Received 1 January 2024, accepted 12 January 2024, date of publication 15 January 2024, date of current version 30 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3354710

RESEARCH ARTICLE

A Character Based Steganography Using Masked Language Modeling

EMİR ÖZTÜRK, ANDAÇ ŞAHİN MESUT, AND ÖZLEM AYDIN FİDAN
Computer Engineering Department, Trakya University, 22030 Edirne, Turkey
Corresponding author: Emir Öztürk (emirozturk@trakya.edu.tr)

ABSTRACT In this study, a steganography method based on BERT transformer model is proposed for hiding text data in cover text. The aim is to hide information by replacing specific words within the text using BERT's masked language modeling (MLM) feature. In this study, two models, fine-tuned for English and Turkish, are utilized to perform steganography on texts belonging to these languages. Furthermore, the proposed method can work with any transformer model that supports masked language modeling. While traditionally the hidden information in text is often limited, the proposed method allows for a significant amount of data to be hidden in the text without distorting its meaning. In this study, the proposed method is tested by hiding stego texts of varying lengths in cover text of different lengths in two different language scenarios. The test results are analyzed in terms of perplexity, KL divergence and semantic similarity. Upon examining the results, the proposed method has achieved the best results compared to other methods found in the literature, with KL divergence of 7.93 and semantic similarity of 0.99. It can be observed that the proposed method has low detectability and demonstrates success in the data hiding process.

INDEX TERMS BERT, masked language modeling, steganography.

I. INTRODUCTION

Information hiding is crucial for the security of computer systems. The communication between sender and receiver should not be accessible to a third party. For this purpose, encryption and information hiding methods can be employed. While encryption protects information, it is vulnerable to attacks since it reveals the presence of the information [1], [2]. The goal of steganography is to ensure that the hidden information remains unnoticed by a third party [3]. Steganography is a significant subfield within the field of information hiding [4].

The environment where the data is hidden is referred to as the cover media [5]. The cover media should be one of the known and commonly used media [6]. Data can be hidden on various types of cover media, such as text, images, audio, and videos, as well as on geospatial data [7] or visible wavelengths [8]. While videos and images can accommodate a larger amount of hidden data due to their higher storage capacities, the same cannot be said for text. Text-based steganography typically has more limited capacity to hide data compared to media files like images and videos [9]. Consequently, due to the lower probability of data being hidden on text, performing information hiding on text aligns better with the purpose of steganography. Furthermore, unlike other media, text possesses the robustness to facilitate data transmission without undergoing distortion during transmission.

Many traditional text steganography methods perform the hiding process by altering the format of the text [10]. Format alterations include modifications in letter spacing, line breaks, font characteristics, and invisible characters. In [11], the size of the spaces between words in text images has been modified, enabling a hiding process. The hiding operation is executed by adding spaces between words, which are not visibly discernible. However, when texts are aligned and separated by lines, differences in spacing can be analyzed to extract hidden information. Similarly, in [12], watermark bits are hidden by altering the spacing between words and paragraphs in the context of watermarking. This approach involves modifying the gaps between words and paragraphs to embed the watermark information effectively. In [13] a

The associate editor coordinating the review of this manuscript and approving it for publication was Maria Chiara Caschera.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

font attribute called character spacing is used to embed the secret. Finally, in [14], space characters from different fonts are employed as individual encoding schemes. The study highlights that using space characters from different fonts is a secure hiding method since the font types do not introduce noticeable changes perceptible to the human eye.

In recent years, introduced steganography applications achieve hiding through symbol and syntax alterations on the document [15].

Some studies opt for using characters and words as symbols instead of spaces for hiding bits. Thus, since there are no structural changes made within the text, an observer would not perceive that data is hidden unless they possess the original text. Only someone who knows the algorithm for word replacement would be able to extract the information.

In [16], a hiding process has been conducted by replacing words in a predetermined fixed text with altered words that do not distort the meaning. Four words have been selected at each replaceable point, allowing for the storage of four values for each word without compromising the meaning. In [17], the hiding process takes place on words during the translation process. Hiding operation is carried out by selecting words across multiple translation choices.

At the core of the word substitution process lies the replacement of specific words selected within the cover text with the intention of concealing information and conveying it to the recipient. In this method, various features of the substituted word, such as its list order, letters, length, etc., can be utilized.

In the process of word alteration, there are works that perform information hiding by substituting similar words based on Synonym Substitution [18], [19], [20], [21], [22], [23]. Some studies employ architectures like WordNet to assist in finding synonyms [24]. The disadvantage of these methods is that the limited number of synonyms restricts the potential alterations that can be made to the text, thereby limiting the amount of information that can be hidden.

With the rise in popularity of deep learning algorithms, many steganography methods utilizing generative models such as Recurrent neural network (RNN) and Long short-term memory (LSTM) have been developed [10], [25], [26]. Additionally, there are numerous generative steganography methods employing Generative Adversarial Networks (GAN) [27] and language models [28], [29]. The adoption of these techniques has been shown to increase the rate at which information can be hidden in comparison to classical text steganography methods.

Traditional text steganography methods require manual preparation of the cover text. Moreover, to prevent the exposure of data, the frequency of changes in locations that will be altered should be kept as low as possible. Additionally, hiding processes based on spaces or word substitutions often occur at the bit level. This implies that the cover text, which needs to be prepared based on the amount of the data to be hidden, must be considerably large. As the size of the data increases, manually preparing such a cover text becomes practically impossible. For example, in a scenario where 2 bits are stored in each word and hiding is performed every 5 words, a text of 20 words is required to store only 1 byte. For hiding a message of 50 characters, a text of 1000 words needs to be prepared. One kilobyte of data consists of 1024 characters, making manual text preparation quite challenging. In generative models, there is no limit to the words that can be generated. Text can be rapidly generated in the desired number of words based on the desired message size. Moreover, unlike manually prepared methods, sentences in the desired format can also be generated. With these features, generative models offer much greater capacity compared to manually prepared cover text.

In methods where words are replaced, in manually prepared sentences, determining the words to replace the target word requires manual effort. This process also has its limits. However, deep learning models can suggest hundreds of words at once in place of a desired word without compromising the overall meaning. This allows for an increase in hiding capacity by multiplying the number of different words, thereby increasing the number of bits hidden per word.

Generative methods aim to perform information hiding on the text they generate, generating text and then embedding data within it [30]. The disadvantage of this approach is that hiding information on the generated text can be statistically detectable [31]. The generated text should closely resemble natural text and should not give away the presence of hidden information. The goal of these studies is to prevent the detectability of artificially generated data's resemblance to natural text. Consequently, in this study, the intention is to perform information hiding through changes made to pre-existing written data. This approach aims to make the detection of hidden data more challenging by modifying a text that has already been naturally written without changing its meaning.

Transformer models are used in various fields like classification, masked language modeling, text generation, and question answering [32]. Numerous steganography studies utilize this transformer architecture. Among these, there are methods employing semantic and syntactic features for hiding, as well as those using embedding vectors during translation [1].

Transformers are bidirectional models. A bidirectional model is neural network architecture that processes input data in both directions. With this, the model can learn from information with past and future contexts. The success of predictions is enhanced by learning this context bidirectionally. In [33], it is mentioned that BERT, a transformer model with masked language modeling support, outperforms other unidirectional methods due to its bidirectionality.

Particularly with masked language modeling, suggestions can be provided by the model for a masked word using [MASK] tag within a given sentence. By utilizing the predicting replacements, data hiding can be carried out. Numerous steganography applications are based on the BERT (Bidirectional Encoder Representations from Transformers) which is developed by Google [34] with some of them using text generation and others focusing on the word alteration process like highlighted in this study [31].

In the process of hiding text, selecting a larger amount of cover text is advantageous for achieving higher security, as the hidden data will be distributed across different areas of this text. Therefore, to maximize confidentiality, it is necessary for the cover text to be as large as possible, while keeping the message itself short. Minimizing the proportion of hidden data in the overall text is crucial for achieving optimal hiding operation.

In this study, two distinct BERT models are fine-tuned for both Turkish and English languages. These BERT models are then used for steganography utilizing the masked language modeling function. In the next section, the proposed method's stages of information hiding and retrieval are explained. In the third section, information about the model's training, the datasets used, and experimental results is provided. In the final section, conclusions are discussed.

II. PROPOSED MODEL

Models used for masked language modeling are capable of predicting a desired number of words and it can be used to replace a masked word within a given sentence. The process in proposed method involves using the letters within the words obtained through this prediction mechanism to perform the hiding process. In steganography applications, the method consists of two stages: hiding and retrieval. In this study, two hiding mechanisms are employed – one for hiding the secret message and another for hiding the necessary parameters, named as headers for the rest of the article, to obtain this text. The details of these mechanisms are provided respectively.

In the process of hiding the secret message within the cover text, a word is used for each character. These characters are replaced with characters from the words predicted by the model. The alteration of the selected characters' positions is essential to complicate the retrieval of information. A *loopIndex* value is stored to handle this position-shifting process. Additionally, to select the words for hiding in cover media from different indexes, a hash function is determined and employed. The hash function is selected as squarehash for the implementation for obtaining experimental results. Additionally, to achieve the random order of selecting words within the text, any desired hash function can be used. For the inclusion of the possibility of chaning any word in text, the maximum value of the hash function should be the count of the words in the cover text. For this function to produce different values with each run, a *seed* value must be specified. As this *seed* value is also required during retrieval, it is one of the variables that need to be stored initially. An example process of calculating loopIndex and hash values are given in Figure 1.

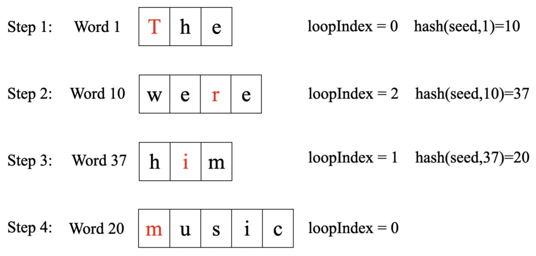

Step 1: Word 1 T h e loopIndex = 0 hash(seed,1)=10
Step 2: Word 10 w e r e loopIndex = 2 hash(seed,10)=37
Step 3: Word 37 h i m loopIndex = 1 hash(seed,37)=20
Step 4: Word 20 m u s i c loopIndex = 0

FIGURE 1. Stages of obtaining wordIndexes and loopIndex values.

In Figure 1, an example is provided where the word "Trim" will be hidden to demonstrate where the specified variables are used. In this example, it is assumed that predictions from the BERT model are obtained and words are replaced. In the first stage, the loopIndex is initialized as 0, and the word index as 1. Although the word index is given as 1 in the example, in the case where header information is included, this index will have larger values. Since the loopIndex value is 0, a word with "T" at index 0 is desired, and "The" is found. After this stage, the loopIndex value is incremented and modulo with the maximum length value is applied. For this example, this value is chosen as 3. If a larger value is selected, longer words will be needed. For the success of the hiding process, this value can be adjusted as desired. To find the next word index, the current index and seed value are given to the hash function, resulting in the next word index. In the example, these values are obtained as 10, 37 and 20, respectively. It's important to note that the hash function does not always have to provide indices in an ascending order, making it more challenging to detect the hidden data. The alteration of word and letter order through the use of loopIndex and hash values is performed to complicate the retrieval of data, aiming to enhance the difficulty in obtaining the information.

The seed value only affects the value generation of the hash function. Therefore, any desired integer value can be chosen as seed value. In the experiments, a random seed value has also been selected, and the same seed value has been used for obtaining all the results. Lastly, since the number of characters to be obtained needs to be known, a *charCount* variable needs to be initially stored. A bit-based method similar to the mechanism in [16] is employed to store these three values.

To store the values, firstly, the values to be hidden are converted into a bit sequence and then divided into blocks of a predetermined size. Each block needs to be hidden over a word. In the hiding process, a transformer model is provided with a specific number of words, and the desired word to be predicted is replaced with the [MASK] tag. In transformers, a specified number of predictions for the [MASK] can be made based on the words in the sentence and they can be provided before or after the masked word. Therefore, the number of words given before and after the mask is referred to as the *windowSize*. As the transformer model will make predictions for the masked word within a given window relative to other words, performing only one masking operation within one window is essential for obtaining hidden data. Thus, the data hiding process starts from the first index and advances by a full window size at each step to mask words at these indices, feed them to the transformer model, and demand predictions that will store the bit sequence. This value will be 2^n for storing n bits. Once the predictions are obtained, the word at the index that corresponds to the value of 2^n in the generated list will replace the masked word. The steps of the creation of the header are outlined in Figure 2.

In Figure 2, an example is provided for storing the first block of the given value 513 within the text. According to this example, the value 513 is converted to binary as 001000000001. Successively, the values 001, 000, 000, 001 should be stored in the text. To store the value 001 (1) in the first block, starting from the first index in the text, the [MASK] label is assigned instead of the word "I" in the text. Subsequently, a prediction is requested from the BERT model. Among BERT's predictions, the word "you", corresponding to the value 001 or 1, is selected, and the word "I" is replaced with this word. This process is carried out for all remaining blocks. After each masking operation, the window will be advanced. This process is repeated sequentially for *charCount*, *seed*, and *loopIndex* values.

In the ASCII table, there are a total of 62 characters for letters and numbers, with 10 between 48 and 57, 26 between 65 and 90, and 26 between 97 and 122. If any of the obtained binary values does not fall within these ranges, the masking process cannot be performed as the transformer model generates words in natural language. Therefore parameter values from the first stage are not hidden just as the letter-hiding method. In some cases, byte values that will not appear in any word might be obtained. In such cases, hiding would not be possible, hence a different approach is taken in the first stage.

In the second stage, after the *loopIndex*, *charCount* and *seed* parameters are hidden, the secret message is divided into letters. A hiding operation is performed for each letter, using the *loopIndex* value to determine the index within the word where the hidden letter will be placed. For each letter the index containing the word to be masked is obtained using the hash function. Hash function should generate indexes with non-overlapping windows. The reason for this is explained in the previous hiding stage. Once this index is obtained, the word is masked, and the transformer model is tasked with making 257 predictions. This value is chosen as 1+256, with the first word being used for a specific case named *skip* element. The reason for choosing 256 is to increase the likelihood of obtaining the required word for the method. The method has been designed to allow flexibility in changing this value as needed.

After making 257 predictions for the word, the process continues by seeking a word containing the letter to be hidden at the *loopIndex* value. If this word is not found within the prediction list obtained, the word at the *skip* index, determined as 0th index, is used to indicate that data cannot be hidden here. Subsequently, in the next step, a new index value is determined using a seed and the hiding process continues. The steps of the data hiding process are given in Figure 3.

In Figure 3, an example is provided for hiding the first letter of the word "secret" in the second index. In this example, the loopIndex value is initialized from 0. At this stage, after creating a window containing the 2nd index value, the word "I" at this index is replaced with [MASK], and predictions are obtained from the BERT model. Among the predicted words, it is checked whether there is a word with the letter "s" at the loopIndex index. At this point, in the 4th prediction the word "she" is found, and the letter at loopIndex (0) is "s". Therefore the word "I" is replaced with "she". If this condition was not met, the word "they" at the 0th index is replaced with "I", and the index is changed with hash function to hide the letter in the next position. After this operation, the loopIndex value is updated. The pseudocode for the information hiding stage is provided in Algorithm 1 given below.

Algorithm 1 Information Hiding Stage

```
predictionSize = 128
Get seedPackets, wordCountPackets, and loopIndexPackets from integer values
Words = split(coverMedia," ")
for i = 0 to len(packets) do
    getPacket($p$)
    Words[index] = "[MASK]"
    predictions = predict(Words[index : index + halfWindowSize])
    Words[index] = predictions[$p$]
end for
while counter $<$ wordCount do
    index = hash(seed)
    Words[index] = "[MASK]"
    predictions = predict(Words[index − halfWindowSize : index + halfWindowSize])
    skip= predictions[0]
    wordList = getWords(loopIndex,predictions)
    if len(wordList) $>$ 0 then
        words[index] = wordList[0]
    else
        words[index] = skip
    end if
end while
return words as text
```

In the process of obtaining the information, firstly, the *seed*, *loopIndex*, and *charCount* values hidden in the first stage are obtained.

The process begins with the first index. The word at the obtained index is replaced with the [MASK] tag and is then fed into the transformer model, including words before and after it within a window of determined size. The index at which the word in the prediction list matches the given word

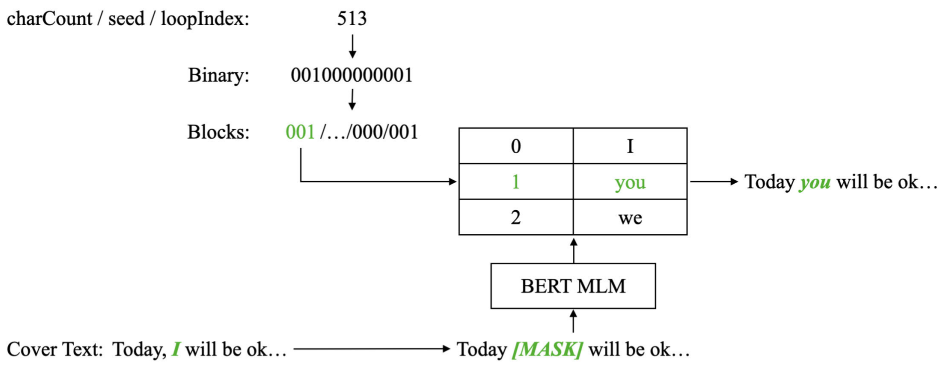


FIGURE 2. Stages of hiding header information in the cover text.

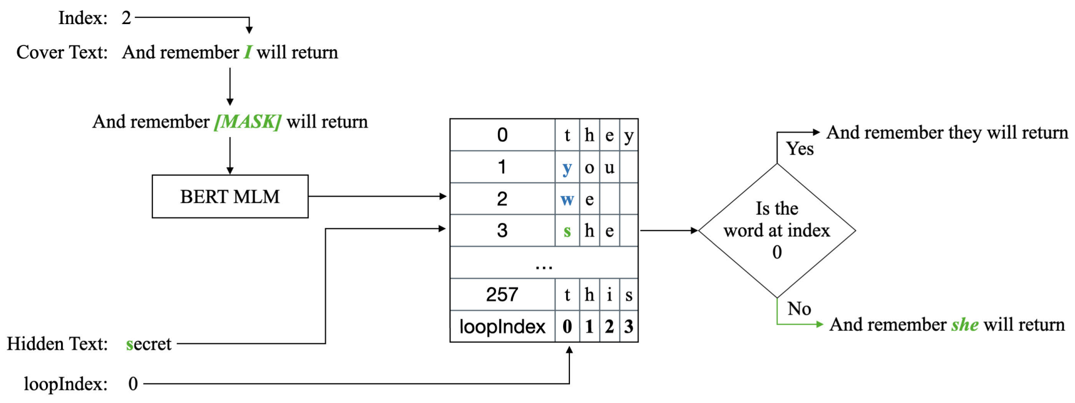


FIGURE 3. Hiding information in the cover text stage.

is identified, and the binary value of this index is obtained. This process is carried out for all variables within all blocks, resulting in the retrieval of integer values for the header variables. The process of obtaining these values is given in Figure 4.

In Figure 4, an example is given for obtaining the first block of the value 513 stored in Figure 2. A portion of the hidden text, containing the first index and equal to the window size, is taken, and after changing the word at first index with [MASK], it is given to the BERT model. Subsequently, predictions are requested from the BERT model. Since the BERT model will produce the same results for the same window, the index corresponding to the hidden word "you" is obtained, which is 1. This index is converted to binary (001) with the block size 3. This process is repeated for all blocks, and these blocks are combined to obtain the binary value 001000000001 (513). Subsequently, using these obtained values, the process of retrieving the hidden letters is carried out.

For retrieving hidden letters, the obtained *seed* value is used, and the process of obtaining the indices is carried out sequentially until the amount of data specified by *charCount* is obtained. The words at these indices are fed into the transformer model with a window, as in the previous stage. If the word at the index is equal to the 0th word predicted by the model, this index is skipped. The next index is obtained using the *seed* value. If the read word is different from the 0th word, the letter corresponding to the *loopIndex* value in this word is read, and the hidden information is obtained in this manner. Subsequently, the *loopIndex* value is updated. This process continues until *charCount* is obtained. The steps of this process are provided in Figure 5.

Figure 5 illustrates the retrieval of the hidden letter "s" example in Figure 3. The window obtained using the index value from the hash function, is given to the BERT model, and predictions are requested. Here, the word at the index is present in the 4th index of the BERT model's predictions, therefore the letter "s" at the loopIndex value is written to the output stream. In the case where the searched word is at index 0, this word will be skipped, and the hash function will be used to move to the next index. After this stage, the calculation of the next loopIndex value is also performed. The pseudocode for the information retrieval stage is provided in Algorithm 2.

The hiding and retrieving process of header and secret message onto the text is presented in as an overall schematic diagram in Figure 6.

In Figure 6, the overall processes of hiding and retrieval are explained. In the first phase of hiding, three parameters needed are hidden in words starting from the first index with

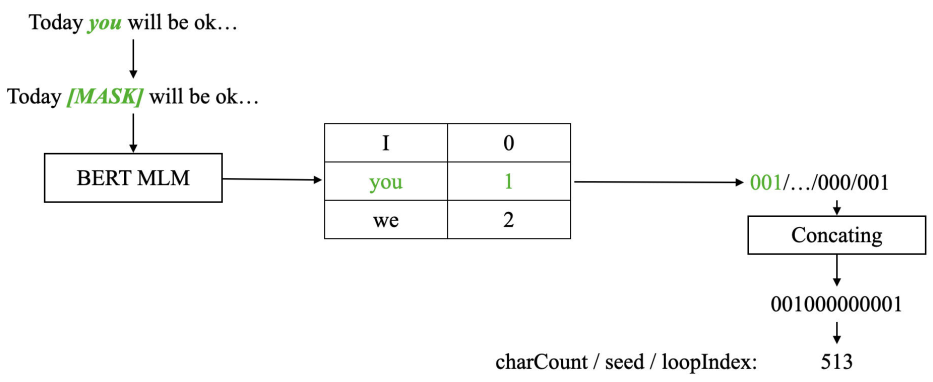


FIGURE 4. Retrieving header information from the stego text.

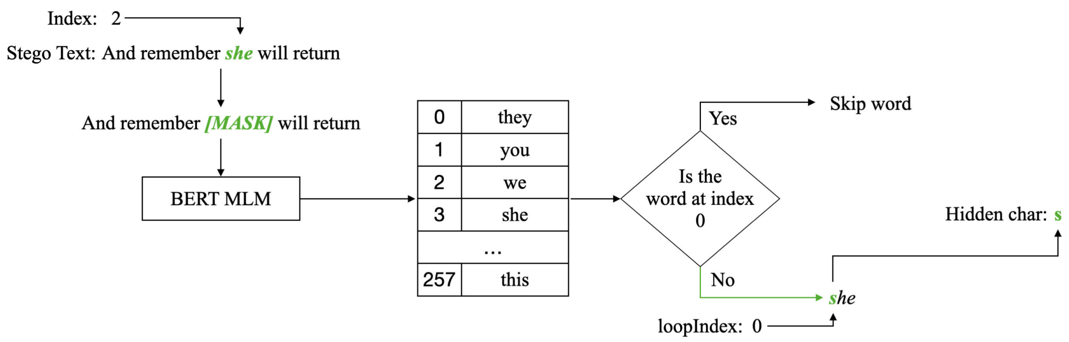


FIGURE 5. Retrieving hidden information from the stego text.

Algorithm 2 Retrieval of Hidden Text
```
packetList = []
for $i = 0$ to len(packets) do
    word = Words[index]
    Words[index] = “[MASK]”
    predictions = predict(Words[index : index + halfWindowSize])
    wordIndex = predictions[word]
    packetList += toBinary(wordIndex)
end for
outputText = ""
while counter < wordCount do
    index = hash(seed)
    word = Words[$index$]
    Words[index] = “[MASK]”
    predictions = predict(Words[index − halfWindowSize : index + halfWindowSize])
    if $W \neq$ predictions[256] then
        outputText+ = word[loopIndex]
    end if
    Update loopIndex
end while
return outputText
```

a specified window size. The words overlaid here are used for the information hiding process, and the suggestions from the BERT model for the values to be hidden are written in place of the words corresponding to the desired index. For the sake of simplifying the diagram, numerical values have been selected to be storable in a single word. If the values to be hidden exceed the block size, multiple words are used for each value. In the second stage, the letters of the hidden text are successively hidden in the words at the 4th, 7th, and 2nd indices obtained from the hash function. At this stage, the words underlined are replaced with the words suggested by the BERT model where the desired letter is found in the loopIndex value. In the third stage, to obtain data from the created stego text, the words are read in the same order to retrieve the header information. In the final stage, index values are obtained through a hash function until the number of hidden characters is reached. At each step, it is checked whether the word at the index has a *skip*. If not, the letter in the calculated loopIndex value is obtained to obtain the hidden message.

For the method to work successfully, the BERT model used for masked language model should have a high prediction accuracy. Therefore, in the method, two distinct models are fine-tuned and utilized for masking for Turkish and English languages. These models are named “dbmdz/bert-base-turkish-cased” and “bert-base-cased”, respectively. For the Turkish model, sentences from the Sabah newspaper in the SUDER [35] corpus were used during the fine-tuning process. For the English model, the English50mb file from the PizzaChili corpus [36] is employed. The loss graphs obtained

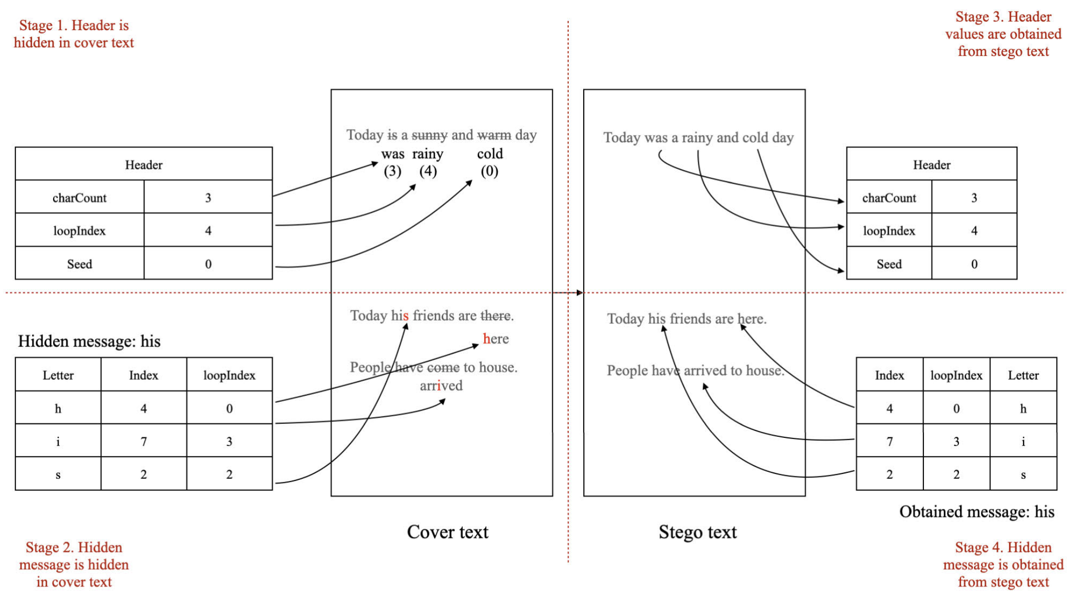

Stage 1. Header is hidden in cover text
Header
charCount 3
loopIndex 4
Seed 0
Hidden message: his
Letter Index loopIndex
h 4 0
i 7 3
s 2 2
Stage 2. Hidden message is hidden in cover text
Today ~~is~~ a ~~sunny~~ and ~~warm~~ day
was (3) rainy (4) cold (0)
Today his friends are ~~there~~.
here
People ~~have~~ ~~come~~ to house.
arrived
Cover text
Today was a rainy and cold day
Today his friends are here.
People have arrived to house.
Stego text
Stage 3. Header values are obtained from stego text
Index loopIndex Letter
4 0 h
7 3 i
2 2 s
Obtained message: his
Stage 4. Hidden message is obtained from stego text

FIGURE 6. Hiding and retrieving data from text.

during the fine-tuning process are presented in Figure 7 for English (a) and Turkish (b).

Both models are trained for 100 epochs. The selected number of epochs aimed to achieve comparable and non-overfitting training and validation loss values for both models. In training, it is observed that the validation loss begins to stabilize after 100 epochs. Beyond this point, continuing the training would result in the training loss decreasing while the validation loss remains constant. Since this indicates the model may be overfitting, training is halted at this stage.

III. EXPERIMENTAL RESULTS

To obtain results, different cover texts and secret messages needed to be prepared. For obtaining cover texts, a plain text output containing Cumhuriyet newspaper articles from the SUDER corpus is obtained for Turkish language. For English texts, the BookCorpus [37] is used. The SUDER corpus comprises 2.5 million distinct news texts in the Turkish language. The BookCorpus encompasses 74 million lines of sentences in the English language. Both datasets are acquired in JSON format and underwent preprocessing to transform them into raw text. To preserve the anonymity of selected data samples, segments of predetermined sizes are randomly chosen from these sentences. To enable experimentation with different scenarios, 10 pieces of 50, 100, and 250Kb sizes are prepared. This choice is made to align with real-world scenarios. Preparing 10 instances of each size aimed to demonstrate the behavior of measurement metrics in various scenarios.

The length of the hidden message is determined as 25%, 50% and 100% of the embedding capacity of cover texts. To obtain the total length of bytes for hidden messages, embedding capacities of cover texts are calculated.

To determine the embedding capacity, it is necessary to delve into how the method works. The method performs hiding on words, storing one character per word. The number of words in the method is calculated by dividing the text based on spaces, excluding punctuation WC_{words}. The method's capacity in terms of characters hidden (*charCount*), the *seed* determining where words will be hidden, and the *loopIndex* value indicating the extent of character reordering within words are hidden as 32-bit integers. Given that the number of bits in each hidden block is denoted as n, the required number of blocks for each of these values is calculated as:

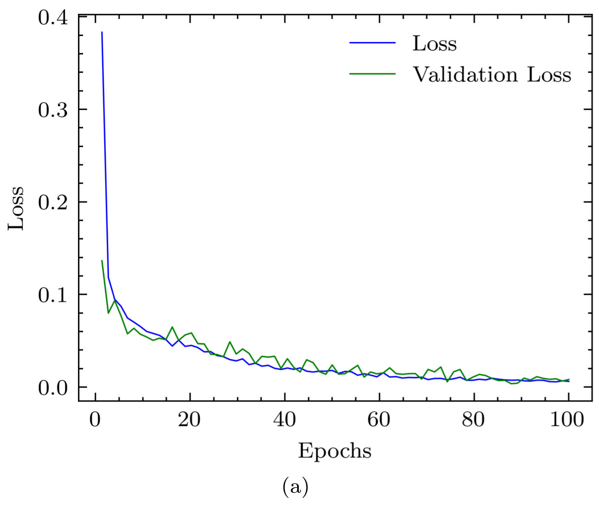
$$b = \lceil 32/n \rceil . \tag{1}$$

where b is the block size. The same number of blocks will be required for the three different values. One word is needed for each block. To accommodate the storage of these words, half of the designated window size is used ($w/2$), and to ensure that the transformer model's prediction process for each word is not disrupted, the overlapping of these windows has been prevented. In this case, considering that each of these 3 values is stored along with the window of each word, the required number of words can be calculated as follows:

$$WC_{header} = 3 \times b \times w/2. \tag{2}$$

In the next step, each character of the hidden text will be stored in a word, and each word will require w number of words without overlapping windows. In this case, the embedding capacity value under the best circumstances will be:

$$EC_{max} = (WC_{words} - WC_{header})/w. \tag{3}$$



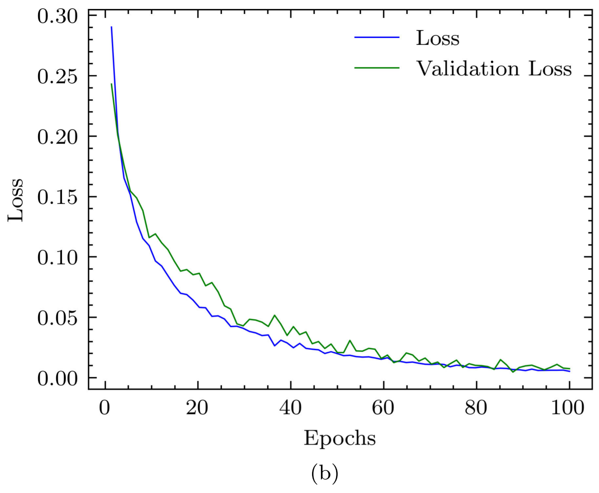


FIGURE 7. Training and validation loss values for English (a) and Turkish (b) models.

The reason of the phrase mentioned as "under the best circumstances" is that due to inappropriate predictions by the transformer model for certain words, the hiding process might not be possible. The obtained value is for a hiding process with 0 *skip* operations.

Using the determined capacity values, stego texts are generated with a language model for half window sizes of 5, 10, and 15. The total byte values of the secret messages are provided in Table 1. Test results for 3 different scenarios, each with 3 different sizes of text, have been provided.

Imperceptibility measurements are conducted for performing tests and comparing the results with other methods. Imperceptibility refers to the similarity between the cover text and the stego text. Imperceptibility detection can be achieved using measures such as perplexity (PPL), KL divergence (KLD), semantic similarity (SS) [31] and BLEU (bilingual evaluation understudy) score. Perplexity is the average occurrence probability of each token in the stego text and it can be calculated as:

$$perplexity = 2^{-\frac{1}{n}\sum_{i=1}^{n} log2(P(x_i|x_{<i}))}. \quad (4)$$

Low perplexity implies high imperceptibility. For obtaining the probability distributions of tokens in the text and calculating their difference, KL divergence is employed. Let P and Q represent the probability distributions of tokens in the original and hidden texts, respectively. A lower KL divergence indicates a higher level of imperceptibility. The calculation of KL divergence is as follows:

$$KLD(P||Q) = \sum_{i=1}^{N}[p(x_i)logp(x_i) - p(x_i)logq(x_i)]. \quad (5)$$

Another comparative method that can be used is semantic similarity. In this approach, the cosine distance between the embeddings of tokens in the cover text and stego text is calculated. To measure semantic similarity, it's necessary to detect the semantic distance for the words. Various models can be used to achieve this. In this study, since the models used are based on the BERT model, the sentence similarity models efederici/sentence-bert-base and emrecan/bert-base-turkish-cased-mean-nli-stsb-tr, which are also BERT-based sentence similarity models, have been utilized for similarity measurement.

Finally, BLEU scores have also been calculated for measuring the similarity of texts. While BLEU is primarily used to assess the performance of machine translations, it can technically be utilized to measure semantic similarity, as it evaluates the semantic closeness of a given sentence in comparison to a reference sentence.

To facilitate comparison with other studies, the embedding rate (ER) calculation from [31] is used, and embedding rate values for the given calculation are obtained. The resulting ER, PPL, KLD, SIM, and BLEU values are presented in Table 2.

In order to ensure a valid comparison for the values presented in Table 2, the unchanged portion of the text has not been considered. For instance, when a 25% data hiding process is applied, values were obtained based on the section up to the last modified word. This way, the intention was to prevent the remaining 75% of the text from contributing to improvements in the results without any changes.

As observed in Table 2, as the half window size decreases and more words are subjected to the hiding process, the ER value increases. Upon examining all configurations, there aren't significant changes in PPL and KLD values. For Turkish, higher PPL values are obtained, and it is evident that changing selected words have a more significant impact on imperceptibility. The reason some SIM values are 1 is due to providing results with only 2 decimal points of sensitivity for English. SIM values are as high as 0.99 for English and 0.88 for Turkish. Finally, when analyzing the BLEU scores, as expected, the scores increase as the data hiding process decreases. Even in the worst case, the BLEU value has not fallen below the 0.78 mark. To facilitate a comparison with results from other studies in the literature and to evaluate the proposed method against them, the obtained values are provided in Table 3.

TABLE 1. Amount of hidden bytes for each cover text scenario.

Corpus	Hidden Message Size Ratio	Half Window Size		
		5	10	15
English50Kb	25%	225	110	75
	50%	400	200	130
	100%	800	400	260
English100Kb	25%	375	200	125
	50%	750	375	250
	100%	1500	750	500
English250Kb	25%	1100	550	375
	50%	2250	1100	750
	100%	4500	2200	1500
Turkish50Kb	25%	125	60	35
	50%	250	125	75
	100%	500	250	150
Turkish100Kb	25%	250	125	75
	50%	500	250	150
	100%	1000	500	300
Turkish250Kb	25%	700	250	225
	50%	1400	700	450
	100%	2800	1400	900

TABLE 2. PPL. KLD. SIM and BLEU values for cover and stego texts.

Cover Text Size	Half Window Size	Ratio	ER	English				Turkish			
				PPL	KLD	SIM	BLEU	PPL	KLD	SIM	BLEU
50	5	100%	0.63	11.55	5.64	0.99	0.78	41.83	12.46	0.88	0.78
		50%	0.31	11.71	5.60	0.99	0.89	40.37	12.46	0.88	0.89
		25%	0.18	11.82	5.62	0.99	0.94	40.41	12.79	0.89	0.94
	10	100%	0.31	11.48	3.31	**1.00**	0.89	40.09	10.64	0.96	0.89
		50%	0.16	11.71	**3.25**	**1.00**	0.94	39.73	10.64	0.96	0.94
		25%	0.09	11.81	3.53	**1.00**	0.97	39.79	10.64	0.96	0.97
	15	100%	0.20	11.77	3.56	**1.00**	0.93	39.65	4.50	**0.97**	0.93
		50%	0.10	11.78	3.56	**1.00**	0.96	39.36	4.50	**0.97**	0.96
		25%	0.06	11.84	3.56	**1.00**	0.98	38.83	4.50	**0.97**	0.98
100	5	100%	0.59	**11.17**	8.35	0.99	0.80	41.37	12.91	0.86	0.78
		50%	0.29	11.22	8.31	0.99	0.90	39.93	12.91	0.86	0.89
		25%	0.15	11.24	8.33	0.99	0.95	39.33	12.72	0.86	0.94
	10	100%	0.29	11.21	7.28	0.99	0.90	39.98	10.03	0.93	0.89
		50%	0.15	11.29	7.28	0.99	0.95	39.39	10.03	0.93	0.94
		25%	0.08	11.33	7.28	0.99	0.97	39.01	9.15	0.93	0.97
	15	100%	0.20	11.25	5.69	**1.00**	0.93	39.38	8.79	0.95	0.93
		50%	0.10	11.28	5.69	**1.00**	0.97	38.96	8.79	0.95	0.97
		25%	0.05	11.30	5.69	**1.00**	0.98	**38.80**	7.88	0.96	0.98
250	5	100%	0.70	11.52	9.81	0.99	0.77	45.37	14.33	0.89	0.76
		50%	0.35	11.46	8.55	0.99	0.88	41.48	12.01	0.88	0.88
		25%	0.17	11.57	8.53	0.99	0.94	40.56	12.01	0.88	0.94
	10	100%	0.35	11.43	6.14	**1.00**	0.89	41.36	10.81	0.93	0.88
		50%	0.17	11.50	5.61	**1.00**	0.94	40.66	9.81	0.94	0.94
		25%	0.09	11.54	5.61	**1.00**	0.97	40.09	9.81	0.94	**0.98**
	15	100%	0.24	11.46	5.28	**1.00**	0.92	39.78	**3.29**	0.96	0.92
		50%	0.12	11.50	4.80	**1.00**	0.96	40.28	5.62	0.96	0.96
		25%	0.06	11.56	4.80	**1.00**	0.98	40.03	6.37	0.96	**0.98**

TABLE 3. Imperceptibility results of different methods.

Model	ER	KLD	PPL	SIM
Proposed Method	0.64	**7.93**	11.44	**0.99**
Block [25]	**0.89**	48.46	234.74	0.64
RNN-Stega [38]	0.78	15.71	48.37	0.66
Patient-Huffman [39]	0.47	15.96	21.61	0.73
Transformer [40]	0.8	15.15	8.94	0.58
Joint Ling. Steg. [31]	0.74	14.61	**6.04**	0.82

In Table 3, the results from the article [31] are given under the condition where each bit hidden per word is 5. For the proposed method, the configuration with the closest ER values was selected, and the results pertaining to this configuration were provided. The proposed method hides a letter for each word, and technically, it hides 8 bits when other algorithms hide 5 bits per word. Nevertheless, as evident in the table, it achieves the lowest KLD values and the highest SIM values compared to other methods. As for PPL, it appears to be quite successful compared to other methods, excluding the approaches in [31] and [40].

Additionally, sentence examples demonstrating the difference between the original and information-hidden data are provided in Table 4 to illustrate this contrast.

Table 4 highlights words in bold that store letters at the loopIndex value. In the first example, the letter "e" in

TABLE 4. Replaced words in cover text for English and Turkish languages.

English
. . . and every moment that ticked by they spent away from callie made it less likely. . .
. . . and every moment that ticked by they spent away from callie made it less likely. . .
. . . how would anyone ever know if they were being assisted from the other side?...
. . . how should anyone ever know if they were being assisted from the other side?....
...and yet he did this for her . and alu didn't...
...and yet he tried this for her . and alu didn't...
...you can have one of these' she said , offering her packet...
...you can have five of these' she said , offering her packet...
...ill be knocking on your doorstep , freezing myself to death soon , said heston...
...ill be knocking on your doorstep , freezing yourself to death soon , said heston...
Turkish
. . . Uçuşlar İstanbul-Sao Paulo-İstanbul hattının Buenos Aires'e uzatılması ile gerçekleştirilecek. . .
. . . Uçuşlar İstanbul-Sao Paulo-İstanbul hattının Buenos Aires'e uzatılması istemiyle gerçekleştirilecek. . .
. . . 3 ay önce dünyaya getirdiği bebeği Batuhan Alan'ın öldürülmesi olayıyla ilgili takibini sürdüren polis,. . .
. . . 3 ay önce dünyaya getirdiği bebeği Batuhan Alan'ın öldürülmesi olayıyla ilgili işlemleri sürdüren polis,. . .
...İkincisi, yerel yönetim gerçeği! 1 milyona yaklaşan...
...İkincisi, doğru yönetim gerçeği! 1 milyona yaklaşan...
...yapılan analizlerde, tanınmış hemen hemen tüm firmaların sularında...
...yapılan analizlerde, incelemelerde hemen hemen tüm firmaların sularında...
...Önümüzdeki yıllarda, 4-7 yıl sonra ne olabileceğini tahmin ediyoruz...
...Önümüzdeki yıllarda, 4-7 yıl neticesinde ne olabileceğini tahmin ediyoruz...

the word "every" within the cover text is already present for loopIndex=2, requiring no modification. In the second example, the word "would" has been replaced with "should" to store the letter "s" at loopIndex=0. Similar adjustments have been made in other examples to store the necessary letters. As evident from the examples, these changes do not disrupt the sentence structure. Additionally, situations where words can store letters without alteration contribute to obtaining better SIM and KLD values.

IV. CONCLUSION

Steganography is a significant field in ensuring information security by concealing the presence of data. It primarily focuses on hiding the existence of information, often achieved at the bit level in many studies. Additionally, methods exist that perform information hiding through text generation, embedding data during the generation process.

In this study, a method is proposed that differs from generation-based approaches. Instead of generating new text, the method utilizes an existing cover media and employs a transformer architecture with a word prediction mechanism to achieve character-level information hiding. Unlike generative methods, this approach utilizes a pre-existing cover text, making the detection of hidden information more challenging compared to generative techniques.

Many existing methods in the literature work with word indices, while language models generate suggestions based on the probability distribution. As the index values increase, the probability of substituting lower-probability words rises, potentially increasing the recoverability of the hidden data. In the proposed method, the use of a specific character at a certain index in the searched word eliminates the significance of the predicted word's index. Consequently, the first suitable word with the highest probability can be used for data hiding.

The success of the method depends on the language model's ability to make accurate predictions or predict different numbers of words. Therefore, employing better-trained or larger models in the method could result in fewer skipping processes during data hiding, enhancing the method's performance. Since the method utilizes the masked language modeling feature of models, it can be easily adapted to different languages and models.

The method achieves a high embedding capacity by storing one character per word. Although reducing the window size could increase this capacity, it might diminish the language model's capability to make accurate word predictions, thereby increasing the likelihood of revealing the presence of hidden data in stego text.

Upon examining the results and comparing them with other methods, it's evident that by achieving close ER values to other methods and low imperceptibility, the hiding process can be performed effectively. As a result, this approach generates more successful outcomes compared to other methods.

The proposed method demonstrates from the test results that the difference between stego text and cover text is minimal, and the low distinguishability of changed words indicates the success of the method. Lastly in the link https://anonymous.4open.science/r /MLMCharStego-4F69, the developed method and models can be accessed.

The proposed model is suitable for hiding text within text, as it performs hiding based on the letter values and can hide information in sections corresponding to alphanumeric characters in the ASCII table. The disadvantage of this approach is its unsuitability for hiding binary data such as images, audio, or video. In future studies, training models compatible with Unicode characters and expanding the character set will open up the possibility of storing specific data with preprocessing and transforming operations. The aim is to carry out this process by mapping binary data to values in the character set.

REFERENCES

[1] T. Lu, G. Liu, R. Zhang, P. Li, and T. Ju, "Robust secret data hiding for transformer-based neural machine translation," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2023, pp. 1–8.

[2] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding—A survey," *Proc. IEEE*, vol. 87, no. 7, pp. 1062–1078, Jul. 1999.

[3] L. Xiang, R. Wang, Z. Yang, and Y. Liu, "Generative linguistic steganography: A comprehensive review," *KSII Trans. Internet Inf. Syst.*, vol. 16, no. 3, pp. 986–1005, 2022.

[4] B. Pfitzmann, "Information hiding terminology," in *Information Hiding* (Lecture Notes in Computer Science), vol. 1, no. 174. 1996, pp. 347–350.

[5] M. Kharrazi, H. T. Sencar, and N. Memon, *Image Steganography: Concepts and Practice* (Lecture Note Series), Singapore: National University of Singapore, Institute for Mathematical Sciences, 2004.

[6] J. Wen, X. Zhou, P. Zhong, and Y. Xue, "Convolutional neural network based text steganalysis," *IEEE Signal Process. Lett.*, vol. 26, no. 3, pp. 460–464, Mar. 2019.

[7] Ö. Kurtuldu and M. Demirci, "StegoGIS: A new steganography method using the geospatial domain," *TURKISH J. Electr. Eng. Comput. Sci.*, vol. 27, no. 1, pp. 532–546, Jan. 2019.

[8] İ. Coşkun, F. Akar, and Ö. Çetin, "A new digital image steganography algorithm based on visible wavelength," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 21, no. 2, pp. 548–564, 2013.

[9] K. Wang and Q. Gao, "A coverless plain text steganography based on character features," *IEEE Access*, vol. 7, pp. 95665–95676, 2019.

[10] Y. Tong, Y. Liu, J. Wang, and G. Xin, "Text steganography on RNN-generated lyrics," *Math. Biosciences Eng.*, vol. 16, no. 5, pp. 5451–5463, 2019.

[11] Y.-W. Kim, K.-A. Moon, and I.-S. Oh, "A text watermarking algorithm based on word classification and inter-word space statistics," in *Proc. 7th Int. Conf. Document Anal. Recognit. (ICDAR)*, Aug. 2003, pp. 775–779.

[12] A. M. Alattar and O. M. Alattar, "Watermarking electronic text documents containing justified paragraphs and irregular line spacing," *Proc. SPIE*, vol. 5306, pp. 685–695, Jan. 2004.

[13] B. K. Ramakrishnan, P. K. Thandra, and A. V. S. M. Srinivasula, "Text steganography: A novel character-level embedding algorithm using font attribute," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 6066–6079, Dec. 2016.

[14] R. Kumar, A. Malik, S. Singh, B. Kumar, and S. Chand, "A space based reversible high capacity text steganography scheme using font type and style," in *Proc. Int. Conf. Comput., Commun. Autom. (ICCCA)*, Apr. 2016, pp. 1090–1094.

[15] M.-Y. Kim, O. R. Zaiane, and R. Goebel, "Natural language watermarking based on syntactic displacement and morphological division," in *Proc. IEEE 34th Annu. Comput. Softw. Appl. Conf. Workshops*, Jul. 2010, pp. 164–169.

[16] A. Ş. Mesut, Ö. Aydin, and E. Öztürk, "Anlamsal yöntemler kullanan bir metin steganografi uygulaması," in *Proc. 1st Int. Symp. Digit. Forensics Secur. (ISDFS)*, 2013, pp 21–24.

[17] R. Stutsman, C. Grothoff, M. Atallah, and K. Grothoff, "Lost in just the translation," in *Proc. ACM Symp. Appl. Comput.*, Apr. 2006, pp. 338–345.

[18] L. Xiang, X. Wang, C. Yang, and P. Liu, "A novel linguistic steganography based on synonym run-length encoding," *IEICE Trans. Inf. Syst.*, vol. E100.D, no. 2, pp. 313–322, 2017.

[19] L. Xiang, Y. Li, W. Hao, P. Yang, and X. Shen, "Reversible natural language watermarking using synonym substitution and arithmetic coding," *Comput., Mater. Continua*, vol. 55, no. 3, pp. 541–559, 2018.

[20] X. Yang, F. Li, and L. Xiang, "Synonym substitution-based steganographic algorithm with matrix coding," *Chin. Comput. Syst.*, vol. 36, pp. 1296–1300, Oct. 2015.

[21] C.-Y. Chang and S. Clark, "Practical linguistic steganography using contextual synonym substitution and a novel vertex coding method," *Comput. Linguistics*, vol. 40, no. 2, pp. 403–448, Jun. 2014.

[22] C. Qi, S. Xingming, and X. Lingyun, "A secure text steganography based on synonym substitution," in *Proc. IEEE Conf. Anthology*, Jan. 2013, pp. 1–3.

[23] I. A. Bolshakov, "A method of linguistic steganography based on collocationally-verified synonymy," in *Proc. Int. Workshop Inf. Hiding*. Cham, Switzerland: Springer, 2004, pp. 180–191.

[24] I. A. Bolshakov and A. Gelbukh, "Synonymous paraphrasing using wordnet and internet," in *Proc. Int. Conf. Appl. Natural Lang. Inf. Syst.* Cham, Switzerland: Springer, 2004, pp. 312–323.

[25] T. Fang, M. Jaggi, and K. Argyraki, "Generating steganographic text with LSTMs," 2017, *arXiv:1705.10742*.

[26] H. Kang, H. Wu, and X. Zhang, "Generative text steganography based on LSTM network and attention mechanism with keywords," *Electron. Imag.*, vol. 32, no. 4, pp. 1–291, Jan. 2020.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 2672–2680.

[28] W. Tang, S. Tan, B. Li, and J. Huang, "Automatic steganographic distortion learning using a generative adversarial network," *IEEE Signal Process. Lett.*, vol. 24, no. 10, pp. 1547–1551, Oct. 2017.

[29] R. Zhang, S. Dong, and J. Liu, "Invisible steganography via generative adversarial networks," 2018, *arXiv:1807.08571*.

[30] B. Yi, H. Wu, G. Feng, and X. Zhang, "ALiSa: Acrostic linguistic steganography based on BERT and Gibbs sampling," *IEEE Signal Process. Lett.*, vol. 29, pp. 687–691, 2022.

[31] C. Ding, Z. Fu, Q. Yu, F. Wang, and X. Chen, "Joint linguistic steganography with BERT masked language model and graph attention network," *IEEE Trans. Cogn. Devel. Syst.*, p. 1, 2023.

[32] K. S. Kalyan, A. Rajasekharan, and S. Sangeetha, "AMMUS : A survey of transformer-based pretrained models in natural language processing," 2021, *arXiv:2108.05542*.

[33] M. S. Jahan, H. U. Khan, S. Akbar, M. U. Farooq, S. Gul, and A. Amjad, "Bidirectional language modeling: A systematic literature review," *Sci. Program.*, vol. 2021, pp. 1–15, May 2021.

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[35] SUDA Research and A. C. Verim. (2018). *Suder Corpus—Turkish News Collections for Text Categorization*. [Online]. Available: https://github.com/suverim/suder

[36] P. Ferragina and G. Navarro. *Pizza&Chili—Corpus Compressed Indexes and Their Testbeds*. Accessed: Jun. 12, 2023. [Online]. Available: http://pizzachili.dcc.uchile.cl/texts.html

[37] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 19–27.
[38] Z.-L. Yang, X.-Q. Guo, Z.-M. Chen, Y.-F. Huang, and Y.-J. Zhang, "RNN-stega: Linguistic steganography based on recurrent neural networks," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 5, pp. 1280–1295, May 2019.
[39] F. Z. Dai and Z. Cai, "Towards near-imperceptible steganographic text," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 4303–4308.
[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023.

EMİR ÖZTÜRK received the B.S. (Hons.) and M.S. degrees from Trakya University, in 2010 and 2012, respectively, and the Ph.D. degree in computer science from Yıldız Technical University, in 2018.

He was a Research Assistant with the Computer Engineering Department, Trakya University, from 2010 to 2021, where he started working as an Assistant Professor, in 2021. His research interests include data compression, deep learning algorithms, information security, and programming languages.

ANDAÇ ŞAHİN MESUT received the B.S., M.S., and Ph.D. degrees in computer engineering from Trakya University, Edirne, Turkey, in 2007.

From 1998 to 2007, she was a Research Assistant with the Computer Engineering Department, Trakya University, where she has been an Assistant Professor, since 2007. Her research interests include cryptography and steganography.

ÖZLEM AYDIN FİDAN received the B.S., M.S., and Ph.D. degrees in computer engineering from Trakya University, Edirne, Turkey, in 2011.

From 2001 to 2011, she was a Research Assistant with the Computer Engineering Deparment, Trakya University, where she has been an Assistant Professor, since 2011. Her research interests include natural language processing and text steganography.