Proceeding of the International Conference on Electrical Engineering and Informatics
Volume. 1 Nomor. 1, 2024
e-ISSN : XXXX-XXXX, p-ISSN : XXXX-XXXX, Hal 45-55
Available Online at: https://prosiding.arteii.or.id/index.php/ICEEI

Enhancement of Markov Chain-Based Linguistic Steganography with Binary Encoding for Securing Legal Documents

Halili A.B., Salangsang G.A
College of Information Systems and Technology Management, Pamantasan ng Lungsod ng Maynila, Maynila, Philippines,
Email: ithelp@plm.edu.ph

Abstract. The Markov Chain-based linguistic steganography algorithm can effectively hide information within human-like cover text, but it is highly limited in processing speed. A traditional implementation relying on Huffman tree-based encoding mainly suffers from slow processing due to the computational overhead of building the tree itself. To address this issue, this study proposes an enhanced algorithm using binary indexing for constant time complexity. The results were experimentally calculated using models of varying state sizes derived from the same text corpus as a control variable. Perplexity analysis was also employed to evaluate imperceptibility and ensure there were no drawbacks to the cover media's integrity. The results indicate that the enhanced algorithm improves processing speed by up to 54 times across all state sizes without compromising imperceptibility. This establishes that the enhancements yielded a significantly faster processing speed for the existing algorithm while remaining secure in its concealment. In practice, the algorithm was applied in legal document storage to strengthen its security.

Keywords security, steganography, markov chain

1. INTRODUCTION

Steganography is a technique that allows for the secure transmission of sensitive information over an exposed public channel (Madison & Dickman, 2007). It does this by hiding said information within seemingly harmless media files such as images or video. Along with steganography, another popular way of concealing information is through cryptography. The benefit of the former over the latter lies in its ability to hide that a secret message is being transmitted in the first place (Mishra & Bhanodiya, 2015). This makes steganography a valuable research topic in cybersecurity.

Various steganographic techniques mostly differ in the media they choose to hide information in. One of the most popular mediums used is through text, given its wide use over the internet. Despite its popularity and practicality, though, text-based steganography is infamously hard to conduct. According to Singh et al. (2009), most steganographic techniques use cover media such as images or sounds rather than text because it is much easier to find redundant bits within them. Information is also harder to embed in text because a slight change in a textual cover is more apparent than in a picture or video file cover.

Given its difficulty, the ease of transfer and storage of its cover media is a tangible benefit over other steganographic covers. Thus, a steganographic algorithm using text that maximizes speed without compromising the integrity of the cover media would be

Received: April 29, 2024; Revised: May 30, 2024; Accepted: June 27, 2024; Published: June 29, 2024;

beneficial for a wide range of scenarios and applications. This is the primary motivation of this article.

Specifically, it proposes an enhancement to a Huffman-based steganographic algorithm by Yang et al. (2018), wherein Huffman trees were used to systematically traverse a Markov chain to embed information within human-like text. A major limitation of this algorithm is its processing speed, given the computational overhead it makes constructing a Huffman tree for every word generated by the algorithm. This article replaces that method with binary indexing to achieve a constant time complexity in word generation instead.

The enhanced algorithm would generate words from the Markov chain by interpreting bit groups of varying lengths from an input bit stream without the need to construct a Huffman tree. This should speed up both the encoding and decoding processes of the algorithm significantly, without negatively affecting the convincingness of the text.

2. LITERATURE REVIEW

There are a variety of techniques when it comes to text steganography but there are three major categories that they fall under: format-based, semantic, and language models. Format based systems rely on changing properties within a given cover text such as punctuation or whitespaces to hide information. In a study by Roy and Manasmita (2011), these techniques were documented as being prone to lose information. This is because media like whitespaces and line shifts are dealt with differently by various applications, with some text editing programs removing unnecessary whitespaces for example. They were also found to have less information hiding capacity due to the nature of their cover media.

Another popular category of text steganography is through semantic means. This revolves around embedding information into a text by swapping certain words with another word that essentially means the same. This was demonstrated in a paper by Umut Topkara et al. (2006) where a thesaurus was used to get all the possible alternatives for a particular word. The problem with this technique lies with the fact that sentences may lose context in the process of swapping. While in a vacuum, a word's synonym possesses the same meaning, it can mean differently when used in a sentence.

The last major category of text steganography involves randomly generating text that mimics natural language and using it as a carrier for secret information. A paper by Wayner (1992) utilizes what he coined “mimic functions” to modify a file such that its statistical properties match another file’s. This was done through matching the probabilities of certain substrings appearing in both files.

A functional inverse to this approach is the basis for the algorithm covered in this paper. Moraldo (2024) developed a steganographic application for Markov Chains that utilizes its capability to generate natural language text that mimics a source input, based on bi-grams and their transitional matrices. In this case, the text generated by the algorithm made more syntactical sense than the output of the mimic functions by Wayner.

While text steganography based on random generation can be prone to semantic errors or nonsensical output, it has significantly more information embedding capacity than other steganographic techniques for text (Lockwood & Curran, 2017). To alleviate the incoherence that these outputs tend to exhibit, active research has been done to develop techniques that generate more convincing human-looking text to cover information.

A paper by Mulunda et al. (2013) discussed a technique for generating steganographic text that leverages the principles of genetic algorithms to obfuscate input. It uses concepts that mimic biological evolution that are commonly used to solve optimization problems. It has a widely configurable generation setting, with options for population sizes and chromosome length. However, the steganographic text that it produces does not mimic natural language and is a more conspicuous cover medium, which could affect security.

Another technique detailed in a paper by Kunal Kumar Mandai et al. (2014) utilized a new number system that mathematically divides a given input number into an ordered pair of numbers. The algorithm they proposed takes in the ASCII value of each character as an input and processes it into an output of two numbers. These numbers would then be used to construct a graph in a two-dimensional coordinate system. While novel in implementation, the graphed cover media faces trouble when it comes to transmission since it is not readily available for generic communication channels.

In the algorithm discussed in a paper published by Xiang et al. (2020), the text output is generated on a per-character basis, as opposed to the word level generation

employed by most content-based text steganographic techniques. It uses a long short-term memory (LSTM) based language model trained on a data corpus to construct the output text. The proposed algorithm achieves significantly more compact embedding rates because of its character-level language (CLM) model. However, character-level text generations generally have less cohesion than word-level generations because they tend to produce non-words (Parrish, 2014). This leads to higher perceptibility in the cover text, affecting the security of the algorithm.

3. METHODS

The enhanced algorithm, along with its changes from the existing algorithm underlined, is as follows:

A. Encoding Algorithm

1. Input secret bit stream.
2. Initialize entry point (keyword) list.
3. Index an entry point using the integer equivalent of the next bit group in the bit stream.
4. While it's not the end of the current sentence:
   a. Query the transitional matrix of the current n-gram from the constructed Markov Chain.
   b. Sort the possible transitions based on weight.
   c. Index a transition using the integer equivalent of the next bit group in the bit stream. Always index end points if they exist.
   d. Output the indexed transition and construct a new n-gram
   e. If at the end of the current sentence:
      i. Index an entry point using the integer equivalent of the next bit group in the bit stream.
   f. If at the last bit group:
      i. Set key as the length of the bit group.
      ii. Convert key to ASCII character starting from 97 ('a').
      iii. Append character to random word in the output.
5. Return generated text.

B. Decoding Algorithm

1. Input generated text.
2. For n-gram in generated text:
    a. Query the transitional matrix of the current n-gram from the constructed Markov Chain.
    b. Sort the possible transitions based on weight.
    c. Calculate the max bit length of the encoded index based on the length of the list of possible transitions.
    d. Find the index of the next word and convert it into binary with trailing zeroes to satisfy the max bit length.
    e. If the next word doesn't exist in the transitions, convert its last character to its ASCII decimal equivalent and set it as the end key.
    f. Output the decoded index in binary.
    g. If at last word:
        i. Use the end key to determine the length of the last bit group.
3. Return the generated bit stream.

This article uses an experimental approach to evaluate the listed changes made in the overall performance of the algorithm. Markov models that are derived from the same corpus were used along with the same input bit stream for all measurements comparing the existing and enhanced algorithms. They are also tested using models with different state sizes to have a more complete perspective on their differences. Additionally, both algorithms were implemented without compiled optimizations and parallel processing for a base level comparison. There were two main metrics used to determine the effectiveness of the algorithm.

The first is **processing speed** which denotes how fast the algorithm takes to execute a full encode-decode cycle. This was calculated through the use of Python's built-in *time.perf_counter()* method as a benchmarking tool.

The second is **perplexity** which denotes the convincingness of a text. In the context of Natural Language Processing (NLP), perplexity is a common metric used to test the quality of sentences that a language model generates (Stephen, 2023). It can be interpreted as the geometric mean of the inverse probabilities of a test set, indicating that a lower perplexity value reflects a more convincing text (Payong, 2024). The mathematical expression for it is as follows:

$$PP(w_1, w_2, \dots, w_N) = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} \quad P(w_1, w_2, \dots, w_{i-1})}}$$

where $PP(w_1, w_2, \dots, w_N)$ is the perplexity of the test $w_1, w_2, \dots, w_N$, N is the number of token transitions, and $P(w_1, w_2, \dots, w_{i-1})$ is the conditional probability of a word given the preceding words $w_1, w_2, \dots, w_{i-1}$.

To account for a Markov model's limited context window and the instability of the product of small probabilities present in the equation, this article uses the following derivation as an alternative which is the one used to calculate perplexity in this article.

$$PP(w_1, w_2, \dots, w_N) = 2^{-\frac{1}{N}\sum_{i=1}^{N} \quad P\,(w_1, \dots, w_{i-1})}$$

4. RESULTS

With regards to **processing speed**, the following table shows that the existing algorithm exhibited significantly higher encoding and decoding times than the proposed algorithm at higher bit stream lengths, scaling faster with size of the input.

Table 1. Runtimes of the Existing and Enhanced Algorithms with a State Size 2 Markov Model (in seconds)

	Existing		Enhanced	
Input Length	**Encoding**	**Decoding**	**Encoding**	**Decoding**
10	0.0005	0.0006	0.0000	0.0001
50	0.0018	0.0018	0.0009	0.0010
100	0.0372	0.0435	0.0006	0.0041
500	0.3722	0.3532	0.0062	0.0074
1000	0.5533	0.5235	0.0070	0.0142
5000	2.7889	2.6493	0.0646	0.1064
10000	7.2787	7.0292	0.1011	0.2088

Furthermore, the encoding and decoding speed of the enhanced algorithm is consistently faster than the existing algorithm across different state-sized Markov models. This is most apparent in Markov models that have a state size of 1, where the difference in speed is up to 54 times greater with the enhanced algorithm compared to the baseline. The following figures show this in a logarithmically scaled bar graph to better show relative differences.

Figure 1. Logarithmic Encoding Times of the Existing and Proposed Algorithms with a Bit Stream of Length 10,000

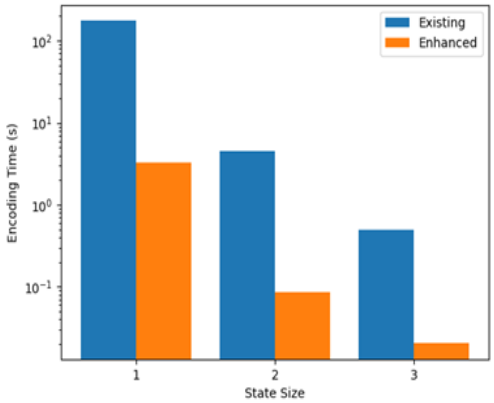
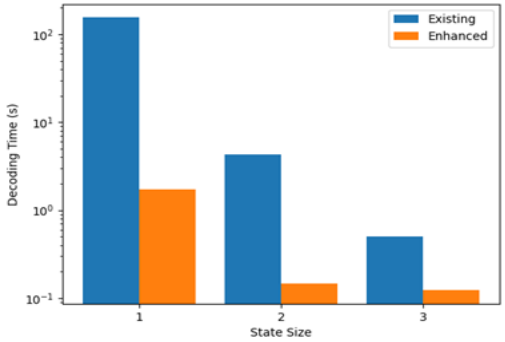
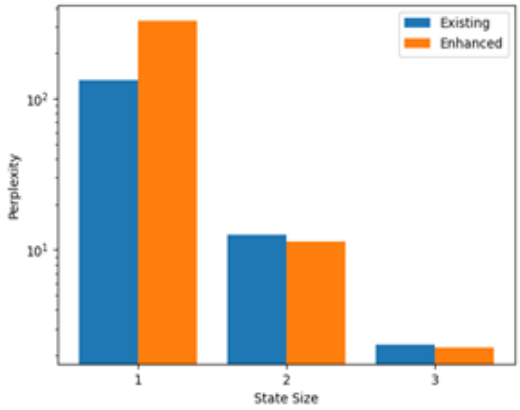


Figure 2. Logarithmic Decoding Times of the Existing and Proposed Algorithms with a Bit Stream of Length 10,000



With regards to **perplexity**, the following figure shows that the existing algorithm has a significantly lower perplexity than the enhanced algorithm at a state size of 1. However, at state sizes 2 and 3, the enhanced algorithm has a marginally lower value. Note that the graph has been logarithmically scaled to better show relative differences.

Figure 3. Logarithmic Perplexity of the Existing and Enhanced Algorithms using Different State Sizes



5. DISCUSSION

Given the relatively less explored field of linguistic steganography compared to the usage of other more common media like images and videos (Chang & Clark, 2014), an algorithm for generating human-like text that has hidden information within it is highly valued. The Huffman-based approach by Yang et al. (2018) fulfills this novelty, but is limited by its processing speed.

This article has shown that the proposed enhancements had a significant effect on the processing speed of both encoding and decoding algorithms. Not only is it faster overall, it also has a better time complexity which means its speed scales much slower with the size of the input.

Additionally, the integrity of the cover media remains intact and within acceptable ranges as shown by the perplexity comparisons of both algorithms. Specifically, in state-size-2 models, the existing algorithm measured a perplexity of 12.5, while the enhanced algorithm achieved 11.34. In state-size-3 models, these values were even lower, at 2.37 and 2.24, respectively — indicating highly human-like text. For comparison, GPT-2, a large language model, has a perplexity of approximately 19.93 (Radford et al., 2019), which was considered favorable at the time, while GPT-4 has been reported to achieve a perplexity of 2.6 (Lukas Görög, 2023).

As for state-size-1 models, both algorithms generated text with high levels of perplexity that are well above the accepted range. In an article by Payong (2024), it was stated that perplexity values higher than 100 indicate that the model has less than a 1% chance of predicting the next token correctly, reflecting a relatively high level of uncertainty in its predictions. Thus, both algorithms generate visibly non-human texts on average if a Markov model with a state size of 1 was used.

6. CONCLUSION

This article showed an enhancement for the Markov Chain-based linguistic steganography algorithm with binary encoding, addressing limitations in the existing algorithm’s computational efficiency. The enhancements resulted in notable improvements in the algorithm's performance whilst preserving the level of concealment of the generated text. This shows that the enhanced algorithm is more efficient and practical than the existing algorithm.

With regards to perplexity, it is important to note that both algorithms produced unconvincing text when a Markov model with a state size of 1 was used for the generation. However, this does not meaningfully indicate a flaw in both algorithms because Markov models with a small context size generally produce more artificial looking text (Healey, 2021).

7. LIMITATION

Metrics not explored in this article but are still factors to be considered for the algorithm's performance include embedding rate, validity, and resistance to steganalysis. While the enhanced algorithm was structured with these in mind, this article mainly focused on maximizing efficiency so these metrics weren't detailed.

Further comparisons can also be made between the two algorithms using different corpora. This will depict a more complete analysis of their differences. This article only used one corpus as a basis because of its large sample size. Experimentation with corpora of smaller sizes or those about a different topic might yield valuable research insights.

8. REFERENCES

Chang, C.-Y., & Clark, S. (2014). Practical Linguistic Steganography Using Contextual Synonym Substitution and a Novel Vertex Coding Method. Computational Linguistics, 40(2), 403–448. https://doi.org/10.1162/coli_a_00176

Healey, A. (2021, January 31). Generating Text With Markov Chains. Retrieved March 22, 2024, from healeycodes.com website: https://healeycodes.com/generating-text-with-markov-chains

Lockwood, R., & Curran, K. (2017). Text based steganography. International Journal of Information Privacy, Security and Integrity, 3(2), 134. https://doi.org/10.1504/ijipsi.2017.088700

Lukas Görög. (2023, March 21). Exploring the Latest Advancements in GPT-4: A Comprehensive Overview - Predictea Digital Care - AI Strategy, Predictions, Data Analysis. Retrieved October 27, 2024, from Predictea Digital Care - AI Strategy, Predictions, Data Analysis website: https://www.predictea.com/exploring-the-latest-advancements-in-gpt-4-a-comprehensive-overview/

Madison, J., & Dickman, S. (2007). An Overview of Steganography An Overview of Steganography. Retrieved from https://digitnet.github.io/m4jpeg/downloads/pdf/an-overview-of-steganography.pdf

Mishra, R., & Bhanodiya, P. (2015, March 1). A review on steganography and cryptography. https://doi.org/10.1109/ICACEA.2015.7164679

Moraldo, H. (2024, March 11). An Approach for Text Steganography Based on Markov Chains. Retrieved April 26, 2024, from ar5iv website: https://ar5iv.labs.arxiv.org/html/1409.0915v1

Mulunda, C. K., Wagacha, P. W., & Adede, A. O. (2013). Genetic Algorithm Based Model in Text Steganography. Erepository.uonbi.ac.ke. Retrieved from http://erepository.uonbi.ac.ke/handle/11295/81186

Parrish, A. (2014, January 14). decontextualize · N-grams and Markov chains. Retrieved from decontextualize website: https://www.decontextualize.com/teaching/rwet/n-grams-and-markov-chains/

Payong, A. (2024, June 10). Baeldung. Retrieved October 27, 2024, from Baeldung on Computer Science website: https://www.baeldung.com/cs/language-models-perplexity

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. Retrieved from https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

Roy, S., & Manasmita, M. (2011). A novel approach to format based text steganography. Proceedings of the 2011 International Conference on Communication, Computing & Security - ICCCS '11. https://doi.org/10.1145/1947940.1948046

Stephen, W. (2023, July 4). Perplexity in AI and NLP — Klu. Retrieved from klu.ai website: https://klu.ai/glossary/perplexity

Umut Topkara, Mercan Topkara, & Atallah, M. J. (2006). The hiding virtues of ambiguity. ACM Workshop on Multimedia and Security. https://doi.org/10.1145/1161366.1161397

Wayner, P. (1992). MIMIC FUNCTIONS. Cryptologia, 16(3), 193–214. https://doi.org/10.1080/0161-119291866883

Xiang, L., Yang, S., Liu, Y., Li, Q., & Zhu, C. (2020). Novel Linguistic Steganography Based on Character-Level Text Generation. Mathematics, 8(9), 1558. https://doi.org/10.3390/math8091558

Yang, Z., Jin, S., Huang, Y., Zhang, Y., & Li, H. (2018). Automatically Generate Steganographic Text Based on Markov Model and Huffman Coding. ArXiv (Cornell University).