Faculty of Informatics and Computer Science

Software Engineering

Linguistic Text Steganography using Transformer Models

By: Haya Gamal Abdel Mohsen

Supervised By

Dr. Abeer Hamdy and Dr. Khaled Nagaty

June 2022

Abstract

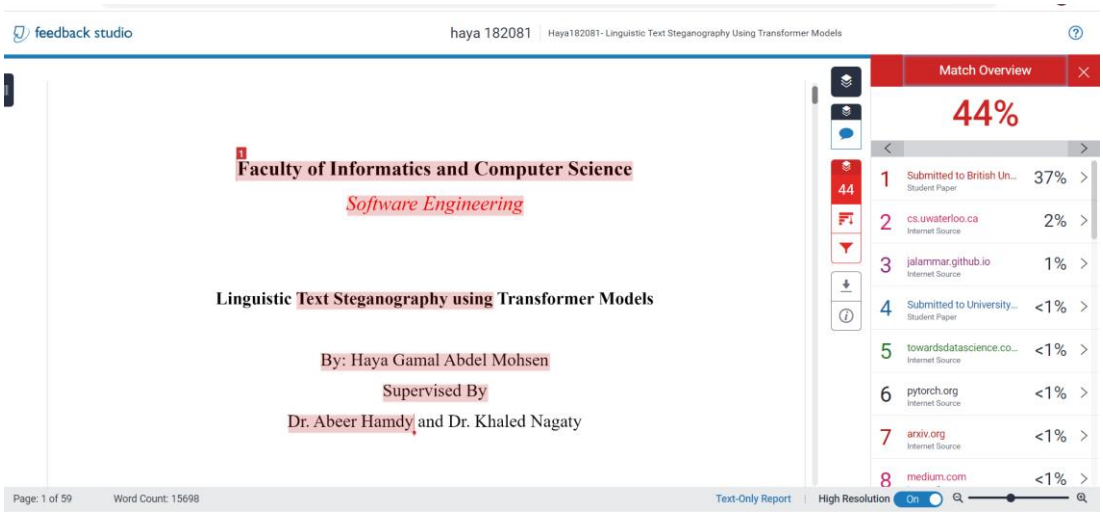
With NLP's great innovation in technological advancements, Text Steganography, which works by unsuspiciously hiding confidential data in innocuous digital text, turns out to have the most considerable attention due to its widespread usage in cybersecurity since that the textual information such as password authentication and banking credentials are considered to be the most sensitive and confidential information that need a solid protection from malicious attacking. Motivated by concerns for user anonymity and privacy, three linguistic methodologies based on Transformer mode architecture, GPT2 a pre-trained transformer language model created by OpenAI, BERT which is an edit-based transformer model and finally, RoBERTa (Robustly Optimized BERT Pre-training Approach) which is a modified version of BERT for training optimization, is proposed to design three stegosystems which embeds a secret text message within a text medium (known as cover message) using their wide capabilities in generating conditional text that embeds the words of the secret message in such unprecedented way to extract the new steganographic text. Using this stegosystem, sender and receiver can exchange encrypted messages that keep their coherency while hiding the secret information undetectably by any third party. This approach is demonstrated on introducing a new steganographic approach using RoBERTa for the first time in comparison with the two pre-trained models, GPT2 and BERT that all has been trained on huge datasets such as Wikipedia, WebText and CommonCrawl News to produce realistic steganographic paragraphs within taking into consideration the maintenance of its context consistency while improving its capacity (encrypted bits per word). The objective of this proposal is to come up with an approach that consumes reasonable capacity percentage of cover text without corrupting the context of steganographic text and at the same time hide as many words and sentences as possible, to output a context coherent steganographic text that attackers will not even manage notice the existence of such embedded secret message.

Attestation &Turnitin Report

I understand the nature of plagiarism, and I am aware of the University's policy on this.

I certify that this report reports original work by me during my university project except for the following (*adjust according to the circumstances*):

The technology review in Section 3 was largely taken from the illustrated GPT-2 (Visualizing Transformer language models) [5], Everything GPT-2 [13] and, The annotated GPT-2 [14], RoBERTa: Robustly Optimized BERT Pre-training Approach [29].



Signature *Haya Gamal Abdel Mohsen* **Date 16/6/2022**

Acknowledgements

This project was done under the supervision of Professor Abeer Hamdy, Software Engineering Department Coordinator, and Professor Khaled Nagaty, Computer Science Department Coordinator, who enlightened me with the greatest help, guidance, insight, comfort, and continuous support throughout this project. Deep appreciation for their time, advice, and patience. I would like to thank TA Israa Lotfy as well for helping me putting my first steps through this project.

I would like to extend my thanks to my parents for always encouraging and having faith in me and providing me with the greatest support throughout my academic journey.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

Steganography is the science of hiding a message within another one without drawing any attention for any anomaly of the carrier message. The term was firstly in 1499 by Johannes Trithemius on his book “Steganographia” which was known as the book of magic. The process is generally hiding messages which is known as “secret message” to be part of another medium such as images, documents, or audio which is known as “cover medium” and sometimes a key is used called as “stego key” in the encoding and decoding processes between two parties. The output of this process is a steganographic text which is a text medium having the secret message hidden inside done using several different techniques. Steganography research done were more likely concerning the embedding of sensitive information within images or audio which in fact, are considered of high steganographic capacity mediums that allows large amount of information to be encoded within making small unrecognized changes to the cover medium. There are a lot of proposed techniques on the images/audio steganography field but the most common one is storing the sensitive data within the least significant bits of the file where it contains the least details of the image/audio file not changing anything in how the file appears to other parties. However, using those mediums is not the best type of steganography to hide critical information due to their widely known decoding techniques among attackers. Text steganography, on the other hand, is capable of securing such critical information.

Despite its lower capacity, text steganography is a secure, faster method for hiding textual critical information as text is a widely used way of communication. Text steganography involves lots of techniques from changing the formatting of an existing text by changing spelling, punctuation, font size, font colour or hide pieces of secret text within white spaces which is called Format based approach, to converting the secret message into their equivalent ASCII values in order to generate a new formula for the steganographic text called Random & statistical generation, to replace words of cover medium with synonyms of secret message words which is known as the Linguistic method.[1]

In recent years, significant advances have been featured in text generation fields using Deep Learning especially works in language models whose main focus is high quality readable text generation that are proposed in many applications such as neural machine translation, image captioning or dialogue systems. From this point, a linkage between high-capacity language models and steganography is proposed in this project at which linguistic steganography can be generalised well hiding not only the content of the secret message, but the fact that a secret

message has been sent at all due to the fluency of paragraphs produced using the three well known transformer models GPT2, BERT and RoBERTa which are pretrained language models of diverse datasets taking the secret message as a sequence of tokens and embedding them in a formulated nonsuspicious context coherent steganographic text which appears as a normal text of any existing topic.

1.2 Problem Statement

In order to deliver a steganographic text in such unnoticeable way is not perfectly accomplished through many text steganography techniques. It is believed that text steganography is the trickiest of all types as encrypting only a small fraction of messages can make the steganographic text suspicious due to its anomaly in appearance because of the deficiency of the redundant bits which is found only in images and audio files creating some capacity limitations. The problem with the format-based technique is the low hiding capacity and low distortion robustness against structural attacks as their hiding algorithms can sometimes lead to failure which can arouse suspicious to a human reader especially using font-sizing. Comparing the plaintext and the resulted steganographic text using Format-based methods makes manipulated parts very visible. Regarding the Random and Statistic generation, although the generated text has similar statistical properties (word length and letter frequencies) with the secret message, generated text does not always provide a coherent meaning which leads to the identification of the hidden communication in addition to its time consumption in encoding and decoding the secret text [2]. This leaves some of the text steganography techniques with capacity limitations and no context coherence which leads to the ease of capturing and understanding by any attacker or third party.

Figure 1 - Example of the failure of Format-based generated steganographic text using font-sizing method

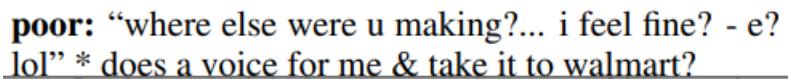


Figure 2 - Example of the failure of achieving context coherence using Random and Statistical generation technique

1.3 Scope and Objectives

This project will focus on implementing a stego-system using a pre-trained transformer model called GPT2 that will solve the two problems which are the most common on text steganographic techniques which are the suspicious steganographic texts that make computer systems or even human reader to easily detect that information data has been encoded/hidden within this plaintext due to the text appearance anomaly or the unrelated linkage between the text words. The project will focus on implementing the three proposed approaches GPT2, BERT and RoBERTa models to enhance the results of these techniques. This will be achieved by employing the pre-trained language models that has proved success in generating appropriate and related context of the input words (secret message) in addition to its capacity payload that can encode as many words as possible without corrupting the context as the smallest GPT2 model has a capacity up to 1024 tokens positions and both BERT and RoBERTa take upto 512 tokens per batch. The objective of this proposal is to compare the three models to get the best out of them that fulfils a great capacity percentage of cover text in order to hide as many words and sentences as possible, to output a context coherent steganographic text that attackers will not even manage notice the existence of such embedded secret message.

1.4 Report Organization (Structure)

Chapter one which is “Introduction” states briefly an overview of what is steganography, different techniques using it, problems with some methods implemented using these techniques, and the objective of the project to overcome those problems using pre-trained language models. Chapter two which is “Related Work” includes background of all techniques used in the project as text pre-processing techniques and background on the proposed model. Moreover, a brief about other steganographic approaches proposed by researchers and examples for further understanding of each approach and the analysis of each approach to point out pros and cons of each of them. Chapter Three “Proposed Solution” contains a precise explanation of the proposed approach of the project and methodologies about transformers models, precisely the GPT2, BERT and RoBERTa and how each processes the input secret to output the desired steganographic text that solves the mentioned problems.

1.5 Work Methodology

The work methodology carried out the following steps:

- Searching for related work to conduct in Literature Survey.
- Analysis of the related work.
- A Research to decide upon a suitable solution methodology.
- Writing Interim Report.
- Implementation of the Stego-system.
- Test the implemented code by conducting several experiments.
- Refining the implemented code to solve debugs.
- Write dissertation for the graduation project.

1.6 Work Plan (Gantt Chart)

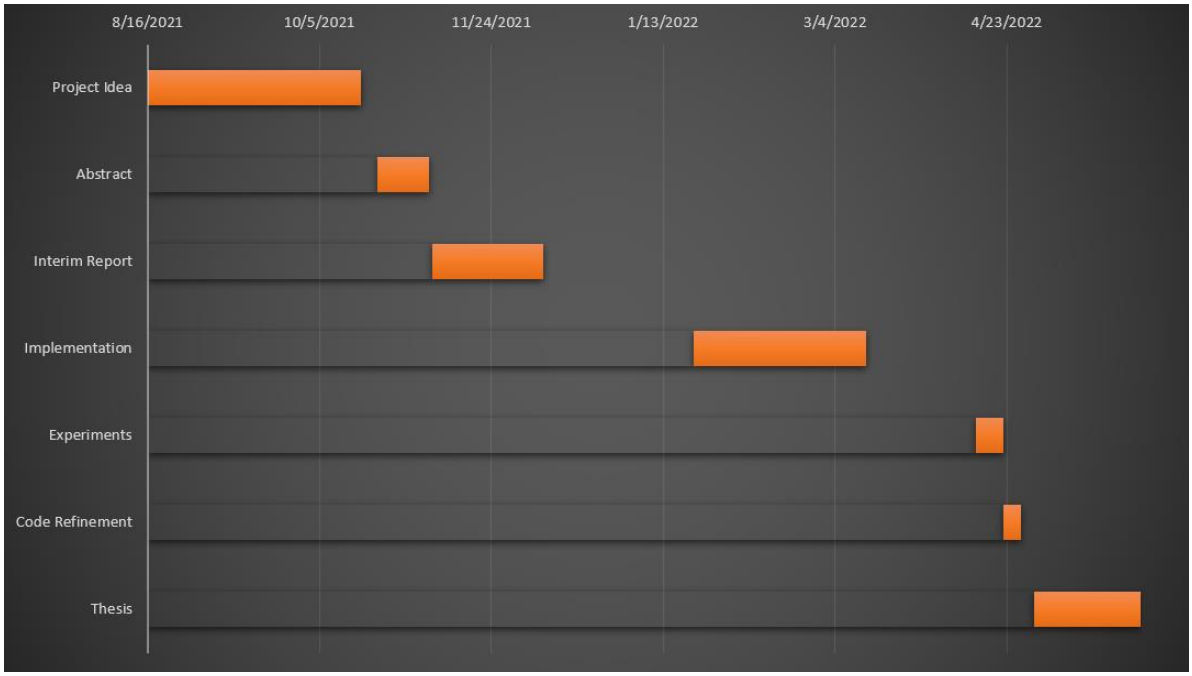


Figure 3 - Graduation Project Work Plan

2 Related Work (State-of-The-Art)

2.1 Background

This section mainly focuses on what experiences or research needed to be considered in order to understand this project and approaches proposed on related work mentioned.

2.1.2 Natural Language Processing

Natural language processing is an artificial intelligence-based solution that concerns giving machines and computers the ability to understand and manipulate textual information and human spoken words.

2.1.2.1 Tokenization

It is a pre-processing common task in NLP in advanced deep learning-based architectures. It is a way of dividing a piece of text (sentence) into smaller units each called token. A token can either be a word, characters, or a sub word. Sub words are considered n-gram characters tokenization at which n refers to number of characters within the sub word [3]. Tokenization plays an important role especially in steganographic purposes as different tokenizers can tokenize and observe the same word differently so it will be tokenized differently that may lead to the model understanding the same word differently so there will be two different contextual embeddings for the same word in two models having the same purpose. This will lead to false understanding of the word meaning leading to false generalization or prediction of the next target word that may result in poor context coherency.

2.1.2.2 Embedding matrix:

A Transformer model takes tokens as an input of only one token embeddings matrix and one positional encodings matrix. Each input word has an embedding at which the model looks up for in the model's **embedding matrix**. Embedding matrix consists of rows, each row is a list of numbers representing a token and some of its meaning which basically convert each number resulted from the tokenization into a vector for a numerical representation for the tokens.

2.1.2.3 Positional encoding:

Unlike RNNs where input words are passed sequentially, it is needed to know the position of each token in the sequence. To indicate the order of the input words in sequence, positional encoding is to be incorporated which is a matrix contains positional encoding vector for each of the 1024 input positions which fulfils the high capacity of hiding secret message within a text medium.

2.1.3 Transformer Models

Transformer model is a neural network that learns context and relationships between words by tracking their relationships in sequential data as words in sentences. Transformers applying a set of mathematical techniques called attention mechanisms to detect stronger level of relationships inspired by the encoder-decoder architecture of RNNs. Unlike RNNs, the transformer does not process data in sequence, which allows for more parallelization and reduces training time. Using attention mechanism along with multi-head attention, the model can deeply understand the relationship between a sequence of words, so it helps in predicting the next target word in an efficient way [4].

2.1.3.1 Attention Mechanism

For each input token, three main vectors are created:

Query: each input token is represented with its query, which is used to score against all other words in order to select the token of the most probability to be related to the input token.

Key: each word has a label known as a key, which matches to the search for relevant words.

Value: once each word is labelled to how relevant it is to the query, values are detailed representation of each word.

to calculate the score of each word in model vocabulary to know how well they match, a dot product calculation is done between query vector of the token searching for its relevant ones by each key vector of each word from the candidates. A matrix is resulted of the scores in order to be slapped on masking process. Masking is a method to hide unwanted small scores that is implemented as a matrix called attention mask. It sets the cells wanted to be masked to negative infinity or a very large negative number. After applying the attention mask, only the large scores are remained on the matrix. Then, a SoftMax activation function is applied on each row in order to produce the actual scores needed for self-attention layer.

2.1.3.2 Multi-Head Attention

Multi-head attention allows the model to jointly attend to the information from different representations at different positions. Multi-head refers to the multiple times where masked self-

attention is conducted on different parts of the key, query, and value vectors those parts are called heads at which each head is a row of each of the key, query, value vectors in order to result a matrix conducting all relevant aspects for this input token. In the previous part, splitting the resulted vector of multiplying the input with the weight of the decoder block gives a long vector for each of the query, key, and value of this input token. Splitting attention heads means reshaping those long vectors into three 12x64 matrices (key, query, value) since 12 is the number of rows (attention heads) that also refers to the 12 attention heads a Transformer base size has.

2.1.3.5 SoftMax Layer

The SoftMax score determines how much each word will be expressed at this position. Clearly the word at this position will have the highest SoftMax score, but sometimes it's useful to attend to another word that is relevant to the current word. The last step in self-attention layer is passing the output of the linear layer to obtain the weights on the values (actual scores) of **each head** of our sequence of words. Those scores refer to how relevant each word in sequence is to the input token. The SoftMax Activation function is the best to be used in our model as it converts the scores to normalized probability distribution to be displayed as a value (actual scores) or probability of how relevant this input token of query Q is relevant to that other token of labelled key K and score V.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Figure 4 – SoftMax activation function

After concatenating all SoftMax results of all of the 12 attention heads, the resulting vector is handed to the fully connected neural network sub-layer to output a single word of the most relevance to the pervious sequence of words as shown precisely in the next section.

3.1.3.6 Fully Connected Feedforward Neural Network Sub-Layer

It is where output of the multi-head function which holds the value weights of the words sequence that shows relevance as it is of a larger weight is processed in order to output the relevant tokens suitable to be placed as the next predicted word. Feedforward neural networks are complex networks consist of input layer that accepts the vector handled on self-attention layer (after projection), hidden layers that capture the hidden correlations and, an output layer which transmits the output. First neural network layer is four times the size of the model (768*4= 3072 neurons) giving the model enough representational capacity to handle all input

tokens and their stored objects. The second neural network layer projects the output of the first one into our model dimensions (768) resulting the transformer block for this token.

2.1.4 Huffman Coding

Huffman coding is a lossless data compression algorithm. It is used to assign variable-length codes to characters of the input. These assigned codes lengths are based on each character's frequency among the corresponding characters. The most frequently repeated characters get the smallest code, and the least frequently repeated characters get the largest code. This text compression technique is used mainly to reduce the bit sequence size of secret message to be encoded so it consumes a reasonable capacity payload to avoid corruption of context.

2.2 Literature Survey

In this section, Existing approaches are presented on the text steganography area that are scoping on different important concerns that will be used in this proposal.

S. Iyer In paper [7] proposed an Alphabet Pairing technique at which each alphabet of the data of both secret message and cover text is converted to its ASCII equivalent value of each alphabet so that a scanning process on each individual alphabet is done for resemblance with its ASCII equivalent value to pair with. A secret message of N alphabets is to be hidden in an original message of M words. A division is done between the 2 counts (M/N), the divisibility indicates that each alphabet of the secret message is to be embedded in odd-indexed alphabets of original text. Meanwhile, the indivisibility indicated that each of the secret message is to be embedded in the even-indexed alphabets of the original text. The embedding capacity of this approach is of 10.5% per word of original text. The following table sums up how this technique works:

	Alphabets	**ASCII Value**	**Pairing of Alphabets**
Original Text	i	**01 10 10 01**	*01 00 01 00*
	a	01 10 00 01	
	m	01 10 11 01	
	h	01 10 10 00	
	e	01 10 01 01	
	r	01 11 00 10	
	e	01 10 01 01	

	Alphabets	ASCII Value	Pairing of Alphabets
Secret Message	D	*01 00 01 00*	**01 10 10 01**

Number of words in original/number of alphabets= 3/1=3 so it is divisible. 'D' is to be embedded in 1st odd-indexed alphabet of the 1st original text word.

Meanwhile, B. Gupta and S. Kumar in paper [8] concerns were mainly on the lexical side of generating the steganographic text Using the Part-of-Speech Tagging technique in NLP. Their method was focused on keeping the original meaning of the cover text by lexical substitution which is replacing a word by a word of the same part of speech tag; "verb from cover text" is to be replaced with "verb from secret text" etc. By this method, it will not be recognizable that the text has a hidden secret text within, as it avoids any anomaly in the context coherence of the steganographic text by trying to craft cover text in the related context of secret text. The Location of the secret message words replacement in cover text is calculated by $\text{Count POS}_{\text{cover}}/\text{Count POS}_{\text{secret}}$ where POS means the count of this part of speech within the cover text (for example 5 nouns) divided by that of the secret text (1 noun). So that 1 noun in the secret text will be replaced in the position of the 5th noun out of the 5 nouns of the cover text (5/1=5).

Cover Text: "February comes in the winter season"
Secret Text: "meeting is in January"
Steganographic text: "January is in the meeting season".

In [9], Hiding Data in Paragraphs approach was presented using the conversion to ASCII method as [7]. It begins with converting the secret text message firstly into cipher text, which is an encryption algorithm used on plain text, then convert this cipher text into its corresponding binary values. Then, the algorithm reads the words on the cover file, sequentially, and writes it on the stego file which holds the steganographic text. If this word is of the same starting and ending letter "example: mom", it is skipped. Moving forward into the binary bits of secret message along with the words of the text in stego file. If bit =0 then the start letter of the pointed word is in the stego key else if bit = 1 the ending letter of the pointed word is in stego key. The process moves on along with bits of secret message and words of cover text until the end of the bin file. A stego key is obtained that can be used by the second party to extract the secret message.

Since texting families and friends became a daily habit, paper [10] proposed an abbreviation text stegosystem is proposed by M. Hassan through the SMS-Texting to generate steganographic messages. As all texters now use abbreviations as a fast way to deliver your

meaning like “ASAP” for “As soon as possible”, using them to hide secret message within their binary equivalence will not be recognized at all for any other third party. Firstly, the secret message to be hidden is converted into ASCII values as [7],[8]. A dictionary of words with the corresponding abbreviations is implemented. Sequentially going through those binary values, if the pointed bit is 0 then the algorithm will place this word as its complete form. Meanwhile, if the bit is 1, then the algorithm will place the abbreviated word in the sentence to hide this bit. //no context coherence

Another approach using ASCII code to generate steganographic text is proposed by U. Shaw in [11]. The approach reads the secret message text and convert each alphabet of it to the corresponding ASCII value. Then, it combines each two characters to be converted into a 16-bit concatenated value. After that, it divides this concatenated ASCII into 5 parts of speech that form a full sentence such that an article (a, the, an) is assigned to the 1st bit, a noun into the next 5 bits, 4 bits for verb another one for an article and the last 5 bits for an object as shown in the figure bellow which is the ASCII for two-character concatenation of “Hi”.

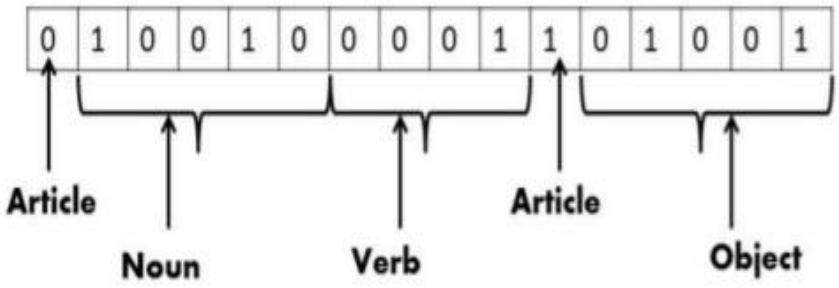


Figure 5 – Example of Linguistic Grammar Approach to Textual Steganography

Using a pre-defined dictionary tokens bin, where each word is assigned to bits block of size according to the partitioning in the previous part (ex: nouns are assigned to 5 bits length), the algorithm checks for the words of equivalent bits and part of speech type to replace it where at the end, a full sentence is generated. So, the output for secret message “Hi” will be “A teacher pushed the table”.

Lastly, LSTM is well-known for its NLP innovations in text generation tasks, not only this but in steganographic purposes since it is one of the efficient generation-based steganographic methods that aims to directly output the steganographic text by generating a serious words based on a language model. An approach was proposed with title Generating a steganographic text using LSTM in 2017, the approach starts with taking the secret message then converting it into bit sequence and using the vocabulary bin in LSTM which is considered the shared key

between the sender and receiver in addition to the LSTM language model, the secret message bit sequence is divided into chunks of equal size each corresponds to a bit block that consist of list of random tokens chosen randomly [12].

2.3 Analysis of the Related Work

After giving a brief explanation for each approach and how each one concerns the aiming of a different aspects whether high capacity, context coherence or applying a secure steganography process, the analysis of each one and the discussion regarding the findings and results is presented in this section.

7 researchers have conveyed 6 text steganography techniques to achieve different concerns. Approach proposed in paper [7] which is alphabets ASCII equivalent pairing is considered a create success for hiding individual alphabets of 10.5% capacity per word. However, this capacity percentage is not enough concerning large secret messages as it treats the sentences as sequence of characters and cannot be applied on embedding a full word within a word resulting a confusion in decoding the secret message back as a full meaningful sentence.

For approach in paper [8], the correct part-of-speech replacement aimed was successfully achieved using this approach which depends on the Part-of-speech tagging to formulate the steganographic text carrying the same meaning of the secret text. However, the output text is of a poor structural vocabulary which may appear confusing and attention catcher to some observers. So, the approach did not achieve the best results in provide ng context coherence and providing an understandable output.

The usage of ciphertext conversion was a great step to achieve security in paper [9] showing impressive results that the hiding process is done without any change in the cover text as shown in the figure below. this paragraph technique can work on any English plaintext and hides the secret message without altering the cover text. However, the downside of this approach is the capacity, the paper shows that the average percentage capacity of this technique is 2.075% which will be very difficult to hide a large secret message. On the other side, it is the most secure of all the presented approaches as it does not appear as a steganographic text so even attackers will not pay attention to such thing to try to decode it.

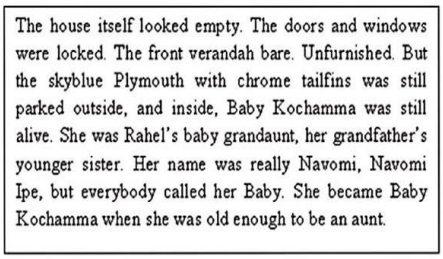


Figure 5. The Cover File

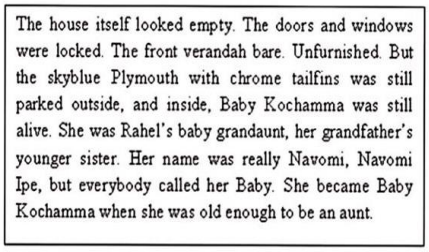


Figure 6. The Stego File

Figure 6 – Results of paragraph technique

The abbreviation conversion approach in paper [10] proved the secret information exchange success on converting the cover words into their abbreviations according to the value of secret message bits. However, on some cases the output may appear as a spam if it contains a lot of abbreviations which disobey the importance of context coherence to be achieved in steganography process. Similarly, the Linguistic Grammar approach presented on paper [11] results the same downside such that the longer the secret message the more repetition of the same sequence and structure of sentence that may attract observers' attention to the anomaly of the output text appearance.

Lastly, paper [12] which proposed the well-known language model for text generation tasks, the LSTM has shown several cons in using it for steganographic purposes. It stated that the LSTM training process is very slow so parallel training can not be performed leading to consuming a lot of time for training only. Moreover, the steganographic text produces lacks logic due to the short memory that is resulted from limited memory size, so LSTM is not considered a good fit for generating steganographic text of long size as it lacks long dependencies between words which leads to poor context coherency in its results.

3 Proposed solution

All steps for understanding the proposed project are clearly illustrated on this section starting from the methodologies, the implementation, to the comparison between the three models showing the pros and cons for each in this domain and how each one of them solve the problems stated compared to the other approaches other researchers used in the related papers taking into consideration how to overcome the faults those approaches have of low capacity, no context coherence which makes the steganographic text widely attracting that it may be hiding a secret message due to its anomaly.

3.1 Solution Methodology

Three Steganography approaches are proposed using deep learning: Generation-based approach (**GPT2)** and Edit-based approach: **BERT** and finally, **RoBERTa**, which will be the focus for this study as it was not used before in steganographic purposes. GPT2 is an unsupervised deep learning transformer-based language generative models taking a textual input as a word/sentence for formulating a single purpose of which is predicting the next word(s) in a sequence of words (sentence). At the end, it has the ability to just take a word as input to generate multiple paragraphs of different context taking into consideration choosing the suitable words that fulfil the correct meaning of sentence and hides the secret message formulating a context coherent paragraph or sequence of words (the steganographic text) and this is what this approach aims. The three transformer pre-trained models process input tokens using a powerful multi-headed self-attention mechanism to capture the internal dependencies of the input sequence in order to generalise a likely continuation of sequence of inputs to be paraphrased into another format of context coherent text by outputting the token with the most probability (score) of relevance to the right-handed sequence of words [5][13][14]. Another two approaches are proposed which are from the same type, Edit-based Transformers, the BERT and RoBERTa which are of the same Encoder only architecture, replacing the masked self-attention layers with self-attention layers as its bidirectional learning from words at both sides of the target token, this creates large range of relationship understanding between words within same sequence that benefits the steganography process. They are cover-based techniques at which the cover text where the secret message is to be hidden inside is given firstly to the tokenizer which masks a percentage from the cover message at random at which those masked tokens are to be replaced with tokens completing the context of the sentence and at the same time encodes the secret message within it using Huffman coding. Although both edit-based models are of the same architecture. However, RoBERTa shows better results due to some modifications on how BERT is trained and processes the input.

3.1.1 Generation-Based Approach

GPT2

Generative Pretrained Transformer 2 (GPT2) is a large transformer-based language model that trained on massive datasets of 1.5 billion parameters, 8 million webpages from Reddit throughout different sources such as trending posts, their comments and content of the attached webpages links which in fact are more then 40GB of text data. GPT2 is an unsupervised deep learning transformer-based language generative models taking a textual input as a sentence for formulating a single purpose of which is predicting the next word(s) in a sequence of words (sentence). The model follows an auto-regressive nature at which after predicting each token, that token is added to the sequence of inputs and then this new sequence is the input to the next step of the model. GPT2 has the same Transformer decoder only model with absolute position embeddings on the masked self-attention layer which means that mainly focuses on the tokens placed on the right side (present and previous tokens) of the point where predicting the next word occurs only, as shown on the figures below.

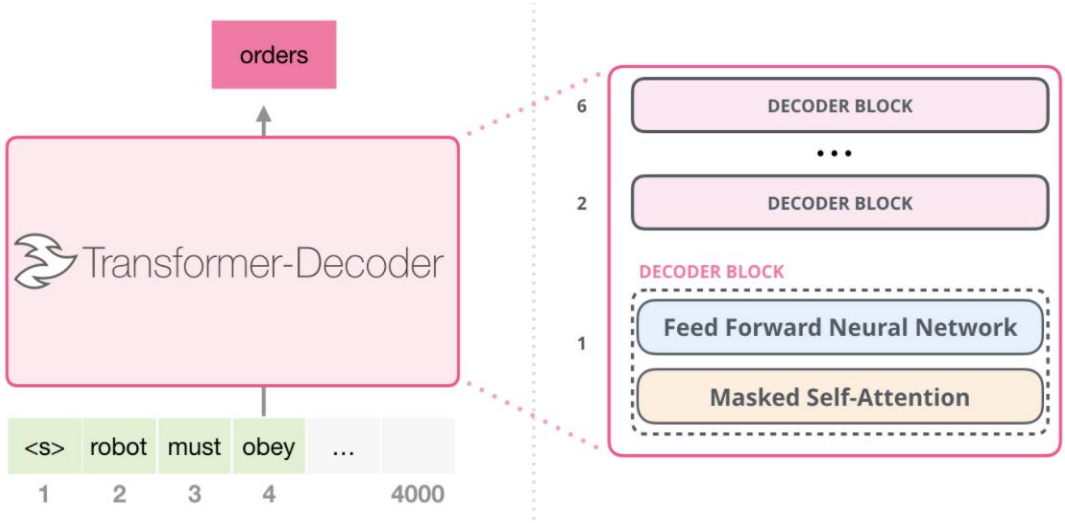


Figure 7 - GPT2 architecture

GPT2 runs depending on the rambling on its own which called generating unconditional samples. It means that the start token is handed as an input (referred by <s>) and the model starts generating words context coherently to it. All the decoder layers have the same structure at which each consist of 2 sublayers: a masked-self attention layer and its feed forward neural network, as shown on the 1st figure. It takes the input token, and successively stacking it to the above and processes it through each layer producing a vector along this path. When the top block produces its output vector, the model multiplies it by the embedding matrix resulting in outputting a single word. GPT2 model continues iteration until the entire context of 1024 tokens is generated [5].

3.1.1.1 Masked Self-Attention Process in GPT2:

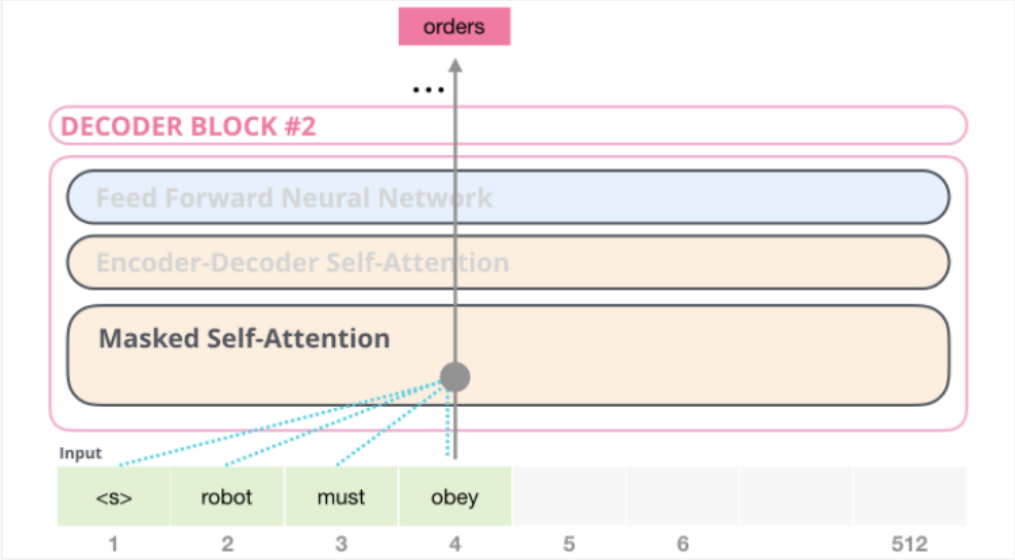


Figure 8 - Masked self-attention layer (allowing only past and present tokens as inputs to predict next word(s).

GPT2 masked self-attention only allows model to peak on the right side of word input sequence in finding the most suitable word that is relevant to the context of the right sided tokens of the sequence only. The masked self-attention layer is located in each of the decoder blocks of the GPT2 where each token passes through along the path. The purpose of self-masking is to make sure that the states of the word 'must' for example, in the figure above, do not attend to see tokens after it 'in the future' and only focus on those on the right side of it, so it only learns the relationship with them only not what's after it. This means that for generating the first token, you cannot attend to anything, when generating the second token, you can attend to states of the first token. At the third one, you can attend the first and to the second.

How does the proposed GPT2 apply Masked Self-Attention using Multi-Head Attention mechanism for Text Steganography?

GPT2 masked self-attention allows only to glean information from prior words in the sentence including the word itself taking an input of only one token as it saved the past words context as object properties [11].

The same steps of transformer model are applied here with slight changes. Starting with processing the first token as mentioned above, the input token is passed to the first decoder layer, specifically to the self-attention layer which holds on vectors of both the key and value of this token. Each self-attention layer has a respective key and value vectors for this token and saves them for reusing later when it is combined with the previous sequence of words as an input for the next iteration to generate a new word. The model, after processing the token, takes the re-

sult of the summation of it's embedding and the position encoding for its position as the input token for the bottom decoder block. As each decoder block has its own weights, this weight is used in order to create the queries, keys, and value vectors for each token. Weights also represent the important parts of data as the more important tokens, the larger weights they are given for the sake of more attention. Therefore, the self-attention layer of the first decoder block multiplies its input token by its own weight matrix resulting a vector that is a concatenation of all needed vectors, so it is to be split in order to have each of the query, key, and value vectors of this input token as shown in the figure below.

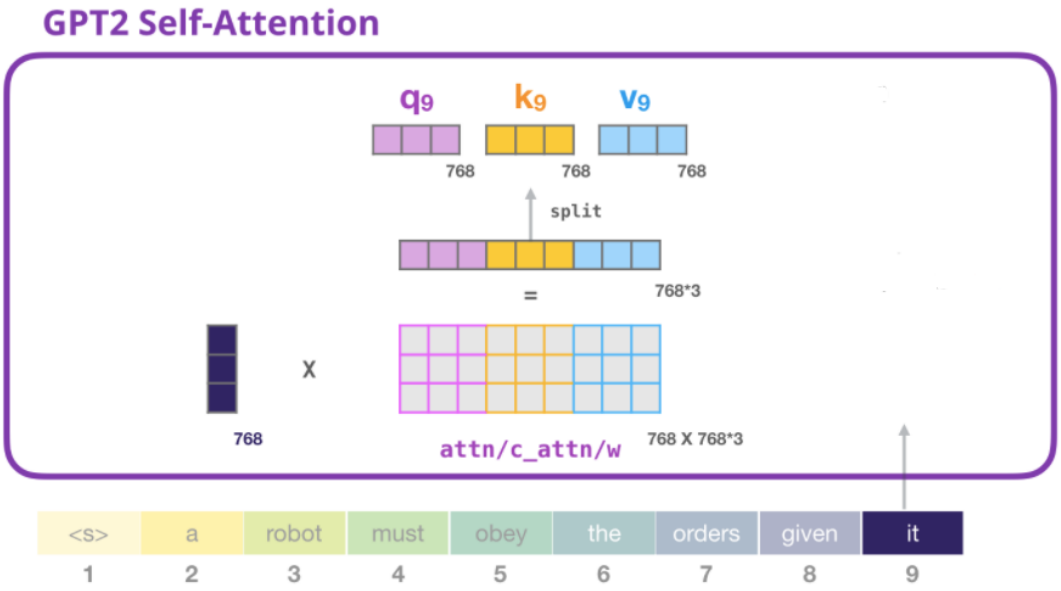


Figure 9 - 768 refers to our GPT2 model dimensionality

3.1.1.2 Multi-Head Attention Mechanism

Now the self-attention layer will go through several steps in processing the token that will be shown precisely per one head since all the upcoming illustrated attention steps will be applied parallelly to all of our GPT2 12 heads of each of the query vector of our newly to be predicted token and each head of the key and value vectors of our sequence of words. Finally, after applying self-attention steps on each head, merging all those heads by concatenating them using the following rule so the result is passed to the feedforward neural network sublayer.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Figure 10 - the output vector of this function is a concatenation of each probability distribution resulted from SoftMax function including all tokens scores.

Here are the steps that happen in the attention layer using **Scaled Dot-Product Attention per one attention-head** as shown in the figure below.

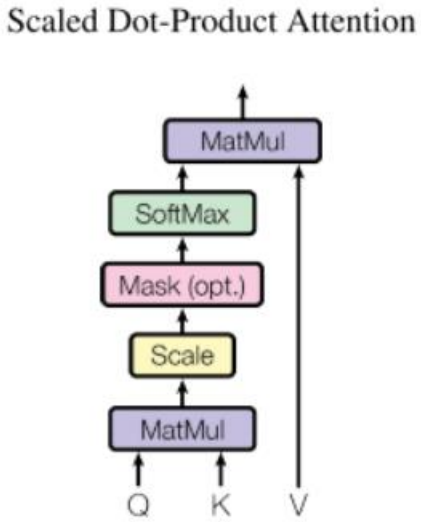


Figure 11 - Scaled Dot-Product Attention

3.1.1.3 Scoring

Concerning one attention head (as each attention head of the twelve conducts the same operation), Query 1 will be searching for the most relevant key of largest score that is calculated by dot product of each key with our query's value vector as mentioned above. The score represents how relevant this key's token is to our query. A summation is done between all dot product results giving Z which refers to self-attention for 1st attention-head. After conducting the same steps to all of our models 12 attention heads, a concatenation is done to convert then into 1 vector. The resulting vector is to be projected in order to be sent to the next sublayer (neural network layer).

3.1.1.4 Projecting (Masking)

Converting the concatenated self-attention results from masked self-attention layer into a matrix holding the most important fitted tokens values is done by projecting. Projection is basically the attention masking process mentioned above, it is done by multiplying the concatenated self-attention results with our second large weight matrix that projects all low scores into zeroes or negative infinity giving concerns to the large scores as they are of the most relevance to our query token.

Figure 12 – Masking Process

3.1.1.5 SoftMax Activation

The SoftMax score determines how much each word will be expressed at this position. Clearly the word at this position will have the highest SoftMax score, but sometimes it's useful to attend to another word that is relevant to the current word. The last step in self-attention layer is passing the output of the linear layer to obtain the weights on the values (actual scores) of **each head** of our sequence of words. Those scores refer to how relevant each word in sequence is to the input token. The SoftMax Activation function is the best to be used in our model as it converts the scores to normalized probability distribution to be displayed as a value (actual scores) or probability of how relevant this input token of query Q is relevant to that other token of labelled key K and score V.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Figure 13 – SoftMax activation function

After concatenating all SoftMax results of all of the 12 attention heads, the resulting vector is to be converted into list of tokens using Byte-Encoding Tokenizer which is list of candidates for next word position and then handled to Huffman Coding to choose the suitable word based on the following considerations.

3.1.1.6 Huffman Coding for choosing output word candidates

Picking the suitable token out of the probability distribution of tokens through a method called Huffman Coding. Huffman coding is considered the main step for achieving **steganography**. The secret message is sent to the Huffman coding converter, converting it into a compressed bit sequence such that payload capacity (number of bits being embedded inside text) is reduced so that it can embed as many bits as possible. And since GPT2 is a generation-based

steganographic approach which is not restricted by size of a cover text. A random word is given of a specific context as an input to GPT2 to generate words sequence of same context. For each position, a list of candidates is resulted that are multiplied by SoftMax activation function to give each a probability distribution of how suitable each word is according to a score. Here comes the Huffman Coding role, the encoded secret message which is resulted from Huffman coding is divided into bit chunks, each chunk is then converted into its decimal equivalent which will hold the index of the chosen word out of the list of candidates as the next predicted word and the process **continues** until a steganographic text is generated of the **specified length** that achieves context coherent paragraph that includes the words of our secret message in a way that does not show any anomaly in our steganographic text also, does not appear as if there is a secret message hidden within this text at all [5][13][14].

Generation based Approach (GPT2)	Edit-based Approach (BERT)
× Uni—directional Model	✓ Bi—directional Model
✓ Coverless Steganographic Method	✓ Cover-based Steganographic Method
✓ More payload capacity than BERT	× Payload capacity is limited by cover text size
× One word/sentence input	✓ Cover text and secret message as inputs
× Stego texts of lower Accuracy than BERT	✓ Reasonable accuracy of the output stego text

Figure 14 - Generation-based Vs Edit-based

A comparison was proposed between Generation based Approach represented by GPT2 and Edit-based Approach represented by BERT in paper [27], stating the important factors that makes Edit-based approach a better fit for steganographic purposes. It shows that edit-based approaches are of bi-directional method in understanding the relationship between words from both sides of target word. However, Generation-based approach only look on the left-sided sequence of words. Since generation-based approach is not restricted by cover text size, so it can encode as many bits of secret message as possible, however edit-based approach are cover based techniques that is restricted with its size. This comparison was a motivation to propose two of the edit-based techniques which are known for their innovation in language models.

3.1.2 Edit-Based Approaches

3.1.2 BERT

BERT (Bidirectional Encoder Representations from Transformers) was an inflection point discovered in 2018 by Hugging Face. It showed a great way of representing words and sentences

and at the same time, capturing the underlying meanings and relationships between words. BERT's innovation key was focusing on both sides of a token (bidirectional) than focusing only on the left-handed side word as implemented in GPT2. BERT is an encoder only Transformer that improves upon standard Transformers by removing the unidirectionality constraint by using a **masked language model** (MLM) pre-training objective. Therefore, it is trained bidirectionally to read the text input all at once, this characteristic allows the model to learn the context of one word based on all its surrounding both left and right words. The model is implemented in two sizes: $BERT_{BASE}$ that has 12 encoder layers, 12 attention heads and 768 hidden units and $BERT_{LARGE}$ that has 24 encoder layers, 24 attention heads and a larger feed-forward-networks of 1024 hidden units. Our focus for this study will be on $BERT_{BASE.}$ During pre-training, the model is trained on unlabelled data over different pre-training tasks such as Masked Language Model and Next Sentence Predication (NSP). For steganographic purpose, the focus will be on the Masked LM for hiding the secret message that will be converted into bit sequence and embedded within the cover text that will be the model's input for further understanding of its words relationship for generating a context coherent steganographic text. As shown in the figure below, BERT can take up to 512 token batches as an input that gets stacked over the encoder layers which each holds a different weight to help in generating the candidates of highest probability distribution to be a replacement for the masked tokens using multi-head attention mechanism. The output of each encoder layer is a representation embedding for each token during the learning process of relationships and features between words until reaching the last encoder layer which results contextual embeddings for each candidate token. For choosing the suitable token, Huffman coding will be used which will take the sender's secret message convert it into bit sequence then divide it into bit chunks whose equivalent is considered the index of the word to be chosen as the replacement and at the same time, holds a part of the secret message.

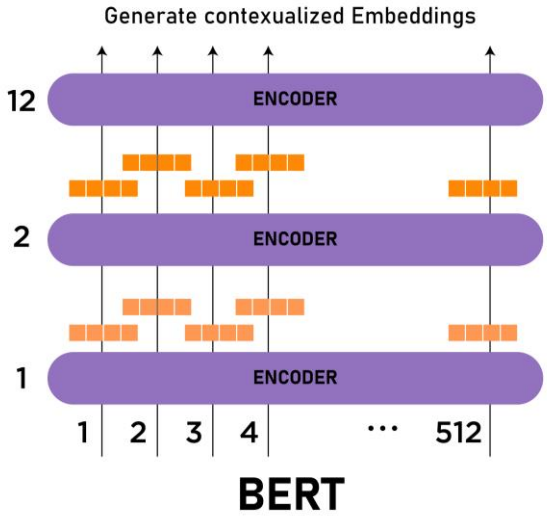


Figure 15 - BERT Architecture

As shown above, the word ‘playing’ is broken down into 2 words ‘play’ and ‘##ing’. This helps in limiting the size of vocabulary by avoiding keeping various forms of words. Moreover, it helps with words that are out of vocabulary, so if the tokenizer does not hold the word ‘ing’, there’s still an embedding holding the meaning of the word ‘play’ so that the meaning of each token in input is not lost.

3.1.2.1 Pre-Training of BERT

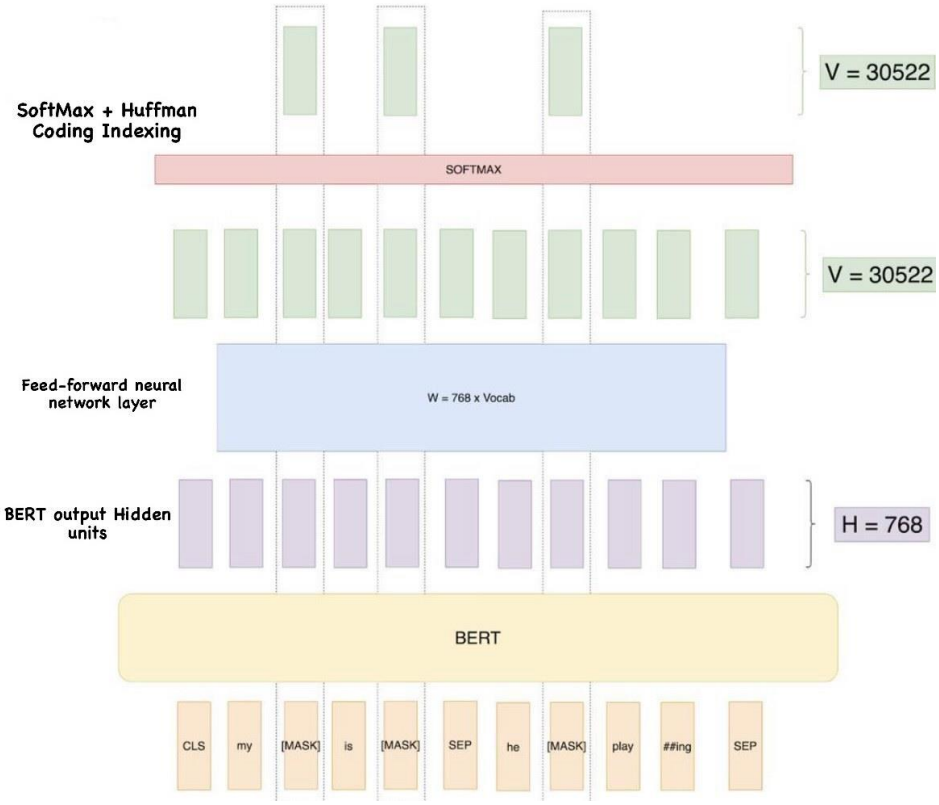


Figure 16 - BERT Architecture in Steganographic Purposes Using Masked Language Model Pretraining Task

BERT has been pre-trained statically using two unsupervised tasks, Masked Language Model and Next Sentence Prediction. Next Sentence Prediction is mainly for answering questions and or bots so in this study, we will focus on the **Masked Language Model** which is the main step for applying linguistic text steganography. BERT is pre-trained bidirectionally which allow each word to “see itself” in order to predict the next target word throughout the multi encoder layers. For deep bidirectional representation, 15% of all WordPiece tokens from the input sequence, the cover text, is masked at random [15]. Masking is the process similar to fill in the gaps which are represented in [MASK] token. Considering the example in the figure above, as humans, according to general world knowledge, we can come to guessing the right words to fill in the mask tokens but how BERT does that? BERT do not know the actual words, but it does know the linguistic patterns given as word embeddings and the context of the rest of input sequence that helps in predicting the **logits**, which are the suitable list of candidates which are the final hidden vectors corresponding to masked tokens are fed into the output SoftMax

which then a candidate is chosen per [MASK] token using secret message that was converted into bit sequence using Huffman encoding.

A Closer look into BERT's Encoder Layers

Attention Mechanism: How words are related to each other

As BERT is built from 12 encoder layers stacked upon each other, each encoder layer has an attention layer and feedforward neural network layer. Each attention layer consists of 12 attention heads. Therefore, at each layer, a token from input sequence can focus on 12 distinct aspects of other tokens. As this study is focused on $BERT_{BASE}$,so there are 12*12 = 144 attention heads at which each head on each layer focuses on different kind of constituent combinations of relationships between tokens. Moreover, a token understands the [SEP] which refers to the end of the 1st sentence of input sequence and used to indicate the absence of matching also, [CLS] which refers to the end of all tokens within input sequence. This shows how BERT trains a token to understand its referral to words within the same sentence it is located at and allow attention heads to detect specific structures where a composition would be appropriate [30].

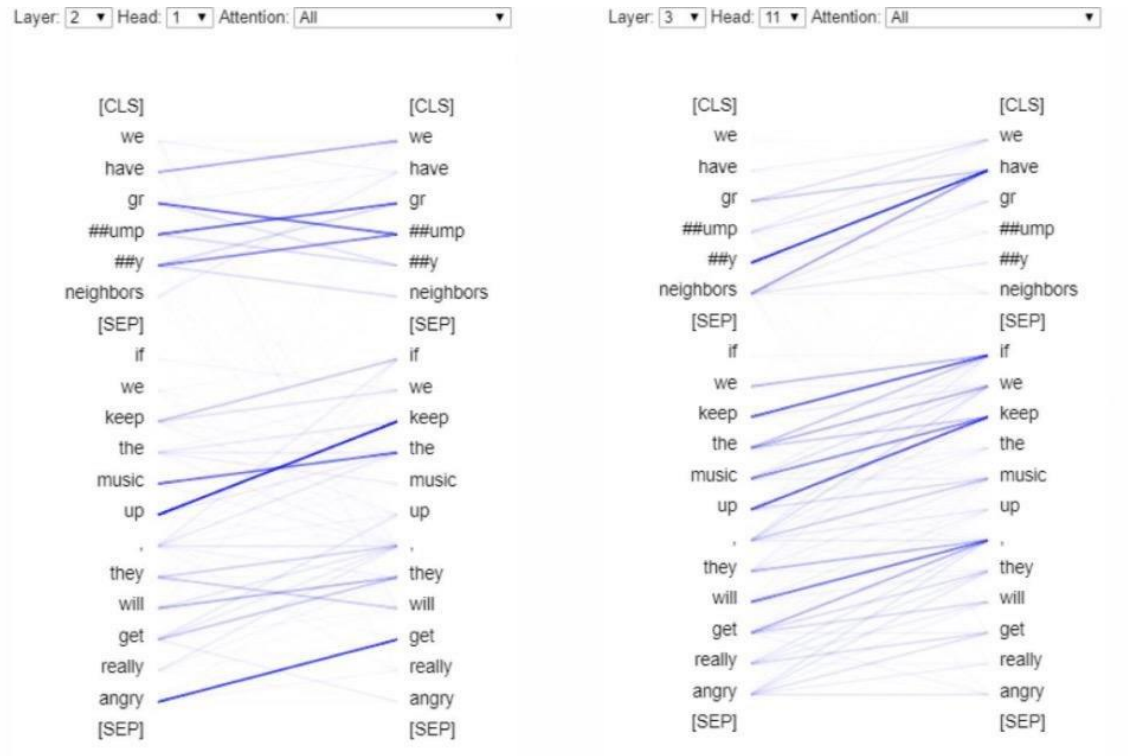


Figure 17 - BERT Attention Mechanism

As shown in figure above, the 1st attention head in encoder layer 2 focuses on the relatedness of each token to another. The right-sided figure, In the 11th attention head of 3rd encoder layer shows a higher level of constituents: how tokens attend to same central words (if, keep and have). Surprisingly, some attention heads can perform conference resolution referring to the

global context of each word. The output of all attention heads are concatenated and being fed to a neural network representing complex nonlinear functions that performs various compositions and returns a paved composed representation of the word [16][30].

Note that BERT learns to predict the missing word based on attention to its previous and forthcoming words, Hence, BERT uses the contextual embeddings resulted from the attention layer in order to predict the [MASK] token. The process continues upon the 12 encoder layers each of higher-level constituents such as verb/adverb, noun/adjective compositions till a list of candidates is resulted as the best match called as **logits** [15].

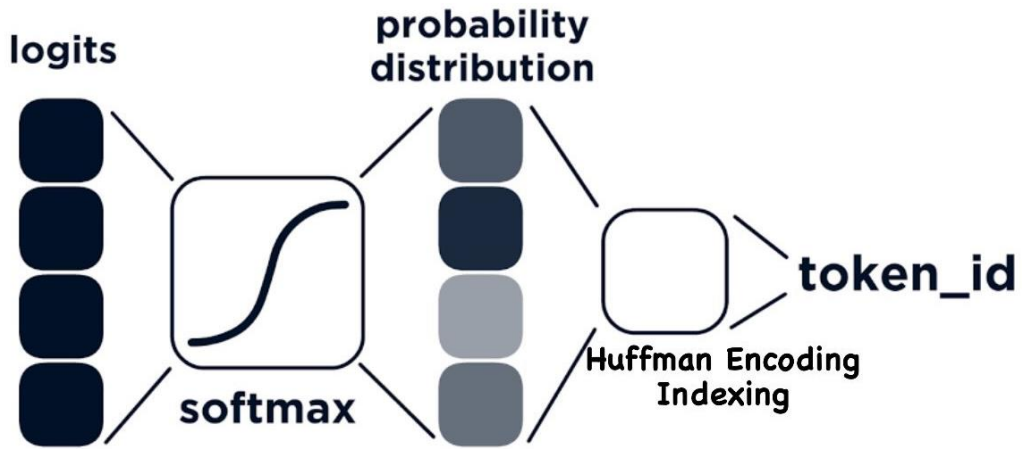


Figure 18 - the logits — which has a vector length equal to the model vocab size. The predicted token_id is extracted from this logit using a SoftMax and Huffman encoding Indexing.

for an input sentence, BERT outputs an embedding for the whole sentence in addition to giving embeddings for every word of the sentence. The logits are the output of the BERT Model before a SoftMax activation function is applied to the output of BERT. By applying a SoftMax onto the output of BERT, we get probabilistic distributions for each of the words in BERT's vocabulary. Words with a higher probability value will be better candidate replacement words for the mask token. Here comes the secret message hiding, As mentioned above in the GPT2 that the secret message is compressed and converted to bit sequence by Huffman Coding. As SoftMax output is a list of candidates, the bit sequence is divided into bit chunks, each chunk is then converted into its decimal equivalent which will hold the index of the chosen word as shown in figure above, (**token_id**), out of the list of candidates as the next predicted word and the process **continues** until all [MASK] tokens found their replacement that is of a high probability distribution and at the same time encodes part of secret message within it [15].

3.1.3 RoBERTa

RoBERTa stands for Robustly Optimized BERT Pre-training Approach. Researchers at Facebook and Washington University presented this model at which it has the **similar architecture** as **BERT** with some simple design changes in its architecture and training procedure [19]. In

this study, the focus will be mainly on RoBERTa as it was not proposed before as text steganography approach that showed efficient results.

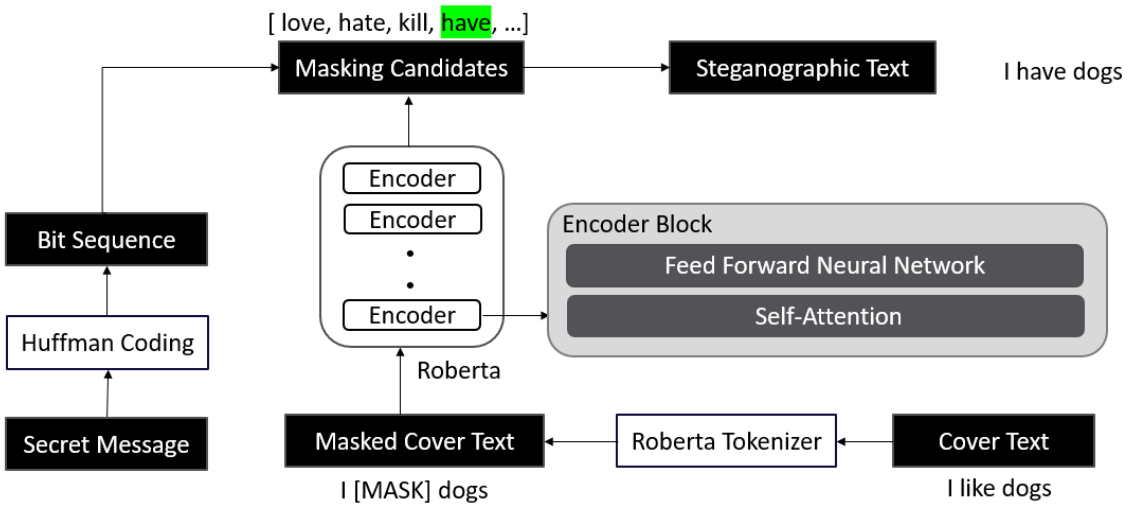


Figure 19 - RoBERTa Architecture for Text Steganography

3.1.3.1 RoBERTa Tokenizer

RoBERTa uses a different tokenizer than BERT which is Byte-Pairing encoding similarly to what GPT2 uses. Tokenization plays an important role especially in steganographic purposes as different tokenizers can tokenize and observe the same word differently so it will be tokenized differently that may lead to the model understanding the same word differently so there will be two different contextual embeddings for the same word in two models having the same purpose. This will lead to false understanding of the word meaning leading to false generalization or prediction of the next target word that may result in poor context coherency as shown in comparison figure below. The Byte-Pairing Encoding relies on pre-tokenization which is equivalent to space tokenization. After that, a set of unique words is created of which each word's frequency has been determined. Consequently, the words are split into symbols. And the tokenizer begins to try symbol pairing and the most frequent pairs are merged and learnt by tokenizer until the merges represent words and applied to new words as long as those new words are not already learnt before until the vocabulary size, that is a hyperparameter to choose, becomes full of unique merges [28].

3.1.3.2 Modifications to BERT

	BERT	RoBERTa
Training Time	64 TPU chips x 4 days	4-5 times more than BERT
Performance	Outperforms GPT-2	2-20% improvement over BERT
Data	16 GB BERT data (Books Corpus + Wikipedia) 3.3 Billion words	160 GB (16 GB BERT data + 144 additional GB from GPT2 (WebText) and CommonCrawl News
Method	Masking Language Model Next Sentence Prediction	Masking Language Model
Tokenizer	Word-Piece Example: n-ug-gets	Byte-Pair Encoding (as GPT2) Example: n-uggets
Masking Strategy	Static Masking	Dynamic Masking

Figure 20 - BERT Vs RoBERTa

A comparison showing how RoBERTa improves the BERT results by following specific modifications which are [21]:

1.Removing the Next Sentence Prediction (NSP) Objective

Next sentence prediction was used to is a task that deciding whether sentence B is the actual next sentence that follows sentence A or not. In the original BERT pretraining procedure, the model observes two concatenated document segments, which are either sampled from the same document or from distinct documents. But it was shown that using individual sentences hurts performance on downstream tasks. Alternative training formats where experiences with and without the NSP [19]. The **first** experiment was segment-pairs using NSP at which the input of BERT has a pair of segments separated using the [SEP] tokens which is the original format for any BERT input with the NSP loss. Total length of input sequence must be less than or equal 512 tokens per batch. The second model input format in the experiment is the sentence pairing which each input contains a pair of sentences either from one document or from separated documents, so they don't have to be related. Since the inputs are shorter than normal size of one batch 512 tokens, so the batch size was to be increase so that the total number of tokens remain similar to segment-pair format. Third one is the full sentences input format with no next sentence prediction (NSP), each input is packed with full sentences sampled contiguously from one or more documents such that the total length is at most 512 tokens and inputs may cross document boundaries. The last model input formal is the document sentences that is similar to full sentences except that inputs can not cross document boundaries so some input batches may be of size small than other batches (less than 512 tokens) and it is with no next sentence prediction (NSP) as well [20][21].

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

Figure 21 - results of masking with different input formats

in the table above, a comparison is done of the original segment-pair format to the sentence-pair format. Both formats have the next sentence prediction task. It shows that using individual sentences in next sentence prediction objective hurts performance on downstream tasks because the model is not able to learn long-range dependencies. The second setting in the table compares training without next sentence prediction and training with blocks of text from a single document. It shows that this setting performs better than the original BERT results and removing next sentence prediction slightly improves downstream task performance. Finally, restricting sequences to come from a single document performs slightly better than packing sequences from multiple documents. However, because the document-sentences format results in varied batch sizes, the rest of the experiments use full-sentences format for fair comparison [19]. So, this shows that better results were given without using related sentences as model is not able to learn on long dependencies, Conditioning on unrelated context from another document adds noise to masked language model and lastly, A model benefits from longer full-length contexts (full sentences) [29]. And this was shown in the results of Roberta as it gives best results for full long length sentences to achieve context coherency.

2.More Data, Larger Batch Size and Train Longer

BERT was originally trained for 1 million steps with a batch size od 256 sequences. For better results, RoBERTa is trained with 125 steps for 2,000 sequences. This has two advantages as the large batches improve perplexity on masked language modelling objective. RoBERTa is trained with 160 GB including Book Corpus + English Wikipedia (16 GB), CC-News (76 GB), OpenWebText (38 GB) and Stories (30 GB) comparing to the only 13 gigabytes of data that BERT is trained on [20][29].

3.Masking Pattern/Strategy

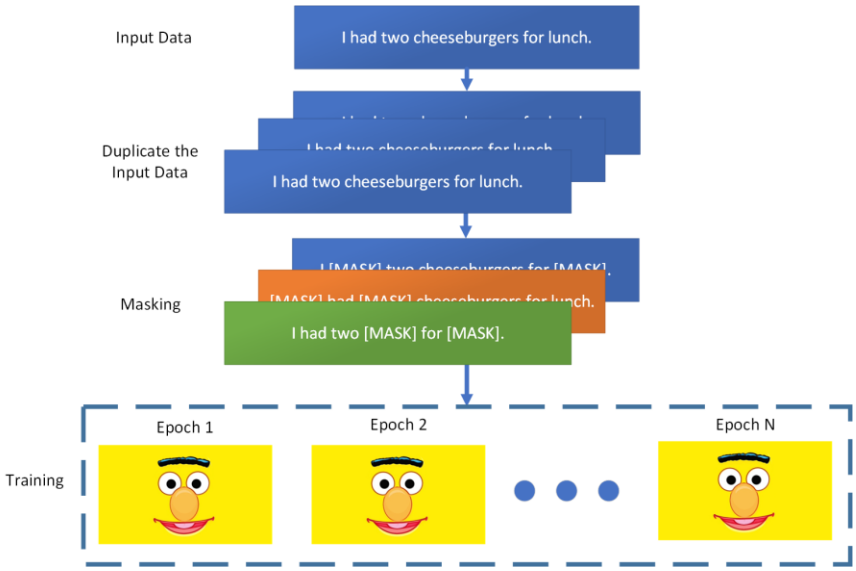


Figure 22 - BERT Static Masking Strategy

In BERT architecture, masking the cover text during data preprocessing in BERT Tokenizer results in a single static mask which means same input masks are fed to the model on every single epoch. This happens as BERT duplicates the same input data 10 times masking each one differently and passes the 10 into the training process once among all epochs so that if the model have 40 epochs each masked sequence was seen 4 times with the same mask. This means that there is not enough learning of relationships of words. Dynamic masking, introduced in RoBERTa has shown impressive training process at which it generates a new masking pattern on every epoch. By this strategy, the model learns much more about replacements for each mask token on each different masked input sequence so learning rate increases and RoBERTa can creatively predict replacements words giving a context coherent text.

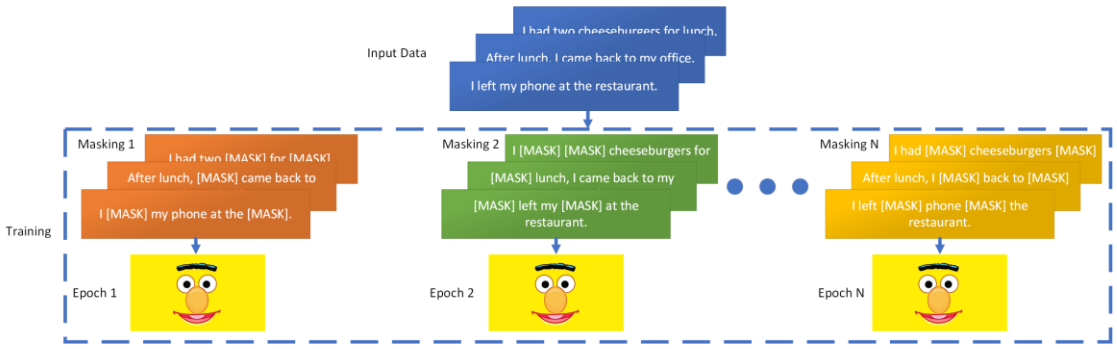


Figure 23 - RoBERTa Dynamic Masking

All these modifications showed how RoBERTa can be a great fit to be introduced for the first time for steganographic purposes for achieving one of the hideous problems faced, the context coherency, due to poor training of models [29].

4 Implementation

4.1 Libraries

The three models were implemented using Python programming language, PyTorch Transfomers and transformers. Pytorch-Transformers is a library of state-of-the-art pre-trained models for NLP. Most of the state-of-the-art models requires tons of training data and take tons of days of trainings on expensive GPU hardware which can only be afforded by big technology companies[22]. So, by using this library, the models are easily utilized. Multiple libraries were used among the previously mentioned libraries for implementing the solution methodology models. Wordpiece library was used for BERT's tokenizer. Lists were used as cover text tokens, candidates and steganographic text tokens where output as lists. StringIO is used as an in-memory file-like object [23]. It is used as an input for the model at where the secret message is to be stored in for extracting the index at which the candidate is place in candidates list achieving hiding the text within the other text. NLTK was used to extract English language stop words to be removed from cover text after being tokenized. A PyTorch Tensor is basically the same as a numpy array: it does not know anything about deep learning or computational graphs or gradients and is just a generic n-dimensional array to be used for arbitrary numeric computation. The biggest difference between a numpy array and a PyTorch Tensor is that a PyTorch Tensor can run on either CPU or GPU [24]. Torch library must be of version 1.7.0 to act up with transformers library that is supported by Tensorflow 2.0. Sublibraries of Transformers library were used in this project as GPT2LMHeadModel for utilizing of GPT2 pretrained model. BertForMaskedLM and RobertaForMaskedLM for using masked language model task on both pretrained models. BertTokenizer, RobertaTokenizer and GPT2Tokenizer which are used as tokenizers for pre-processing of cover text. Heaps are binary trees for which every parent node has a value less than or equal to any of its children. Heaps are used for Huffman coding the secret message [25]. Python's heapq module can be especially helpful when one has to repeatedly select the smallest element of a list, and this is what concerns compressing the secret message into as small a bit sequence as possible for consuming least payload capacity.

4.2 Datasets

Model	Datasets
GPT2	WebText (**40 GB**)
BERT	Books Corpus + Wikipedia (**16 GB**)
RoBERTa	CommonCrawl News + WebText + Book Corpus (**160 GB**)

GPT2 is pretrained using language modeling on a very large corpus of approximately 40 GB of text data from WebText corpus of over 8 million documents scraped all outbound link from Reddit.

BERT is pretrained on Books Corpus and Wikipedia which is equivalent to 3.3 billion words and 16 GB of text data considered as the smallest dataset among the three proposed models.

RoBERTa uses 160 GB of text for pre-training, including 16 GB of Books Corpus and English Wikipedia that was used in BERT. Additionally, it included CommonCrawl News dataset which contains 63 million articles of approximately 76 GB, Web text corpus (38 GB) and stories from Common Crawl of 31 GB.

4.3 Pre-Processing of Cover Text

Tokenization In GPT2

GPT2 relies on Byte-Pair Encoding that was introduced in Neural Machine Translation of Rare Words with Sub-word Units in 2015. BPE relies on pre-tokenization which is equivalent to space tokenization. After that, a set of unique words is created of which each word's frequency has been determined. Consequently, the words are split into symbols. And the tokenizer begins to try symbol pairing and the most frequent pairs are merged and learnt by tokenizer until the merges represent words and applied to new words as long as those new words are not already learnt before until the vocabulary size, that is a hyperparameter to choose, becomes full of unique merges [28]. So, whenever the tokenizer takes a sentence, whose words are already learnt, it tokenizes the sentence normally into set of tokens used as an input for GPT2 model. GPT2 has a vocabulary size of 50,257 corresponding to 256 bytes of tokens. Using tokenizer.encode(init_tokens, return_tensors='pt'), the input, which is the random word that holds the context at which we want to generate a sequence upon it is tokenized using the BPE tokenizer which returns the tokenized sequence in terms of tensors.

Tokenization in BERT

WordPiece embeddings were used as BERT tokenizer with a 30,000 token vocabulary to tokenize the input into set of tokens each of an input representation embedding. The first token of every input sequence is always a special classification token [CLS]. As one of the great features of BERT is that it can take sentence input which refers to pairs of sentences or a single sentence. To differentiate paired sentences, we separate them with a special token [SEP]. BERT has the ability to understand how a sentence is separated from another sentence in the process of learning the attention between words. The model puts for every given token a learned embedding that indicates whether it belongs to sentence A or sentence B in input sequence. So, in the tokenization process, for each token a summation between token

embedding, segment embedding, and position embeddings results the contextual embedding which is the representation embedding for each token in cover text. The output of this summation is BERT's input as shown in figure below [31].

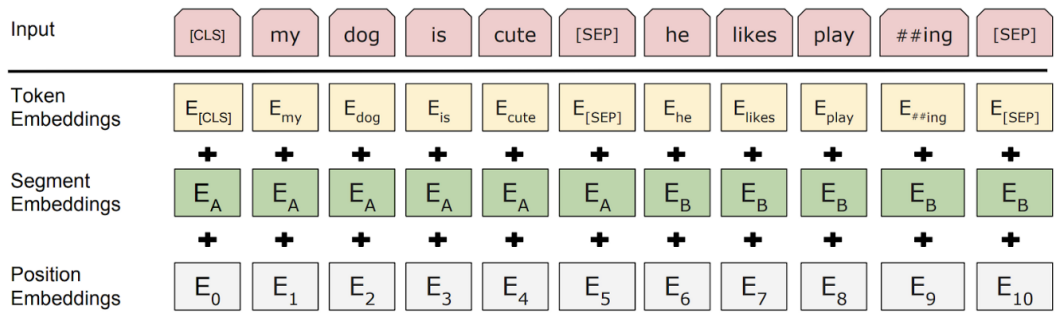


Figure 24 - the hashtags refer to sub-words in WordPiece tokenizer.

RoBERTa Tokenizer

RoBERTa uses a different tokenizer than BERT which is Byte-Pairing encoding similarly to what GPT2 uses [31]. Tokenization plays an important role especially in steganographic purposes as different tokenizers can tokenize and observe the same word differently so it will be tokenized differently that may lead to the model understanding the same word differently so there will be two different contextual embeddings for the same word in two models having the same purpose. This will lead to false understanding of the word meaning leading to false generalization or prediction of the next target word that may result in poor context coherency as shown in comparison figure below. The Byte-Pairing Encoding relies on pre-tokenization which is equivalent to space tokenization. After that, a set of unique words is created of which each word's frequency has been determined. Consequently, the words are split into symbols. And the tokenizer begins to try symbol pairing and the most frequent pairs are merged and learnt by tokenizer until the merges represent words and applied to new words as long as those new words are not already learnt before until the vocabulary size, that is a hyperparameter to choose, becomes full of unique merges [28].

4.4 Hyperparameters and sequence of model

Generation-based approach (GPT2)

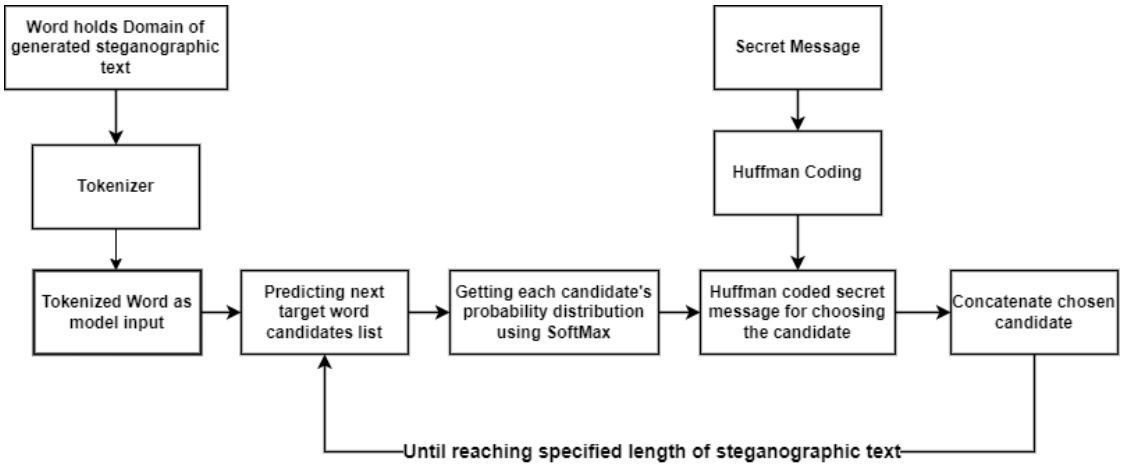


Figure 25 - Generation-based Approach Implementation Steps (GPT2)

The generate function which is responsible for generating the list of candidates takes the following parameters at which the candidates are the output of this function:

1. **max_length=1 + idx + init_len :** this is where the length of the steganographic text is adjusted which equals to the sum of the length of tokens
2. **num_return_sequences=TOKEN_COUNT**: returned steganographic text will be equal to length adjusted
3. **num_beams=5:** the search beams which are the number of generated steganographic text will be equal to 5 so that we choose upon the probability distribution.
4. **early_stopping=True**: whenever the words generated are being repeated there will be an early stopping.

The candidates here are considered the next predicted word and this function is looped on until it reached the specified length of steganographic text which is generated by concatenating the chosen candidates based on the Huffman coding then decoding the output text into tokens using **tokenier.decode(tokens[0],skip_special_tokens=True).**

Edit-based approach (BERT and RoBERTa)

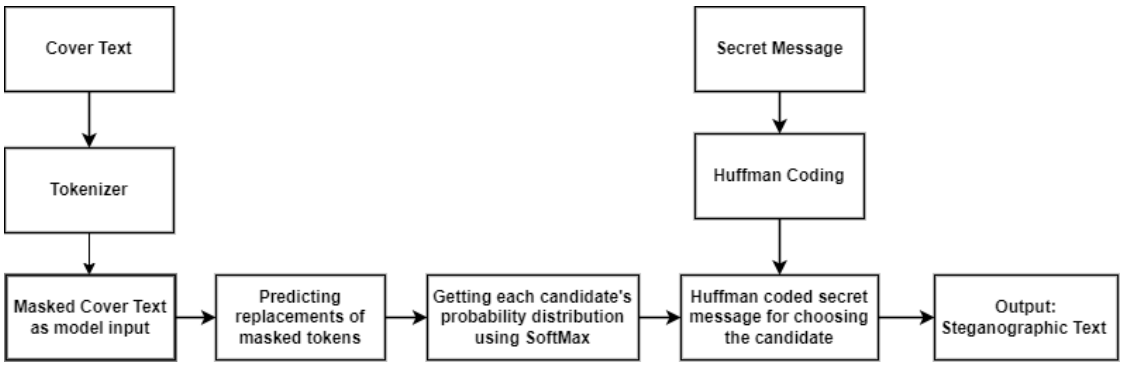


Figure 26 - Edit-based approaches Implementation Steps (BERT and RoBERTa)

masked_stego(val, tobits, 3, 0.01) : this is the model's class for edit-based approach at which it takes **val** (the cover text), **tobits** which is the message after being converted to bits by Huffman coding, and **3** is the **masking_interval**, **0.01** is the probability **score_threshold** at which it is fixed in both models for outputting outputs of comparable masking replacements. An Assertion is done on the secret message to make sure it is in terms of binary bits and then get stored within a **StringIO()** function to store it in file. After that, the tokens of the cover text is saved as the **encoded_is** ant the masking process happens within a clone of these encoded_ids and stored in **masked_ids**. Then, the masked_ids are taken as parameters for the **predict()** function at which the attention heads layer process the relationship between tokens of cover text to get each token's contextual embeddings that help in predicting the [MASK] tokens. The **predict()** then takes the tokens in terms of list of type Tensor and then evaluates the model at which the candidates are resulted as outputs which are in shape of word embeddings from the feed forward neural network. Then to get probability distribution of how much each word of the output candidates is related to the context of the cover text given, the softmax() function takes the output list of candidates of **dim=1** as this list is a unidimensional list then returns the list as sorted of **descending=True** at which the candidates of highest probability distribution is to be listed first then gradually till the word of least probability distribution. Now, a filtering process is done on the list of candidates to choose only candidates whose score is greater than or equal the **score_threshold** also it eliminates stopwords, subwords and check if all tokens are of alphabetic structure using the **_pick_candidates_threshold()**. Now the text steganography process begin using the Huffman coding.

4.5 Steganography implementation within Transformers

Picking the suitable token out of the probability distribution of tokens through a method called Huffman Coding. Huffman coding is considered the main step for achieving **steganography**. The secret message is sent to the Huffman coding converter, converting it into a compressed bit sequence such that payload capacity (number of bits being embedded inside text) is reduced so that it can embed as many bits as possible. And since GPT2 is a generation-based steganographic approach which is not restricted by size of a cover text. A random word is given of a specific context as an input to GPT2 to generate words sequence of same context. For each position, a list of candidates is resulted that are multiplied by SoftMax activation function to give each a probability distribution of how suitable each word is according to a score. Here comes the Huffman Coding role using **Indexing()** function, the encoded secret message which is resulted from Huffman coding is divided into bit chunks, each chunk is then converted into its decimal equivalent which will hold the index of the chosen word out of the list of candidates as the next predicted word and the process **continues** until a steganographic text is generated of the **specified length** that achieves context coherent paragraph that includes the words of our secret message in a way that does not show any anomaly in our steganographic text also, does not appear as if there is a secret message hidden within this text at all. Regarding Edit-based approaches which are cover-based so we have to give the cover text as an input firstly for tokenizer in order to get masked by 15% of its tokens, after that, the masked cover text is handled to the model as given along with the secret message to be stored in StringIO file, the masking intervals, and score_threshold which is the probability set at which any word whose probability distribution is more than it will be taken as one of list of candidates. After that, the model predicts the list of candidates upon the attention heads process discussed in details in architecture, Here comes the Huffman Coding role using **Indexing()** function, the encoded secret message which is resulted from Huffman coding is divided into bit chunks, each chunk is then converted into its decimal equivalent which will hold the index of the chosen word out of the list of candidates as the next predicted word and the process continues along with the number of masked words.

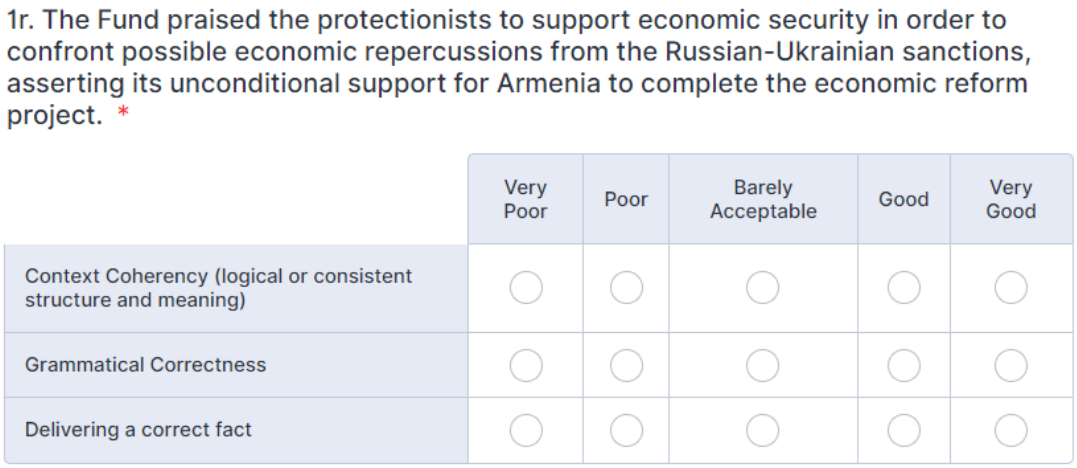
5 Results and Discussions

5.1 Experiments

5.1.1 Data

For GPT2, the word input for the model in order to predict the upcoming sequence of words upon its domain is selected from 4 journal articles from 4 different domains: Academic, Egyptian Politics, Medical, and finally from consolidated topics. A word is randomly selected out of the 4 topics text files and given as an input to the model. Each model has to output a stego text for each of the domains mentioned for reasonable comparison. On the other side, Edit-based approaches proposed can specify the context or domain where we want to hide the secret message by providing a cover text of the required domain.

5.1.2 Human Evaluation



1r. The Fund praised the protectionists to support economic security in order to confront possible economic repercussions from the Russian-Ukrainian sanctions, asserting its unconditional support for Armenia to complete the economic reform project. *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	○	○	○	○	○
Grammatical Correctness	○	○	○	○	○
Delivering a correct fact	○	○	○	○	○

*Figure 27 - Screenshot of Survey, Rest of the Survey is shown in **Appendix***

A survey was conducted on a group of students to give 5-point scale ratings on resulted stego-texts from the three proposed models to give an evaluation about which model give the most non-suspicious sentences according to **grammatical correctness**, **context coherency** and **delivering a correct fact** for political stego-texts only. Each model's hyperparameters were tuned to generate stego texts with comparable length. Hyperparameters were as follows: probability distribution threshold = 0.01 and both stop words and sub words were skipped. Participants were asked to rate texts (each model propose 4 stego texts for each domain) with a Likert scale from 1 which is the lowest to 5, as shown in the table below, the ratings were described with the instructions shown for each aspect. The survey consists of fifteen questions, five stego texts for each model. Thirty-four students have participated in doing the survey.

Rating	Description
5	Very Good
4	Good
3	Barely Acceptable
2	Poor
1	Very Poor

Human Evaluation Results

Histograms were implemented showing the Human evaluation results on

1. **Grammatical Correctness**

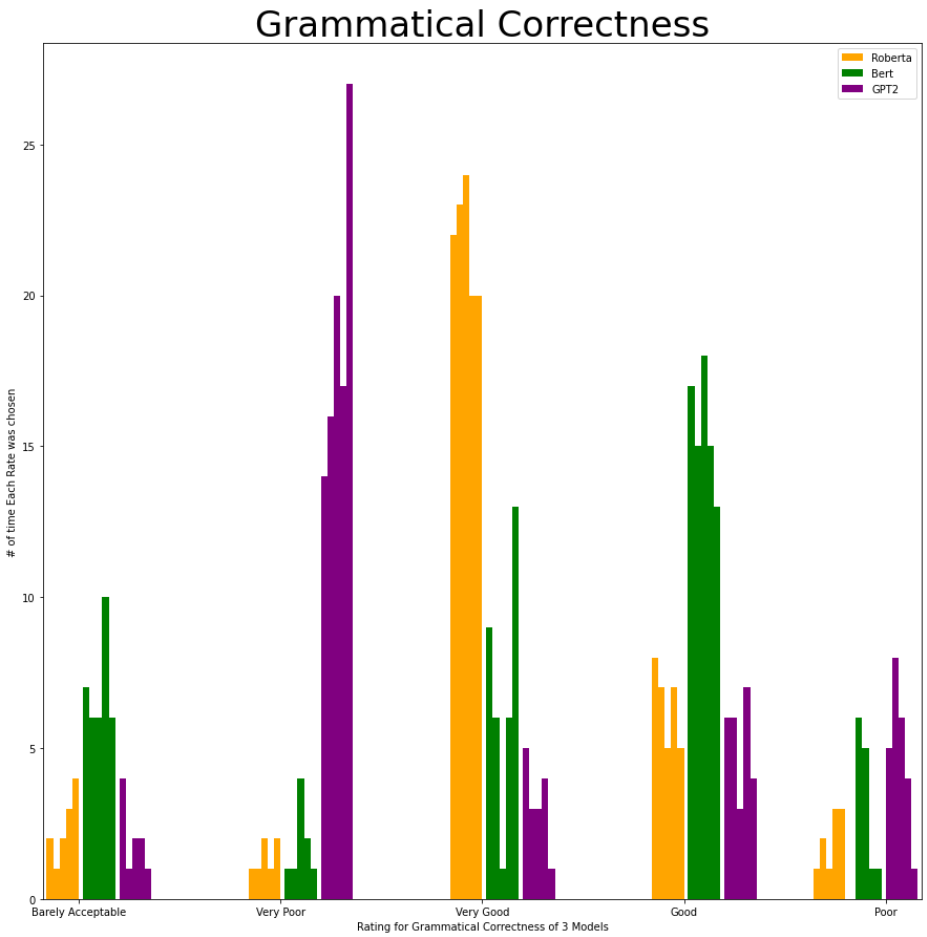


Figure 28 - Human Evaluation Results 1

The first histogram is showing each question's evaluation for the three models, considering the labels above and the bars, it is shown that GPT2 shows the poorest grammatical correctness

out of the three models as approximately 20 participants out of 34 rated the GPT2 results of ‘very poor’ grammatical correctness, while RoBERTa shows the most grammatically correct steganographic texts as between 20 to 25 rates out of 34 agreed that RoBERTa’s stego results are of very good grammar structure. while BERT resulted fairly good grammatically correct steganographic text according to our participants.

2. Context Coherency

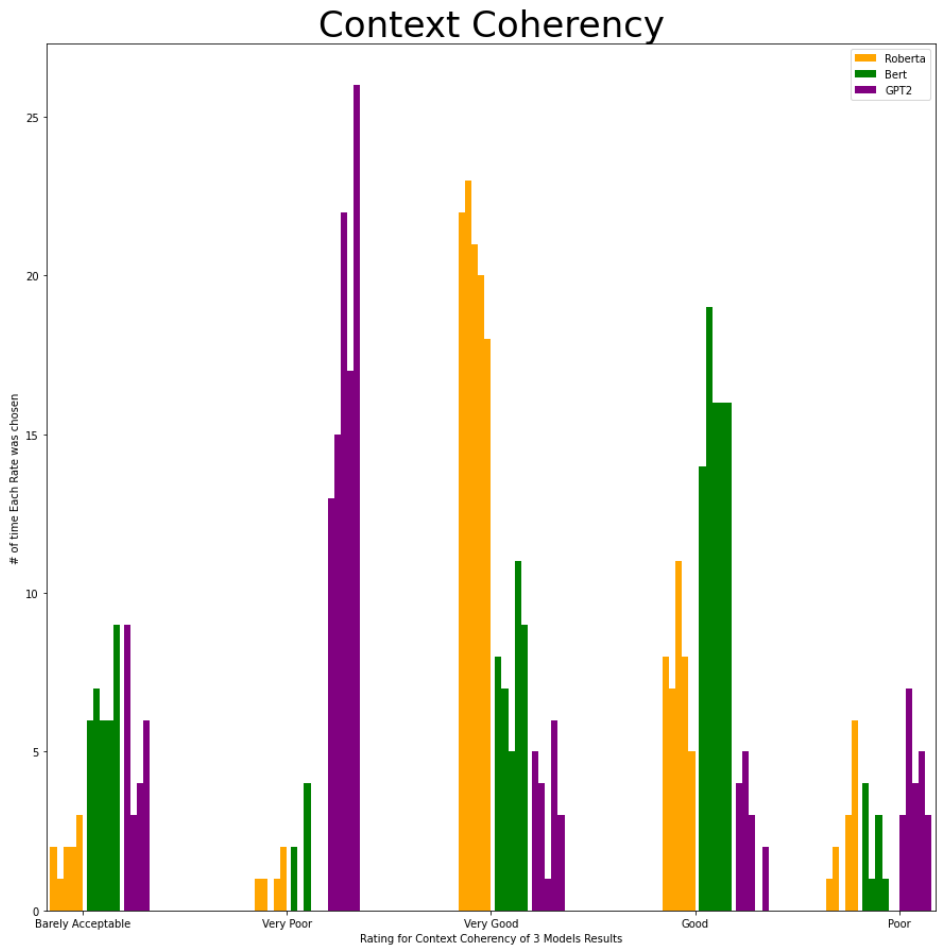


Figure 29 - Human Evaluation Results 2

The second histogram is showing each question’s evaluation for the three models, considering the labels above and the bars, it is shown that GPT2 shows the poorest context coherency out of the three models as approximately between 15 to 25 participants out of 34 rated the GPT2 results of ‘very poor’ context coherency, while RoBERTa shows the most context coherent steganographic texts as between 20 to 25 rates out of 34 agreed that RoBERTa’s stego results are of very good context coherency. while BERT resulted fairly good grammatically correct steganographic text according to our participants.

3. Delivering a correct fact on political domain

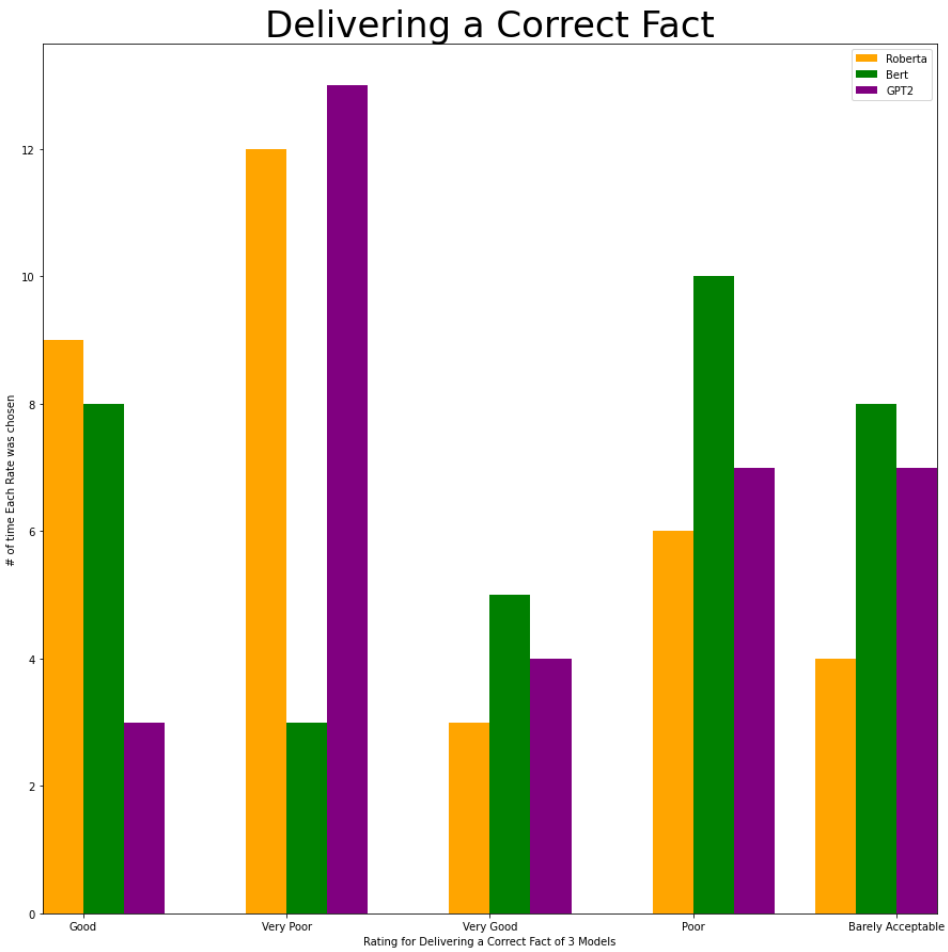


Figure 30 - Human Evaluation Results 3

One of the important factors to consider while testing if the model is well implemented is testing it on critical information to see if it delivers correct factors and can understand the criticalness of this data do not lose or deliver it wrongly leading to suspiciousness and corrupting the meaning of important data. The three models shows false political facts while being used in steganographic purposes which shows that transformers regarding of how powerful and efficient language models they are, they still can not fully understand how critical a data can be to change its context and cause false spreading of news that can not only cause suspicion to the resulted steganographic text but disastrous consequences upon spreading such texts among a large group of people.

5.1.3 Payload Capacity

Besides using Huffman coding in compressing the bit sequence of secret message so to consume the least capacity as possible, An experiment was done with different lengths of secret messages being encoded within different lengths of cover texts on edit-based approaches to get to know how much of payload capacity which is number of bits of secret message is being fully encoded within the steganographic text resulted. Moreover, to evaluate how efficient is each model in encoding the longest secret message within the least payload capacity. Here is the parameters specified for the experiment.

Cover Text	**Size**
Long Cover Text	**~35-40 tokens**
Short Cover text	**~10-15 tokens**

Secret Message	**Content**
Long Secret Message	Meet me at Downtown at 9
Short Secret Message	Help

To be precise, these are the experiments done for each of the three proposed models:

Regarding BERT and RoBERTa

1. Long secret message within long cover text.
2. Long secret message within short cover text.
3. Short secret message within short cover text.
4. Short secret message within long cover text.

Regarding GPT2

1. Long secret message within long specified length of steganographic generated text.
2. Long secret message within short-specified length of steganographic generated text.
3. Short secret message within long specified length of steganographic generated text.
4. Short secret message within short-specified length of steganographic generated text.

Experiments were done considering the same hyperparameters of masking interval = 3 and probability distribution threshold = 0.01. Since GPT2 is not restricted by a cover text and masking specific tokens, so it is not restricted by a specific length to encode a secret message, whatever the length of it is, therefore, **it can encode as long a secret message as required**. However, in edit-based approaches, the capacity load per experiment will be calculated as:

$$\frac{Total\ number\ of\ tokens\ in\ steganographic\ text}{total\ number\ of\ masked\ tokens}$$

In BERT's original paper [18], it was found that the masking interval, which is the masking strategy shared between the sender and receiver, is what decide on how many tokens will be masked within the cover text taking into consideration that BERT roughly masks 15% of cover text tokens as it's the baseline masking language model. Moreover, it was found that when the masking interval increases, the capacity payload of masking decreases but on the other hand the accuracy increases of the resulted steganographic text.

The following tables are the results of the conducted experiments:

Experiment 1:

RoBERTa

30% Masking Capacity Results (**masking interval = 2**) – Most Accurate	
Experiment Type	Results
Long Cover Text, Long Secret Message	**Failed** (30-40% of secret message is encoded)
Long Cover Text, **Short** Secret Message	**Passed** (All secret message is encoded)
Short Cover Text, Long Secret Message	**Failed** (~15% of secret message is encoded)
Short Cover Text, **Short** Secret Message	**Passed** (All secret message is encoded)

BERT

30% Masking Capacity Results (**masking interval = 2**) – Most Accurate	
Experiment Type	Results
Long Cover Text, Long Secret Message	**Failed** (27% of secret message is encoded)
Long Cover Text, **Short** Secret Message	**Passed** (All secret message is encoded)
Short Cover Text, Long Secret Message	**Failed** (12% of secret message is encoded)
Short Cover Text, **Short** Secret Message	**Passed** (All secret message is encoded)

Experiment 2:

RoBERTa

20% Masking Capacity Results (**masking interval = 3**) - Accurate	
Experiment Type	Results
Long Cover Text, Long Secret Message	**Failed** (~28% of secret message is encoded)
Long Cover Text, **Short** Secret Message	**Passed** (All secret message is encoded)
Short Cover Text, Long Secret Message	**Failed** (~14% of secret message is encoded)
Short Cover Text, **Short** Secret Message	**Failed** (~79% of secret message is encoded)

BERT

20% Masking Capacity Results (**masking interval = 3**) - Accurate	
Experiment Type	Results
Long Cover Text, Long Secret Message	**Failed** (24% of secret message is encoded)
Long Cover Text, **Short** Secret Message	**Passed** (All secret message is encoded)
Short Cover Text, Long Secret Message	**Failed** (9-11% of secret message is encode)
Short Cover Text, **Short** Secret Message	**Failed** (67% All secret message is encode)

Experiment 3:

RoBERTa

10% Masking Capacity Results (**masking interval = 4**) – Least Accurate	
Experiment Type	Results
Long Cover Text, Long Secret Message	**Failed** (~24% of secret message is encoded)
Long Cover Text, **Short** Secret Message	**Passed** (All secret message is encoded)
Short Cover Text, Long Secret Message	**Failed** (~13% of secret message is encoded)
Short Cover Text, **Short** Secret Message	**Failed** (~61% of secret message is encoded)

BERT

10% Masking Capacity Results (**masking interval = 4**) – Least Accurate	
Experiment Type	Results
Long Cover Text, Long Secret Message	**Failed** (19% of secret message is encoded)
Long Cover Text, **Short** Secret Message	**Failed** (88% of secret message is encoded)
Short Cover Text, Long Secret Message	**Failed** (8-9% of secret message is encode)
Short Cover Text, **Short** Secret Message	**Failed** (~53% of secret message is encode)

As shown in tables above, the RoBERTa model has showed success in more experiments in comparison with BERT conducted which are hiding the short secret message completely within both long and short cover text but regarding the long secret message, it was shown that it needs a very long cover text to be fully encoded inside it. Regarding BERT results for capacity, it only fully encoded the short also compressed secret message within long cover text which shows the limitations of masking process in addition to very low-capacity payload. Moreover, RoBERTa it is obvious that it can take up to masking approximately 30% to encode the whole secret message although it has the same masking interval as BERT which only mask approximately 15% of cover text within the same masking interval. A study was published

days ago stating that masking up to 40% of input tokens can outperforms the 15% baseline masking process. But sometimes when highly increasing the masking rate can cause corruption to a large portion of cover text resulting in reducing the context size. However, increasing masking rating means more model predictions also more benefit to training as RoBERTa actually do from the beginning [26].

m	Example	PPL (m / 15%)	MNLI	QNLI	SQuAD[3]
80%	We [redacted] high [redacted] models	1141.4 / 43.7	80.8	87.9	86.2
40%	We study high [redacted] rates [redacted] pre- [redacted] models .	69.4 / 20.3	**84.5**	**91.6**	**89.8**
15%	We study high [redacted] ing rates [redacted] pre-training language models .	17.7 / 17.7	84.2	90.9	88.0

Figure 31 - Results of Increasing masking rating on masking language model tasks

This shows the tradeoff relationship between masking intervals and capacity payload, along with the accuracy of steganographic text produced. This means that, the larger the masking interval the less capacity payload but the more accuracy produced. The following figure shows the tradeoff relationships. This shows that edit-based approaches are not a good fit for encoding long secret messages due to their minimal capacity payload in comparison with generation-based approaches. However, RoBERTa's capacity payload is considered high for an edit-based method due to its Dynamic Masking.

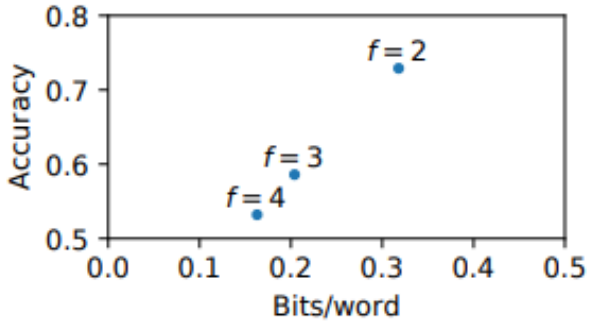


Figure 32 - tradeoff relationship between masking interval f and payload capacity in relevance with accuracy

5.1.4 Tokenization

BERT	RoBERTa
WordPiece Tokenizer	Byte Pairing Encoding
LZ4 is a lossless data compression algorithm that is focused on compression and decompression speed. It belongs to the LZ77 family of byte - oriented compression algorithm.	LINK4 is a stateful data compression algorithm that is focused on compression and decompression techniques. It belongs to the LZ77 family of byte-oriented compression schemes.

One of the important findings throughout this study is the effect of tokenization process in understanding the tokens, tokenize them then mask the cover text. Byte Pairing Encoding is known as a subword-based tokenization algorithm at which it is known for merging the characters till having several subwords that occurs frequently, so it has learnt a lot of subwords to used it for tokenization the input. As shown in the table above, the word ‘compression’ and ‘losseless’ is not sub-word-ly tokenized in BERT, but it is in RoBERTa. This makes the model, in the prediction process, see two tokens holding the meaning of one word so it can be chosen as a [MASK] token in masking process, therefore it would be replaced by a word giving the right context. This shows how the difference in tokenizer can lead to different understanding of the same word thus, different replacements that can give totally different context.

6 Conclusion and Future work

6.1 Summary

Since Transformer Models achieved a great success in predicting context coherent words to their previous sequence, it would be a great approach to be used to generate steganographic text in a linguistic mean. Linguistic Steganography is the act of embedding the words of the secret message within a cover text such as a paragraph that is relative in context such that it seems like normal text to readers' eye yet holds the secret text words safely. In this study, Three Steganography approaches are proposed using deep learning: Generation-based approach (**GPT2)** and Edit-based approach: **BERT** and finally, **RoBERTa**, which will be the focus for this study as it was not used before in steganographic purposes. GPT2 is an unsupervised deep learning transformer-based language generative models taking a textual input as a word/sentence for formulating a single purpose of which is predicting the next word(s) in a sequence of words (sentence). At the end, it has the ability to just take a word as input to generate multiple paragraphs of different context taking into consideration choosing the suitable words that fulfil the correct meaning of sentence and hides the secret message formulating a context coherent paragraph or sequence of words (the steganographic text) using Huffman coding. Another two approaches are proposed which are from the same type, Edit-based Transformers, the BERT and RoBERTa which are of the same Encoder only architecture, replacing the masked self-attention layers with self-attention layers as its bidirectional learning from words at both sides of the target token, this creates large range of relationship understanding between words within same sequence that benefits the steganography process. They are cover-based techniques at which the cover text where the secret message is to be hidden inside is given firstly to the tokenizer which masks a percentage from the cover message at random at which those masked tokens are to be replaced with tokens completing the context of the sentence and at the same time encodes the secret message within it using Huffman coding. The three proposed models process input tokens using a powerful multi-headed self-attention mechanism to capture the internal dependencies of the input sequence in order to generalise a likely continuation of sequence of inputs to be paraphrased into another format of context coherent text. List of candidates is given as output of SoftMax activation function of last block at which the suitable is candidate is chosen using the equivalence of chunks of secret message after being compressed using Huffman coding. Although GPT2 is not restricted by a cover text and can encode inside it as many bits of secret message as required, it's resulted steganographic text in comparison with that of edit-based approach was a lot poorer in grammatical structure and context coherence, so it was motivational to focus on the edit-based models in order to make a fair comparison. Although, both edit-based models are of the same architec-

ture. However, RoBERTa shows better results due to some modifications on how BERT is trained and processes the input. RoBERTa has optimized the training of BERT by using longer training, larger batch sizes and more steps. Additionally, RoBERTa has followed a different masking strategy that helped benefits the prediction process by Dynamic masking showing impressive training process at which it generates a new masking pattern on every epoch instead of same masking pattern during the whole training process. Several experiments were tested in order the know which model out of the three is the best fit for steganographic purpose. A survey was conducted on a group of students to give 5-point scale ratings on resulted stego-texts from the three proposed models to give an evaluation about which model give the most non-suspicious sentences according to **grammatical correctness**, **context coherency** and **delivering a correct fact** for political stego-texts only resulting in showing the RoBERTa is the best in terms of grammatical correctness and context coherency however, none of them is advised to be using the political domain as cover medium due to criticalness of this information and preventing spreading false news. Another finding was discovered which stated how the difference in tokenizer can lead to different understanding of the same word thus, different replacements that can give totally different context. All the results has shown that RoBERTa is considered the best fit in terms of providing context coherent, grammatically correct steganographic text of reasonable capacity payload due to its dynamic masking strategy shown in handling long secret message efficiently.

6.2 Future Work

Deep testing on each of the three transformer models will be implemented using an enhanced larger collected dataset of different sizes of cover texts and secret message including different domains to test its reliability on outputting a correct steganographic text in other cases. Exploration on new steganographic methods can be done on RoBERTa such as Arithmetic coding that enhance the results. More experiments on tokenizers could be done for better understanding for sub-words. A common tokenizer is to be implemented from scratch to be trained so it can be used to tokenize cover texts input for BERT and RoBERTa to see how each model would react to the new tokenizer and which enhance their results, and which does not.

References

[1] M. Agarwal, “Text steganographic approaches: A comparison,” *arXiv.org*, 14-Feb-2013. [Online]. Available: https://arxiv.org/abs/1302.2718. [Accessed: 09-Dec-2021].

[2] “Ah4s: An algorithm of text in text ... - wiley online library.” [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/sec.1752. [Accessed: 09-Dec-2021].

[3] Techopedia, “What is tokenization? - definition from Techopedia,” *Techopedia.com*, 03-Jun-2021. [Online]. Available: https://www.techopedia.com/definition/13698/tokenization. [Accessed: 09-Dec-2021].

[4] "The Transformer Model", *Medium*, 2022. [Online]. Available:

https://towardsdatascience.com/attention-is-all-you-need-e498378552f9. [Accessed: 10- Jun-2022]

[5] J. Alammar, “The illustrated GPT-2 (Visualizing Transformer language models),” *The Illustrated GPT-2 (Visualizing Transformer Language Models) – Jay Alammar – Visualizing machine learning one concept at a time.* [Online]. Available: https://jalammar.github.io/illustrated-gpt2/. [Accessed: 09-Dec-2021].

[6] M. Elizabeth and Mary Elizabeth Mary Elizabeth is passionate about reading, “What is a cipher key?,” *EasyTechJunkie*, 23-Jan-2021. [Online]. Available: https://www.easytechjunkie.com/what-is-a-cipher-key.htm#:~:text=The%20cipher%20key%20is%20the%20information%20needed%20to,wants%20to%20conceal.%20The%20plaintext%20is%20manipulated%20twice. [Accessed: 09-Dec-2021].

[7] “(PDF) new robust and secure alphabet pairing text ...” [Online]. Available: https://www.researchgate.net/publication/315045335_New_robust_and_secure_alphabet_pairing_Text_Steganography_Algorithm. [Accessed: 09-Dec-2021].

[8] Additional informationNotes on contributorsBarnali Gupta Banik Barnali Gupta Banik, “Novel text steganography using natural language processing and part-of-speech tagging,” *Taylor & Francis*. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/03772063.2018.1491807. [Accessed: 09-Dec-2021].

[9] M. Agarwal, “Text steganographic approaches: A comparison,” *arXiv.org*, 14-Feb-2013.

[Online]. Available: https://arxiv.org/abs/1302.2718. [Accessed: 09-Dec-2021].

[10] Ieeexplore.ieee.org. 2022. *Text Steganography in chat*. [online] Available at:

<https://ieeexplore.ieee.org/document/4401716/> [Accessed 10 June 2022].

[11] “Linguistic grammar approach to textual steganography.” [Online]. Available: https://www.researchgate.net/profile/Smriti-Medhi/publication/316562249_Linguistic_Grammar_Approach_to_Textual_Steganography/links/5904791d4585152d2e936b13/Linguistic-Grammar-Approach-to-Textual-Steganography.pdf. [Accessed: 09-Dec-2021].

[12] Fang, T., Jaggi, M. and Argyraki, K., 2022. *Generating Steganographic Text with LSTMs*. [online] arXiv.org. Available at: <https://arxiv.org/abs/1705.10742> [Accessed 10 June 2022].

[13] E. Girling, "Everything GPT-2: 1. architecture overview," *Medium*, 18-Dec-2020. [Online]. Available: https://rowlando13.medium.com/everything-gpt-2-1-architecture-overview-132d16fe985a. [Accessed: 09-Dec-2021].

[14] "The annotated GPT-2," *Committed towards better future*, 18-Feb-2020. [Online]. Available: https://amaarora.github.io/2020/02/18/annotatedGPT2.html. [Accessed: 09-Dec-2021].

[15] Medium. 2022. *Masked-Language Modelling With BERT*. [online] Available at: <https://towardsdatascience.com/masked-language-modelling-with-bert-7d49793e5d2c> [Accessed 10 June 2022].

[16] Medium. 2022. *Understanding BERT Transformer: Attention isn't all you need*. [online] Available at: <https://medium.com/synapse-dev/understanding-bert-transformer-attention-isnt-all-you-need-5839ebd396db> [Accessed 10 June 2022].

[17] Medium. 2022. *How to use BERT from the Hugging Face transformer library*. [online] Available at: <https://towardsdatascience.com/how-to-use-bert-from-the-hugging-face-transformer-library-d373a22b0209> [Accessed 10 June 2022].

[18] Devlin, J., Chang, M., Lee, K. and Toutanova, K., 2022. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. [online] arXiv.org. Available at: <https://arxiv.org/abs/1810.04805> [Accessed 10 June 2022].

[19] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V., 2022. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. [online] arXiv.org. Available at: <https://arxiv.org/abs/1907.11692> [Accessed 10 June 2022].

[20] Medium. 2022. *Robustly optimized BERT Pretraining Approaches*. [online] Available at: <https://towardsdatascience.com/robustly-optimized-bert-pretraining-approaches-537dc66522dd> [Accessed 10 June 2022].

[21] Medium. 2022. *Exploring BERT variants (Part 1): ALBERT, RoBERTa, ELECTRA*. [online] Available at: <https://towardsdatascience.com/exploring-bert-variants-albert-roberta-electra-642dfe51bc23#:~:text=The%20key%20points%20of%20difference,of%20the%20sentences%20are%20masked.> [Accessed 10 June 2022].

[22] Analytics Vidhya. 2022. *PyTorch-Transformers with Python Implementation*. [online] Available at: <https://www.analyticsvidhya.com/blog/2019/07/pytorch-transformers-nlp-python/> [Accessed 10 June 2022].

[23] GeeksforGeeks. 2022. *StringIO Module in Python - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/stringio-module-in-python/> [Accessed 10 June 2022].

[24] Pytorch.org. 2022. *PyTorch: Tensors — PyTorch Tutorials 1.11.0+cu102 documentation*. [online] Available at: <https://pytorch.org/tutorials/beginner/examples_tensor/polynomial_tensor.html#:~:text=PyTorch%3A%20Tensors,-A%20third%20order&text=A%20PyTorch%20Tensor%20is%20basically,used%20for%20arbitrary%20numeric%20computation> [Accessed 10 June 2022].

[25] Web.mit.edu. 2022. [online] Available at: <http://web.mit.edu/6.02/www/s2010/handouts/labs/lab10.shtml> [Accessed 10 June 2022].

[26] Wettig, A., Gao, T., Zhong, Z. and Chen, D., 2022. *Should You Mask 15% in Masked Language Modeling?*. [online] arXiv.org. Available at: <https://arxiv.org/abs/2202.08005> [Accessed 10 June 2022].

[27] Ueoka, H., Murawaki, Y. and Kurohashi, S., 2022. *Frustratingly Easy Edit-based Linguistic Steganography with a Masked Language Model*. [online] arXiv.org. Available at: <https://arxiv.org/abs/2104.09833> [Accessed 10 June 2022].

[28] "Byte-Pair Encoding: Subword-based tokenization algorithm", *Medium*, 2022. [Online]. Available: https://towardsdatascience.com/byte-pair-encoding-subword-based-tokenization-algorithm-77828a70bee0. [Accessed: 12- Jun- 2022].

[29] B. Zain, *Cs.uwaterloo.ca*, 2022. [Online]. Available: https://cs.uwaterloo.ca/~mli/Bin.pptx. [Accessed: 12- Jun- 2022].

[30] "Deconstructing BERT, Part 2: Visualizing the Inner Workings of Attention", *Medium*, 2022. [Online]. Available: https://towardsdatascience.com/deconstructing-bert-part-2-visualizing-the-inner-workings-of-attention-60a16d86b5c1. [Accessed: 12- Jun- 2022].

[31] "Summary of the tokenizers", *Huggingface.co*, 2022. [Online]. Available: https://huggingface.co/docs/transformers/tokenizer_summary. [Accessed: 12- Jun- 2022].

Appendix

Human Evaluation Survey

Text Steganography Using Transformer Models Human Evaluation

rate these sentences upon the following categories: grammatical correctness, Context Coherency and delivering a correct fact

Name

First Name

Last Name

1r. The Fund praised the protectionists to support economic security in order to confront possible economic repercussions from the Russian-Ukrainian sanctions, asserting its unconditional support for Armenia to complete the economic reform project. *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	◯	◯	◯	◯	◯
Grammatical Correctness	◯	◯	◯	◯	◯
Delivering a correct fact	◯	◯	◯	◯	◯

2b. Tourism partner, Qatar Airways began cooperation in attracting additional tourist traffic to Bahrain. *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	◯	◯	◯	◯	◯
Grammatical Correctness	◯	◯	◯	◯	◯
Delivering a correct fact	◯	◯	◯	◯	◯

3g. Al-Sisi had told reporters this past weekend in Ankara they would only seek revenge. *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	◯	◯	◯	◯	◯
Grammatical Correctness	◯	◯	◯	◯	◯
Delivering a correct fact	◯	◯	◯	◯	◯

4b. Earlier this morning, I visited the library at the Philippine National University with my uncle so he could borrow some materials for an essay on Chinese Literature. Wandering past shelf after shelf, he asked me, " How does it feel to be read another book that no - one will read? *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	◯	◯	◯	◯	◯
Grammatical Correctness	◯	◯	◯	◯	◯

5r. When several South African universities joined, it was expected this would improve access and support. It had the opposite effect. *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	◯	◯	◯	◯	◯
Grammatical Correctness	◯	◯	◯	◯	◯

6g. Campaigners for Life.There are still questions about the future of it. *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	◯	◯	◯	◯	◯
Grammatical Correctness	◯	◯	◯	◯	◯

7r. Don't you think you should buy her something a little more traditional, like jewelry and maybe a travel bag? *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	◯	◯	◯	◯	◯
Grammatical Correctness	◯	◯	◯	◯	◯

8b. I have been looking for my cash in every inch i owned and sat on yesterday, either in the street or at home, I used all the my debit cards. After returning home with no money, i was making bread and found it in a very weird box, under the stove. *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	◯	◯	◯	◯	◯
Grammatical Correctness	◯	◯	◯	◯	◯

9g. Anniversary Special!We have a little girl who is a little girl. *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	◯	◯	◯	◯	◯
Grammatical Correctness	◯	◯	◯	◯	◯

10r. With large levels of infection driven by the R. 1. 1. 529 (omicron) variant of severe systemic respiratory syndrome coronavirus 2 (SARS - CoV - 2), showing evidence of waning immunity after the booster stages of coronavirus disease 2019 (Covid - 19) vaccine. *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	◯	◯	◯	◯	◯
Grammatical Correctness	◯	◯	◯	◯	◯

11b. We evaluated the covid relative effectiveness of a fourth daily dose as compared with that of a third injection given at least 4 months earlier in persons 60 years of age or younger. *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	◯	◯	◯	◯	◯
Grammatical Correctness	◯	◯	◯	◯	◯

12g. respiratory disorders : Clinical aspects , and Pulletie-Fazza. The American Journal of Clinical Nutrition.. *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	◯	◯	◯	◯	◯
Grammatical Correctness	◯	◯	◯	◯	◯

13g. staff in this respect: a small, very small, very small, very small, very small, very small, very small,. *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	○	○	○	○	○
Grammatical Correctness	○	○	○	○	○

14b. A new article published in the journal Research in Behavioral Neuroscience shows that older people tend to release more oxytocin in response to social situations that arouse empathy. A lower oxytocin response was also correlated with greater levels of complex behaviors and increased involvement with life. *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	○	○	○	○	○
Grammatical Correctness	○	○	○	○	○

15r. After skin cancers, prostate cancer became the most Common Trusted skin cancer among Americans in the United States.' *

	Very Poor	Poor	Barely Acceptable	Good	Very Good
Context Coherency (logical or consistent structure and meaning)	○	○	○	○	○
Grammatical Correctness	○	○	○	○	○

Submit