Syntactic Steganography

Hardik Rajpal

November 29, 2023

1 Introduction

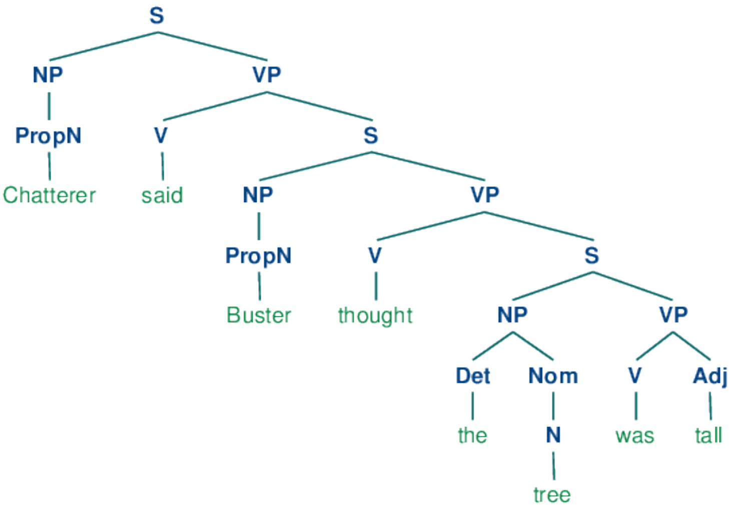
Steganography refers to the concealment of information within a non-secret message or object. The technique has been applied in images with goals of copyright protection and confidentiality (references 6,7). This project explores a form text steganography that exploits the syntactic redundancies of the English language. Given a sentence, it can be paraphrased to a different sentence, while retaining the (almost entire) meaning. Given a **cover text**, one can use the above facts to encode a **message** into it to produce an **stego text**. To extract the message from the output document, the implementation requires the **cover text**.

2 Approaches

This section highlights the first approach, problems and the final approach.

2.1 Syntax-Tree Manipulations

My first approach was to encode bits into sentences based on the structure of their syntax tree. For this, I studied references 1 and 4. The broad ideas were:



1. For each sentence in the cover text, extract its syntax tree.
2. Use a syntax bank to encode message bits into the extracted syntax and obtain new syntax tree.
3. Generate new sentence from old sentence based on new syntax tree.

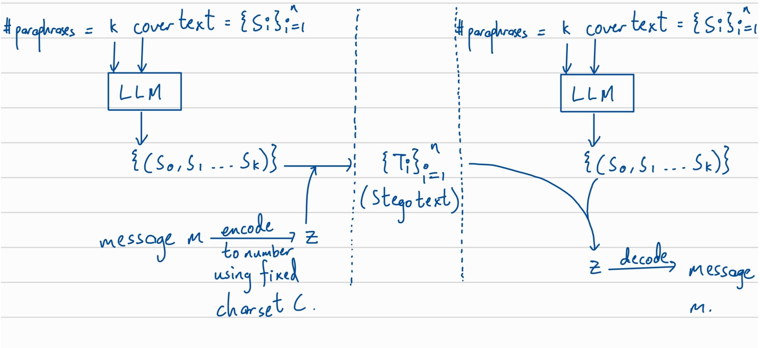
There were two road-blocks in proceeding with this:

1. Unavailability of a syntax-bank as required by reference 1.
2. Poor quality outputs from rule-based paraphrasing (employed in generation of a syntax bank).

2.2 Predictable LLM Paraphrasing

The broad ideas were:

1. For each sentence in the cover text, procure paraphrased sentences by querying the LLM with temperature set to zero.
2. As the responses are predictable (with zero temperature), encode the message using the index of the sentence in the list of sentences generated.
3. Stego text is formed by combining the indexed paraphrases.



# paraphrases = k
cover text = $\{S_i\}_{i=1}^{n}$
LLM
$\{(S_0, S_1 \dots S_k)\}$
message m
encode
to number using fixed charset C.
z
$\{T_i\}_{i=1}^{n}$
(Stego text)
# paraphrases = k
cover text = $\{S_i\}_{i=1}^{n}$
LLM
$\{(S_0, S_1 \dots S_k)\}$
z
decode
message m.

3 Review

3.1 Steganography Dimensions

As highlighted in Reference 1, steganography techniques can be studied with three dimensions. Below I review the project with each dimension:

3.1.1 Payload Capacity

It refers to the ratio of hidden information to cover information. The payload capacity is much better than that provided lexical steganography, where synonyms are used to encode bits (0.31 bits per sentence on average) with the recurring example providing at most 2 bits per sentence for most sentences. Additionally, the payload capacity is flexible and can be tuned based on the text provided.

3.1.2 Robustness

It refers to the ability of the system to resist against changes in the cover object. While the method has near zero robustness if the cover object is shared in docx form, there may exist ways to circumvent this. For example, using a tool like ghostscript to remove fonts from the pdf version (see Reference 2) of the output document (making the text unselectable), assuming there exists a tool to map the nofont version to a font version back. However this has not been explored in the interest of time. Furthermore, HMACs and checksums could be incorporated to protect against substitution of entire sentences in an attempt to modify the message.

3.1.3 Imperceptibility

It refers to the potential of the generated stego object to remain indistinguishable from other objects in the same category. This is one aspect where the method is much better than the existing methods; the sentences produced are equivalent to what you would find in everyday language and are much more natural than the outputs from methods of rule-based paraphrasing (see references 3 and 4) and possibly lexical synonym-based steganography, where alternating synonyms of a repeated word may potential come across as unnatural.

3.2 Limitations

- The method uses intensive computation power, with at least 12GB of RAM.
- The method is rather slow, but the variations once generated can be stored and reused.
- The method requires both the sender and receiver to have a copy of the document. This is fine for, detecting document leaks, it might not be a suitable method for securing a communication channel.
- Furthermore, the method is not resistant to paraphrasing. However, considering that the document is large, paraphrasing it would require extensive computational power, stands in the way of such an attack.

3.3 Takeaways

- Concept of syntax trees in NLP.
- Steganography literature and terms.
- Survey of available paraphrasing techniques.
- Methods for paraphrase generation.
- Using the LLM API provided by `gpt4all`.
- Using the document manipulation API provided by `docx` in python.
- Next time, just do the labs.

3.4 Applications

As initially intended, the method may find applications in document leak source detection. When a user requests to view a confidential document x (cover text), we give him access to a modified document x' (stego text) that has his user id (message) (from a database) encoded into it. Thus, unaltered leaks can be traced back to him by decoding the stego text for his user id.

4 References

1. Lexical Steganography
2. Linguistic Steganography Researchgate
3. Ghostcript makes text unselectable in pdfs.
4. Rule based paraphrasing 1
5. Rule based paraphrasing 2 (Czech language)
6. Image Steganography for copyright protection
7. Image steganography for confidentiality.