§3), present preliminary experimental results on Twitter and email messages (§4), and conclude with future directions (§5).

2 Linguistic Steganography

In this section, we summarize related work (§2.1), then present out proposal (§2.2).

2.1 Related Work

Traditional linguistic stegosystems are based on modification of an existing cover text, e.g., using synonym substitution (Topkara et al., 2006; Chang and Clark, 2014) and/or paraphrase substitution (Chang and Clark, 2010). The idea is to encode the secret information in the transformation of the cover text, ideally without affecting its meaning or grammatical correctness. Of these systems, the most closely related to ours is CoverTweet (Wilson et al., 2014), a state-of-the-art cover modification stegosystem that uses Twitter as the medium of cover; we compare to it in our preliminary evaluation (§4).

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics- Student Research Workshop, pages 100–106
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-3017

Cover modification can introduce syntactic and semantic unnaturalness (Grosvald and Orgun, 2011); to address this, Grovsald and Orgun proposed an alternative stegosystem where a human generates the stegotext manually, thus improving linguistic naturalness at the cost of human effort (Grosvald and Orgun, 2011).

Matryoshka (Safaka et al., 2016) takes this further: in step 1, it generates candidate stegotext automatically based on an n-gram model of the English language; in step 2, it presents the candidate stegotext to the human user for polishing, i.e., ideally small edits that improve linguistic naturalness. However, the cost of human effort is still high, because the (automatically generated) candidate stegotext is far from natural language, and, as a result, the human user has to spend significant time and effort manually editing and augmenting it.

Volkhonskiy et al. have applied Generative Adversarial Networks (Goodfellow et al., 2014) to image steganography (Volkhonskiy et al., 2017), but we are not aware of any text stegosystem based on neural networks.

2.2 Our Proposal: Steganographic LSTM

Motivated by the fact that LSTMs (Hochreiter and Schmidhuber, 1997) constitute the state of the art in text generation (Jozefowicz et al., 2016), we propose to automatically generate the stegotext from an LSTM (as opposed to an n-gram model). The output of the LSTM can then be used either directly as the stegotext, or Matryoshka-style, i.e., as a candidate stegotext to be polished by a human user; in this paper, we explore only the former option, i.e., we do not do any manual polishing. We describe the main components of our system in the paragraphs below; for reference, Fig. 1 outlines the building blocks of a stegosystem (Salomon, 2003).

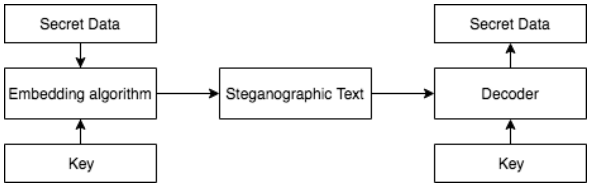

Figure 1: Stegosystem building blocks.

Secret data. The secret data is the information we want to hide. First, we compress and/or encrypt the secret data (e.g., in the simplest setting using the ASCII coding map) into a *secret-containing bit string* S. Second, we divide S into smaller *bit blocks* of length $|B|$, resulting in a total of $|S|/|B|$[1] bit blocks. For example, if $S = 100001$ and $|B| = 2$, our bit-block sequence is 10, 00, 01. Based on this bit-block sequence, our steganographic LSTM generates words.

Key. The sender and receiver share a key that maps bit blocks to token sets and is constructed as follows: We start from the *vocabulary*, which is the set of all possible tokens that may appear in the stegotext; the tokens are typically words, but may also be punctuation marks. We partition the vocabulary into $2^{|B|}$ *bins*, i.e., disjoint token sets, randomly selected from the vocabulary without replacement; each token appears in exactly one bin, and each bin contains $|V|/2^{|B|}$ tokens. We map each bit block B to a bin, denoted by W_B. This mapping constitutes the shared key.

Bit Block	Tokens
00	This, am, weather, ...
01	was, attaching, today, ...
10	I, better, an, Great, ...
11	great, than, NDA, ., ...

Table 1: Example shared key.

Embedding algorithm. The embedding algorithm uses a modified word-level LSTM for language modeling (Mikolov et al., 2010). To encode the secret-containing bit string S, we consider one bit block B at a time and have our LSTM select one token from bin W_B; hence, the candidate stegotext has as many tokens as the number of bit blocks in S. Even though we restrict the LSTM to select a token from a particular bin, each bin should offer sufficient variety of tokens, allowing the LSTM to generate text that looks natural. For example, given the bit string "1000011011" and the key in Table 1, the LSTM can form the partial sentence in Table 2. We describe our LSTM model in more detail in the next section.

Bit String	10	00	01	10	11
Token	I	am	attaching	an	NDA

Table 2: Example stegotext generation.

[1] If $|B| \nmid |S|$, then we leave the remainder bit string out of encryption.

Decoder. The decoder recovers the original data deterministically and in a straightforward manner: it takes as input the generated stegotext, considers one token at a time, finds the token's bin in the shared key, and recovers the original bit block.

Common-token variant. We also explore a variant where we add a set of *common tokens*, C, to all bins. These common tokens do not carry any secret information; they serve only to enhance stegotext naturalness. When the LSTM selects a common token from a bin, we have it select an extra token from the same bin, until it selects a non-common token. The decoder removes all common tokens before decoding. We discuss the choice of common tokens and its implication on our system's performance in Section 4.

3 Steganographic LSTM Model

In this section, we provide more details on our system: how we modify the LSTM (§3.1) and how we evaluate its output (§3.2).

3.1 LSTM Modification

Text generation in classic LSTM. Classic LSTMs generate words as follows (Sutskever et al., 2011): Given a word sequence $(x_1, x_2, \ldots, x_T)$, the model has hidden states $(h_1, \ldots, h_T)$, and resulting output vectors $(o_1, \ldots, o_T)$. Each output vector o_t has length $|V|$, and each output-vector element $o_t^{(j)}$ is the unnormalized probability of word j in the vocabulary. Normalized probabilities for each candidate word are obtained by the following softmax activation function:

$$softmax(o_t)_j := \exp(o_t^{(j)}) \Big/ \sum_k \exp(o_t^{(k)}).$$

The LSTM then selects the word with the highest probability $P[x_{t+1} \mid x_{\leq t}]$ as its next word.

Text generation in our LSTM. In our steganographic LSTM, word selection is restricted by the shared key. That is, given bit block B, the LSTM has to select its next word from bin W_B. We set $P[x = w_j] = 0$ for $j \notin W_B$, so that the multinomial softmax function selects the word with the highest probability within W_B.

Common tokens. In the common-token variant, we restrict $P[x = w_j] = 0$ only for $j \notin (W_B \cup C)$, where C is the set of common tokens added to all bins.

3.2 Evaluation Metrics

We use perplexity to quantify stegotext quality; and capacity (i.e., encrypted bits per output word) to quantify its efficiency in carrying secret information. In Section 4, we also discuss our stegotext quality as empirically perceived by us as human readers.

Perplexity. Perplexity is a standard metric for the quality of language models (Martin and Jurafsky, 2000), and it is defined as the average per-word log-probability on the valid data set: $\exp(-1/N \sum_i \ln p[w_i])$ (Jozefowicz et al., 2016). Lower perplexity indicates a better model.

In our steganographic LSTM, we cannot use this metric as is: since we enforce $p[w_i] = 0$ for $w_i \notin W_B$, the corresponding $\ln p[w_i]$ becomes undefined under this vocabulary.

Instead, we measure the probability of w_i by taking the average of $p[w_i]$ over all possible secret bit blocks B, under the assumption that bit blocks are distributed uniformly. By the Law of Large Numbers (Révész, 2014), if we perform many stegotext-generating trials using different random secret data as input, the probability of each word will tend to the expected value, $\sum p[w_i, B]/2^{|B|}$, Hence, we set $p[w_i] := \sum p[w_i, B]/2^{|B|}$ instead of $p[w_i] = 0$ for $w_i \notin W_B$.

Capacity. Our system's capacity is the number of encrypted bits per output word. Without common tokens, capacity is always $|B|$ bits/word (since each bit block of size $|B|$ is always mapped to one output word). In the common-token variant, capacity decreases because the output includes common tokens that do not carry any secret information; in particular, if the fraction of common tokens is p, then capacity is $(1 - p) \cdot |B|$.

4 Experiments

In this section, we present our preliminary experimental evaluation: our Twitter and email datasets (§4.1), details about the LSTMs used to produce our results (§4.2), and finally a discussion of our results (§4.3).

4.1 Datasets

Tweets and emails are among the most popular media of open communication and therefore provide very realistic environments for hiding information. We thus trained our LSTMs on those two domains, Twitter messages and Enron emails

(Klimt and Yang, 2004), which vary greatly in message length and vocabulary size.

For Twitter, we used the NLTK tokenizer to tokenize tweets (Bird, 2006) into words and punctuation marks. We normalized the content by replacing usernames and URLs with a username token (<user>) and a URL token (<url>), respectively. We used 600 thousand tweets with a total of 45 million words and a vocabulary of size 225 thousand.

For Enron, we cleaned and extracted email message bodies (Zhou et al., 2007) from the Enron dataset, and we tokenized the messages into words and punctuation marks. We took the first 100MB of the resulting messages, with 16.8 million tokens and a vocabulary size of 406 thousand.

4.2 Implementation Details

We implemented multi-layered LSTMs based on PyTorch[2] in both experiments. We did not use pre-trained word embeddings (Mikolov et al., 2013; Pennington et al., 2014), and instead trained word embeddings of dimension 200 from scratch.

We optimized with Stochastic Gradient Descent and used a batch size of 20. The initial learning rate was 20 and the decay factor per epoch was 4. The learning rate decay occurred only when the validation loss did not improve. Model training was done on an NVIDIA GeForce GTX TITAN X.

For Twitter, we used a 2-layer LSTM with 600 units, unrolled for 25 steps for back propagation. We clipped the norm of the gradients (Pascanu et al., 2013) at 0.25 and applied 20% dropout (Srivastava et al., 2014). We stopped the training after 12 epochs (10 hours) based on validation loss convergence.

For Enron, we used a 3-layer LSTM with 600 units and no regularization. We unrolled the network for 20 steps for back propagation. We stopped the training after 6 epochs (2 days).

4.3 Results and Discussion

4.3.1 Tweets

We evaluate resulting tweets generated by LSTMs of 1 (non-steganographic), 2, 4, 8 bins. Furthermore, we found empirically that adding 10 most frequent tokens from the Twitter corpus was enough to significantly improve the grammatical correctness and semantic reasonableness of tweets. Table 3 shows the relationship between capacity (*bits per word*), and quantitative text quality (*perplexity*). It also compares models with and without adding common tokens using perplexity and bits per word.

Table 4 shows example output texts of LSTMs with and without common tokens added. To reflect the variation in the quality of the tweets, we represent tweets that are good and poor in quality[3].

We replaced <user> generated by the LSTM with mock usernames for a more realistic presentation in Table 4. In practice, we can replace the <user> tokens systematically, randomly selecting followers or followees of that tweet sender, for example.

Re-tweet messages starting with "RT" can also be problematic, because it will be easy to check whether the original message of the retweeted message exists. A simple approach to deal with this is to eliminate "RT" messages from training (or at generation). Finally, since we made all tweets lower case in the pre-processing step, we can also post-process tweets to adhere to proper English capitalization rules.

	Original		**Common Tokens**	
# of Bins	bpw	ppl	bpw	ppl
1	0	134.73	0	134.73
2	1	190.84	0.65	171.35
4	2	381.2	1.17	277.55
8	3	833.11	1.53	476.66

Table 3: An increase of of capacity correlates with an increase of perplexity, which implies that there is a negative correlation between capacity and text quality. After adding common tokens, there is a significant reduction in perplexity *(ppl)*, at the expense of a lower capacity *(bits per word)*.

4.3.2 Emails

We also tested email generation, and Table 5 shows sample email passages[4] from each bin. We post-processed the emails with untokenization of punctuations.

The biggest difference between emails and tweets is that emails have a much longer range for

[2]https://github.com/pytorch

[3]For each category, we manually evaluate 60 randomly generated tweets based grammatical correctness, semantic coherence, and resemblance to real tweets. We select tweets from the 25th, 50th, and 75th percentile, and call them "good", "average", and "poor" respectively. We limit to tweets that are not offensive in language.

[4]We only present passages "average" in quality to conserve space.

# of Bins	Tweets	Tweets with Common Tokens
2	**good:** i was just looking for someone that i used have. **poor:** cry and speak! rt @user421: relatable personal hygiene for body and making bad things as a best friend in lifee	**good:** i'm happy with you. i'll take a pic **poor:** rt: cut your hair, the smallest things get to the body.
4	**good:** @user390 looool yeah she likes me then ;). you did? **poor:** "where else were u making?... i feel fine? - e? lol" * does a voice for me & take it to walmart?	**good:** i just wanna move. collapses. **poor:** i hate being contemplating for something i want to.
8	**good:** @user239 hahah. sorry that my bf is amazing because i'm a bad influence ;). **poor:** so happy this to have been working my ass and they already took the perfect. but it's just cause you're too busy the slows out! love... * dancing on her face, holding two count out cold * (a link with a roof on punishment... - please :)	**good:** i hate the smell of my house. **poor:** a few simple i can't. i need to make my specs jump surprisingly.

Table 4: We observe that the model with common tokens produces tweets simpler in style, and uses more words from the set of common tokens. There is a large improvement in grammatical correctness and context coherence after adding common tokens, especially in the "poor" examples. For example, adding the line break token reduced the length of the tweet generated from the 8-bin LSTM.

context dependency, with context spanning sentences and paragraphs. This is challenging to model even for the non-steganographic LSTM. Once the long-range context dependency of the non-steganographic LSTM improves, the context dependency of the steganographic LSTMs should also improve.

# of Bins	Sample Email
1	——Original Message—— From: Nelson, Michelle Sent: Thursday, January 03, 2002 3:35 PM To: Maggi, Mike Subject: Hey, You are probably a list of people that are around asleep about the point of them and our wife. Rob
2	If you do like to comment on the above you will not contact me at the above time by 8:00 a.m. on Monday, March 13 and July 16 and Tuesday, May 13 and Tuesday, March 9 - Thursday, June 17, - 9:00 to 11:30 AM.
4	At a moment when my group was working for a few weeks, we were able to get more flexibility through in order that we would not be willing.

Table 5: The issue of context inconsistency is present for all bins. However, the resulting text remains syntactical even as the number of bins increases.

4.4 Comparison with Other Stegosystems

For all comparisons, we use our 4-bin model with no common tokens added.

Our model significantly improves the state-of-the-art capacity. Cover modification based stegosystems hide 1-2 bits per sentence (Chang and Clark, 2012). The state-of-the-art Twitter stegosystem hides 2.8 bits of per tweet (Wilson and Ker, 2016). Assuming 16.04 words per tweet[5], our 4-bin system hides **32 bits per tweet**, over 11 times higher than (Wilson and Ker, 2016).

We hypothesize that the subjective quality of our generated tweets will be comparable to tweets produced by CoverTweet (2014). We present some examples[6] in Table 6 to show there is potential for a comparison. This contrasts the previous conception that cover generation methods are fatally weak against human judges (Wilson et al., 2014). CoverTweet was tested to be secure against human judges. Formal experiments will be necessary to establish that our system is also secure against human judges.

CoverTweet (2014)	Steganographic LSTM
yall must have 11:11 set 1 minute early before yall tweet it, because soon as 11:11 hit yall don't wastes no time. lol	i wanna go to sleep in the gym, ny in peoples houses & i'm in the gym..! :((
you can tell when somebody hating on you!	i would rather marry a regular sunday!!
most of the people who got mouth can't beat you.	my mom is going so hard to get his jam.

Table 6: The tweets generated by the 4-bin LSTM (32 bits per tweet) are reasonably comparable in quality to tweets produced by CoverTweet (2.8 bits per tweet).

Our system also offers flexibility for the user to freely trade-off capacity and text quality. Though we chose the 4-bin model with no common tokens for comparison, user can choose to use more bins

[5]Based a random sample of 2 million tweets.

[6]Tweets selected for comparison are "average" in quality.

to achieve an even higher capacity, or use less bins and add common tokens to increase text quality. This is not the case with existing cover modification systems, where capacity is bounded above by the number of transformation options (Wilson et al., 2014).

5 Conclusion and Future Work

In this paper, we opened a new application of LSTMs, namely, steganographic text generation. We presented our steganographic model based on existing language modeling LSTMs, and demonstrated that our model produces realistic tweets and emails while hiding information.

In comparison to the state-of-the-art steganographic systems, our system has the advantage of encoding much more information (around 2 bits per word). This advantage makes the system more usable and scalable in practice.

In future work, we will formally evaluate our system's security against human judges and other steganography detection (steganalysis) methods (Wilson et al., 2015; Kodovsky et al., 2012). When evaluated against an automated classifier, the setup becomes that of a Generative Adversarial Network (Goodfellow et al., 2014), though with additional conditions for the generator (the secret bits) which are unknown to the discriminator, and not necessarily employing joint training. Another line of future research is to generate tweets which are personalized to a user type or interest group, instead of reflecting all twitter users. Furthermore, we plan to explore if capacity can be improved even more by using probabilistic encoders/decoders, as e.g. in Matryoshka (Safaka et al., 2016, Section 4).

Ultimately, we aim to open-source our stegosystem so that users of open communication systems (e.g. Twitter, emails) can use our stegosystem to communicate private and sensitive information.

References

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*. Association for Computational Linguistics, pages 69–72.

Ching-Yun Chang and Stephen Clark. 2010. 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, pages 591–599.

Ching-Yun Chang and Stephen Clark. 2012. Adjective deletion for linguistic steganography and secret sharing. In *COLING*. pages 493–510.

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.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. pages 2672–2680.

Michael Grosvald and C Orhan Orgun. 2011. Free from the cover text: a human-generated natural language approach to text-based steganography. *Journal of Information Hiding and Multimedia Signal Processing* 2(2):133–141.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410* .

Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*. Springer, pages 217–226.

Jan Kodovsky, Jessica Fridrich, and Vojtěch Holub. 2012. Ensemble classifiers for steganalysis of digital media. *IEEE Transactions on Information Forensics and Security* 7(2):432–444.

James H Martin and Daniel Jurafsky. 2000. Speech and language processing. *International Edition* 710.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*. volume 2, page 3.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)* 28:1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Pál Révész. 2014. *The laws of large numbers*, volume 4. Academic Press.

Iris Safaka, Christina Fragouli, and Katerina Argyraki. 2016. Matryoshka: Hiding secret communication in plain sight. In *6th USENIX Workshop on Free and Open Communications on the Internet (FOCI 16)*. USENIX Association.

David Salomon. 2003. *Data privacy and security: encryption and information hiding*. Springer Science & Business Media.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. pages 1017–1024.

Umut Topkara, Mercan Topkara, and Mikhail J Atallah. 2006. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions. In *Proceedings of the 8th workshop on Multimedia and security*. ACM, pages 164–174.

Denis Volkhonskiy, Ivan Nazarov, Boris Borisenko, and Evgeny Burnaev. 2017. Steganographic generative adversarial networks. *arXiv preprint arXiv:1703.05502* .

Alex Wilson, Phil Blunsom, and Andrew Ker. 2015. Detection of steganographic techniques on twitter. In *EMNLP*. pages 2564–2569.

Alex Wilson, Phil Blunsom, and Andrew D Ker. 2014. Linguistic steganography on twitter: hierarchical language modeling with manual interaction. In *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, pages 902803–902803.

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

Yingjie Zhou, Mark Goldberg, Malik Magdon-Ismail, and Al Wallace. 2007. Strategies for cleaning organizational emails with an application to enron email dataset. In *5th Conf. of North American Association for Computational Social and Organizational Science*.