Scientific Research Publishing

Journal of Information Security, 2024, 15, 279-297
https://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

Cyber Deception Using NLP

Igor Godefroy Kouam Kamdem, Marcellin Nkenlifack

Department of Mathematics and Computer Science, URIFIA Laboratory, Faculty of Science, University of Dschang, Dschang, Cameroon

Email: igorkouam5@gmail.com, marcellin.nkenlifack@gmail.com

How to cite this paper: Kamdem, I.G.K. and Nkenlifack, M. (2024) Cyber Deception Using NLP. *Journal of Information Security*, **15**, 279-297.
https://doi.org/10.4236/jis.2024.152016

Received: March 24, 2024
Accepted: April 27, 2024
Published: April 30, 2024

Abstract

Cyber security addresses the protection of information systems in cyberspace. These systems face multiple attacks on a daily basis, with the level of complication getting increasingly challenging. Despite the existence of multiple solutions, attackers are still quite successful at identifying vulnerabilities to exploit. This is why cyber deception is increasingly being used to divert attackers' attention and, therefore, enhance the security of information systems. To be effective, deception environments need fake data. This is where Natural Language (NLP) Processing comes in. Many cyber security models have used NLP for vulnerability detection in information systems, email classification, fake citation detection, and many others. Although it is used for text generation, existing models seem to be unsuitable for data generation in a deception environment. Our goal is to use text generation in NLP to generate data in the deception context that will be used to build multi-level deception in information systems. Our model consists of three (3) components, including the connection component, the deception component, composed of several states in which an attacker may be, depending on whether he is malicious or not, and the text generation component. The text generation component considers as input the real data of the information system and allows the production of several texts as output, which are usable at different deception levels.

Keywords

Cyber Deception, Cybersecurity, Natural Language Processing, Text Generation

1. Introduction

The digital revolution has allowed the development of sophisticated information systems as well as made information exchange easier. This has given rise to security problems at both the local and network levels. Numerous solutions have been developed to deal with security problems, such as encryption and hashing

algorithms [1] [2] [3] [4]. Kouam and Marcellin [1] have proposed a security model that provides multi-level security of data by using biometric authentication of information exchanges. The interest is to prevent data theft and fraudulent information transfers in cyberspace.

Cybersecurity is the branch of computer science that focuses on protecting information systems in cyberspace. With the rapid proliferation of digital devices, the number of attacks is increasing, making it more important to maintain security. However, using cyber deception is a component of ensuring quality security. Many models like [5] [6] [7], and others have proposed various solutions using different methods for the deception of attackers in systems with the use of chatbots, game theory, and others. Machine-based text analysis was proposed by [8] for sentence classification. Implementing a deception system involves the creation of the environment itself and the fake data. The need to automate the data generation process becomes crucial with the volume of data to consider.

Natural Language Processing (NLP) can be defined as the automatic manipulation of natural languages, like speech and text, by software. NLP is used in cybersecurity to address several tasks, like malicious domain name detection [9], spam detection or mail classification [10], vulnerability detection [11], and others. NLP is also used for deception to encrypt the context of a message [12], encrypt relevant information in data [13], duplicate documents in a server [14], and many others. To the best of our knowledge, we have not seen any model using NLP for data generation in a cyber deception context, and this work is the first one to address the issue.

The fake data generation in the context of deception must consider the actual data during the generation process to be consistent. Given the fact that some texts (documents) can be composed of several domains, the model must be able to consider each field contained in the text. Our main object is to propose a text generation model for the cyber deception context.

The rest of this work will be organized as follows: section 2 will give a brief state of the art on NLP in cybersecurity domain and text generation models, section 3 will present methodology, and finally, section 4 will present some results.

2. State of the Art

Cyber deception tends to touch multiple domains in order to implement an effective security solution. It uses game theory, graph theory, NLP and others. Language is the main tool that promotes human-machine teamwork in cybersecurity activities. Among the different tasks that can be performed using NLP, some can be used to create threats or attacks or also to secure resources in information systems.

We can use NLP to encrypt the context of a text [12]. The method takes a message and replaces a part of the message with another message which does not have a relationship but we can read a part on real message and this part can contain the main information which is a real problem in the deception context. To perform a deception environment, Tanmoy [14] proposes "a fake repository

Engine for cyber deception". The model can identify important documents in a server and for each document, a set of fake documents is generated and associated with the original document. The goal is to waste the time of the attacker to intercept it either during the download of the set of documents or the search of the original document directly in the server. The authors did not mention document content, but just how to identify and duplicate it.

Prakruthi *et al.* [15] proposed fake document generation for cyber deception by manipulating text comprehensibility. They adopt a set of quantitative measures based on qualitative principles of psycholinguistics and reading comprehension: connectivity, dispersion, and sequentiality. Given an input text, the authors introduce some sentences or remove some sentences to make sure that attacker misunderstands the proposed text. The problem is that the proposed method identifies sentences containing target concepts to manipulate their positioning; however, the selection of sentences is purely based on the occurrence of concepts. It does not consider a different rephrase of a concept or related information in the selected sentences. Another problem is that we can see a part of real information contrary to our work.

Syntactic analysis is one of the key tools used to complete the tasks of the NLP, which is used to determine how the natural language aligns with the grammatical rules. The most widely used techniques in NLP are lemmatization, morphological segmentation, word segmentation, part-of-speech marking, parsing, sentence breaking, stemming, etc.

Indeed, semantic analysis can allow the classification of data and can thus classify, for example, emails received as an attack or a normal email [10]. The applicability of feature extraction for malicious message filtering is determined by text mining methodologies [16].

Data analysis has become more complicated with large volumes of data and the diverse properties that data presents. This does not make it easy for data analysis methods to consider all the properties to produce a good analysis. Thus, NLP can be used to reduce the dimensionality of the data by extracting features for a more efficient analysis [17]. This allows for the removal of duplicates and others in the considered data. Many vulnerabilities exploited by attackers, however, are sometimes found in the programs we use. Indeed, some programs have had security flaws since the development process. Mokhov *et al.* [18] used n-grams NLP techniques combined with machine learning for the detection, classification, and reporting of weaknesses related to vulnerability or bad coding practices found in artificially constrained language.

The expanding number of reports of cyberattacks can make a deeper analysis of such data prohibitively time-consuming. However, shallow text analysis cannot provide many of the details necessary to yield actionable steps for improving security measures against the vast number of cyberattacks occurring each day. Pawlick *et al.* [7] use game theory in order to deceive the attacker. Yanlin Chen *et al.* [8] proposed machine-based text analysis, which builds automatic sentence classification. As given a new category, it can automatically update training data

and build a tool to analyze the text of cybersecurity strategies. Another problem is malicious domain name detection, which can be done using lexical analysis [9]. NLP is increasingly used nowadays by cybercriminals and security defense tools in the understanding and processing of unstructured data generated. NLP's ultimate aim is to extract knowledge from unstructured data or information [19]. Behavior modeling is used to detect malware and attacker in an information system. To formulate a behavior report, [11] used the bag-of-word (BoW) of NLP.

The NLP can be used for text generation tasks and it is evolving rapidly. The goal is to generate text that looks as real as possible to humans. Text generation involves the prediction of words for sentence construction. One of the most widely used models for this task is the sequence-to-sequence (Seq2Seq) model, where the recent model includes attention [20] [21], etc.

There are some topics in the text generation domain. We have open domain dialogue [22] [23] [24] where [22] focuses on the specific topic of the current conversation and makes automatic changes, [23] uses unconventional texts for training the model and [24] proposes a model which associates images on the different word during the conversion process.

Many generation models have focused on building long texts such as paragraphs, long sentences or documents [25]-[31], but some problems can be observed. These models cannot generate a long sentence; they have problems with the style of output, syntax, context, and others. However, there is no specific point on text generation for the cyber deception context that chatGPT addresses. In order to be used in cyber deception, the text generation model must keep some properties such as context preservation, domain preservation, syntax, consistency, size, etc.

3. Methodology

Many information systems are vulnerable to multi-level attacks. Attackers use this technique to disrupt defenders and to carry out their attacks. However, multi-level deception aims at setting deception barriers to intercept them.

3.1. Deception Architecture

Figure 1 presents our architecture on multi-level deception which has two (2) levels:

All users who want to access in system start by an authentication point. We used the authentication model proposed by Kouam and Nkenlifack [1] with biometric authentication.

After the authentication process, the intrusion detection system classifies users into three groups: malicious users, legitimate users, and uncertain users. Uncertain users are those whose system has not been able to identify them as legitimate or as attackers.

The attackers then have access to a database containing fake data, while legitimate users access the real data. The fake data is obtained using a generation module that takes the real data as input and provides the generated data as output. However, the actions of attackers are collected for analysis and security

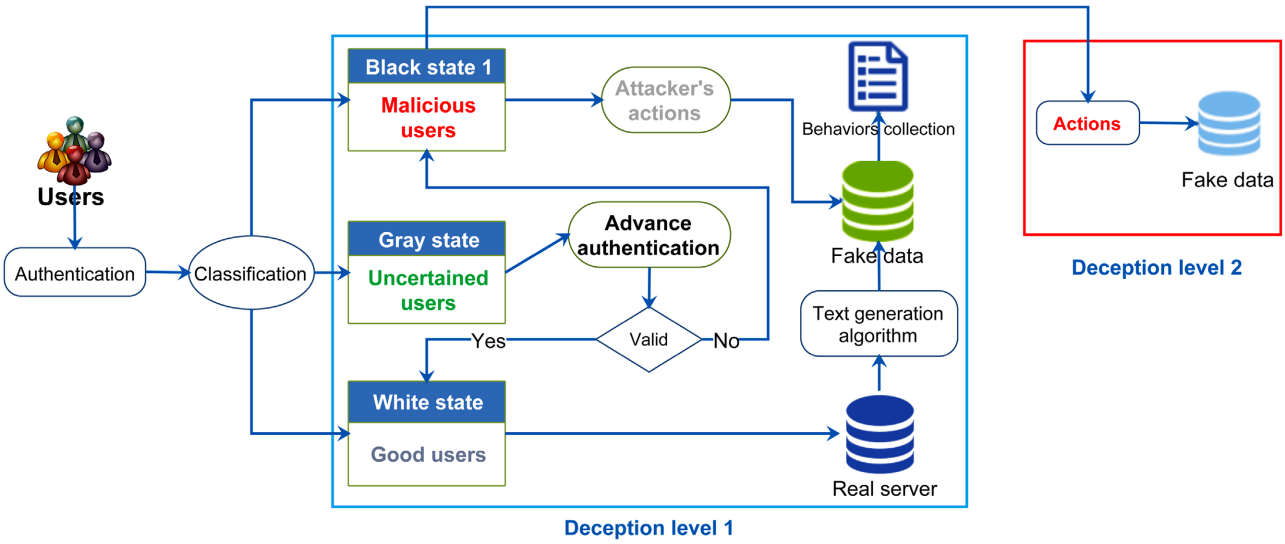

Figure 1. Multi-stage deception architecture.

enhancement purposes. These actions can include attempts at unauthorized access, data deletion, query modification, and many others. When the system realizes that the attacker's interaction is decreasing (multiple attempts at fraudulent action), the attacker is redirected to the second level of deception with the response to his request. The attacker will then believe that he has achieved his objective or is making progress in the system. The data generation module is designed in such a way that the data generated at the first deception level slightly generates the data available at the second level.

3.2. Multi-Stage Deception

Multi-level deception consists of moving the attacker from one level of deception to another. Indeed, when the attacker's behavior shows that he is no longer interacting sufficiently with the system and that he is making requests to access the system, he is redirected to the second deception level. This shows the attacker that he is making progress. Based on Figure 1, we proposed the following multi-level deception management algorithm (algorithm1) in Figure 2.

Let BS, WS, and GS be the black state, white state, and gray state, respectively. Let u be a user, and L the level of deception the user u is in. When the user authenticates, he or she is placed in one of the system states: black, white, or gray. When the user is placed in the black state, he is redirected to the first level of deception, and the data to which he has access is fake. If the user is placed in the gray state and the number of connection attempts has not been exhausted, the system switches to a forced authentication approach. If this fails, the user is considered an attacker and placed in the black state. If not, he's considered a legitimate user. Legitimate users are redirected to the real information system, where they can access real data.

If the attacker is at the first deception level and his actions demonstrate persistence in breaking into the real system, he is redirected to the second deception

```
Input: File, S, u, L;
c ← authentication_analysis(u);
if (c = True) then
    WS ← u;
else
    if (c = False) then
        BS ← u;
    else
        GS ← u;
    end if
end if
for (All user u in GS do
    aut ← Avanced_authentication(u)
    if (aut) then
        WS ← u
    else
        BS ← u
    end if
end for
for (All user u in WS do
    system ← u
end for
for (All user u in BS do
    l1 ← u
    File ← action_collection(u)
end for
for (all u ∈ l1) do
    if (Untrusted) then
        l2 ← u;
    end if
end for
```

Figure 2. Algorithm for multi-level deception.

level in order to renew his belief. All the attacker's actions at different levels are collected to increase the system's level of security.

3.3. Text Generation Component

The text generation component is used to generate the data used to feed the servers in deception environments. This component considers as input the real data and produces as output the generated data. It must be able to produce several outputs for the same data. It will allow the creation of several deception servers from the same real data server. Considering that the second level of deception contains particular attackers, that is to say, who have an idea of the quality of the data and the system functioning, the data generated for the server of this environment must be more practical.

Deceiving an attacker with text requires that the text be generated in accordance with a number of principles. Among these principles, there are two that are essential for deception to be effective:

- Firstly, the generated text must not provide the attacker with any sensitive information or allow the attacker to deduce any sensitive information. This ensures that the model is risk-free.
- Secondly, the generated text must remain in the same domain as the original text. This increases the attacker's confidence. If, for example, the attack targets the banking system and the attacker receives a medical text, he will quickly realize that the text received is not credible.

In addition to these two elements, we can add the proportionality of input and output text, the consistency of generated text, and multi-context generation. Multi-context or multi-domain generation refers to a text in which several unrelated subjects can be identified.

Thus, the generation model consists of four stages: pre-processing, extraction of sensitive information, replacement of sensitive information, and, finally, generation of the final text. If our model is applied to an enterprise data server with small data sets, the segmentation stage is not necessary, as all the data in the database will be of the same type. Figure 3 shows the architecture of our model.

3.3.1. Step 1: Pre-Processing

This stage consists of removing all characters that could bias the model's training. These include punctuation characters such as commas, semi-colons, periods, question marks, exclamation marks, etc., as well as specific characters such as $, #, &, |, /, etc.

3.3.2. Step 2: Extraction of Sensitive Information

Assume a text T consisting of n sentences and each sentence consisting of m words. The set of sentences and words in text T can be defined by $S = \{s_1, s_2, ..., s_n\}$, with each sentence s_i defined by $w_i = \{w_{1,i}, w_{2,i}, ..., w_{m,i}\}$, respectively. To extract the key parameters, we used the Stanford parser model available at https://nlp.stanford.edu/software/.

Given $G = (C, E, w)$ a graph, where C is the set of sensitive information (concepts) defining the nodes, E the links between sensitive information such that $E \subseteq C * C$, and w a function that associates with each link $(c, c') \in E$ a weight representing the frequency of occurrence of c and c' in T.

If c is used to explain c', then it is important to understand c before c'. Agrawal

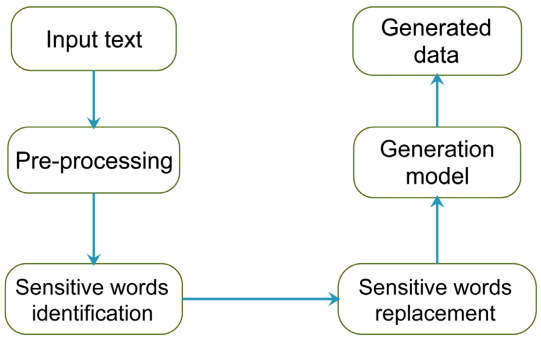

Figure 3. Text generation step.

et al. [32] propose a quantification between c and c' to define the level of comprehension. Given the input text T is made up of a set of concepts $R(c)$ ($c' \in R(c)$, if $w(c, c') > 1$ [32] and the frequency of appearance of c in T noted $f(c, T)$, the sensitivity level of c in T is defined by

$$\delta(c,T) = f(c,T) * |R(c)| \tag{1}$$

The key phrase of the text (the most sensitive phrase) noted, s_k, is defined by the phrase with the highest sensitivity level, $\delta(c, T)$.

Given the graph $G = (C, E, w)$ defined above and the connection links between these different critical words $(c, c') \in E$, the connectivity of c is defined by:

$$\theta(c) = \left(\sum_{(c,c') \in E} w(c,c') \right)^{\lambda} \tag{2}$$

where w is the weight of edges connecting to c, and λ is a constant set [33].

The t connectivity can be given by

$$connect(T) = \frac{\sum_{c \in C} \theta(c)}{|C|} \tag{3}$$

Based on Equations (2) and (3), we can replace each sensitive word $c \in C$. The connectivity graph can help you choose a good word for replacement. This can allow for text consistency.

3.3.3. Step 3: Critical Word Replacement

Critical word replacement involves taking $\theta(c)$ and $connect(T)$ and finding a word or group of words c_r that can replace c in such a way that the text retains its consistency. By replacing the set of critical words in T, we end up with a text T'. Thus, the critical words are defined by the function f_c by:

$$f_c : \begin{matrix} C \to C_r \\ c \mapsto c_r \end{matrix} \tag{4}$$

where C_r is the set of replaced critical words. Thus, c_r connectivity is defined by:

$$\theta(c_r) = \left(\sum_{(c_r, c_r') \in E} w(c_r, c_r') \right)^{\lambda} \tag{5}$$

Similarly, T' connectivity is given by:

$$connect(T') = \frac{\sum_{c_r \in C_r} \theta(c_r)}{|C_r|} \tag{6}$$

3.3.4. Step 4: Text Generation Process

Text is generated after the critical words have been replaced. This ensures that the output text supplied to the attacker cannot contain any sensitive information.

Text generation can be performed in two ways:

- Either a total generation of T', which consists in generating a text from all the words in T'.

- Or by partially generating T', *i.e.*, generating all the other words in the T' text, but without touching the critical words.

The generation process consists of two phases as shown in Figure 4:

- The first phase consists of segmenting the input text into sentences (input sequence) and identifying the context of each sentence (context extraction). This is passed to the generator, which uses the sequence-to-sequence generator and outputs n candidate sequences for each input sequence.
- The second phase takes the candidate sentences and the general context of the input text as input and selects the best candidate sentence close to the general context using similarity based on the cosine function [34].
- The selection function then performs a post-processing operation to reconstitute the various segments into a single output segment.

3.4. Deception Data Base

Generation in the context of cyber deception differs from classical generation in the sense that it is not enough to generate text randomly, but a text that can convince an attacker. We assume that the attackers have partial knowledge of the information they are attacking.

Our generation model is therefore a module that lies between real data and generated data as presented in Figure 5.

For each data of the real system, we have generated data in a deception environment. This module can thus be regarded as a function defined in the following way:

Let D be the set of real data and D' the set of generated data. We can define a function f using Equation (7):

$$f : \begin{matrix} D \to D' \\ d \mapsto f(d) = d' \end{matrix} \tag{7}$$

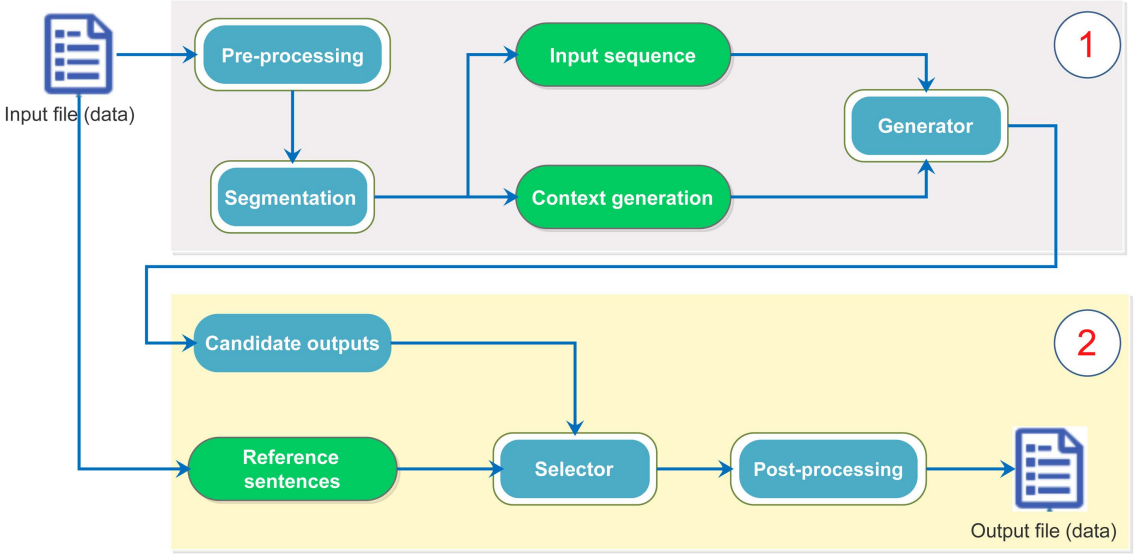

Figure 4. Text generation process.

where d is real data and d' is the generated data. d can be a word, sentence or file.

According to the generation model process expressed by Equation (7), f must be:

- Injective: for two distinct elements d_1 and d_2 in D, we have two distinct elements d'_1 and d'_2 in D'. It is not possible to have the same output from two distinct texts. It ensures that if we have a document with similar content, it will not possible to have a repetitive sentence after generation;
- Surjective: for each element d of D' we have an element d of D from which we have generated d. Each text generated in D' is based on a text d of D.

In view of the above, we can therefore conclude that f must be a bijective function.

In addition to being bijective, this function must be a one-way function. This means that from a generated text d, it must not possible to find the original text d (line number 2). The fact that f must be bijective makes the model more interesting. So, from the same text d of D, we can have several outputs d'_1, d'_2, ..., d'_n of D'. Hence, the ability of the model to provide several outputs makes it possible to build several deception environments from the same data set. The fact that the f function is unidirectional means that there is no risk of finding the real message in a text that has been generated.

4. Simulation Result

4.1. Deception Environment Implementation

The authentication model implemented in this model the one proposed by Kouam and Nkenlifack [1] using two-factor authentication: password and biometric. We also used an intrusion detection system proposed by [35], which enabled us to create three user queues: malicious, uncertain, and legitimate.

We have installed and used WAMP server and MySQL database management systems with three databases: real_base, fake_base1, and fake_base2. Malicious users are connected to fake_base1, legitimate users to real_base, and unsure users are sent back for an authentication process as long as the number of attempts is not exceeded. We have considered three attempts in our simulations.

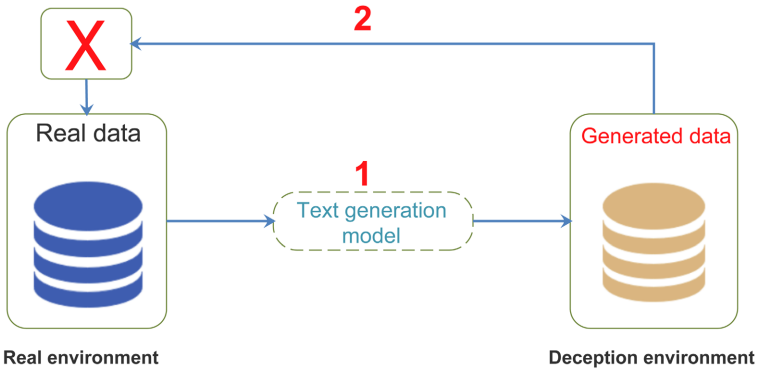

Figure 5. Deception process in data base.

If the first deception level records repetitive actions or bypass attempts, for example, this implies that the attacker is aware that he is in a deception system. We can then transfer him to the second deception level by connecting him to fake_base2 to make him believe that he has succeeded in bypassing security since he has different data. In this way, the first level of deception is used to connect newly detected users, and when the attacker's actions raise doubts, he is sent to the second level of deception.

4.2. Dataset and Implementation Parameters

We used several datasets in order to vary the model evaluation contexts. The first dataset name True dataset [36] was download on Kaggle. This dataset contains a list of articles considered as real news. It contains four columns: a title (the title of the article), a text (the text of the article), a subject (the subject of the article), and a date (the date that this article was posted). For our test, we focus on text and subject columns. The text column contains 21192 unique values and the subject column contains 53% of political News and 47% of world News. In this dataset, we have two (2) contexts (political data and world data) that we will manage. The second one is the dataset of OpenMRS [37]. This database contains a maximum size of 5000 patients with almost 500,000 observations. The third dataset has been collected on the Internet.

The pre-processing process consists of transforming the text into the lower case to make training easier. But also to remove special characters such as ?, |, #, etc., for example. We used the Generative Pre-trained Transformer, one of today's most widely used pre-processing models.

For the implementation, we used the Seq2Seq with attention method for the first module and the Doc2Vec method for the second module. The cosine comparison method is used to make the comparison between two texts. We used the LSTM method with a size of 120 layers, a softmax activation function, and a dropout of 0.1 to build the first module. For the second module, the model is trained using a bidirectional LSTM layer with a size of 512, RELU as activation function, and a dropout of 0.5; a dense layer with the size of the dimension vector, we use ADAM optimizer and logcosh function to compute the loss error, with a learning rate of 0.001. After training the model on 70 epochs, we have an accuracy of 98% for True dataset, 89% for OpenMRS dataset and 91% for data collected on the Internet.

4.3. Example of Text Generated

We ran a simulation, taking data from social networks on the one hand and also running tests on members of our laboratory on the other. The model was evaluated on its ability to identify and replace sensitive words so as to hide the real message. Let's consider the following text in **Figure 6**.

Using the keyword identification model, we obtain **Figure 7**.

In this text, the words in red are words identified as sensitive according to our sensitive word dictionary. The graph of sensitive words and input text is given in

Figure 8 and Figure 9, respectively.

These graphs make it possible to carefully select words to substitute for sensitive words. The result is Figure 10, with the replaced words in blue.

At this point, we can proceed with text generation. The text obtained after generation is present in Figure 11.

In this output, the sensitive words have been replaced with alternative terms to obfuscate the original meaning. As we can see, "packet delivery" has been replaced with "secure item transfer," "Cleveland Clinic" has been replaced with "City Medical Center," "Hospital" has been replaced with "Facility Medical Facility," and "parcel" has been replaced with "package." These changes aim to deceive potential attackers by altering the context and making it harder to discern the true meaning of the text.

4.4. Evaluation Metrics

For evaluation, we have use four metrics:

Precision: Precision measures how accurately our model identifies and replaces sensitive information. It calculates the ratio of true positive replacements (accurately identified and replaced sensitive information) to the total number of replacements made by the model. A higher precision indicates a lower rate of false positives.

> The UAV is moving from the US to the UK for packet delivery. The drone will move from Cleveland Clinic at 7:00 AM to Barnet Hospital, with an expected arrival time of 9:00 PM. After delivering the packer, the drone will return after 3 hours.

Figure 6. Input text.

> The UAV is moving from the US to the UK for packet delivery. The drone will move from Cleveland Clinic at 7:00 AM to Barnet Hospital, with an expected arrival time of 9:00 PM. After delivering the packet, the drone will return after 3 hours.

Figure 7. Text with identified sensitive information.

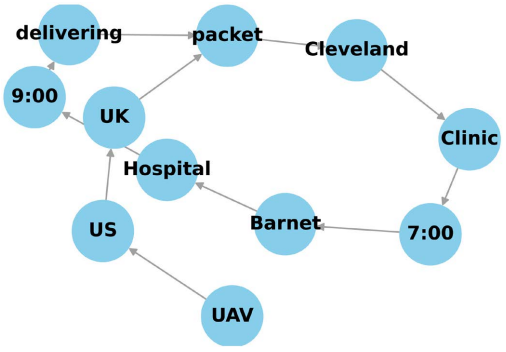

Figure 8. Sensitive words graph.

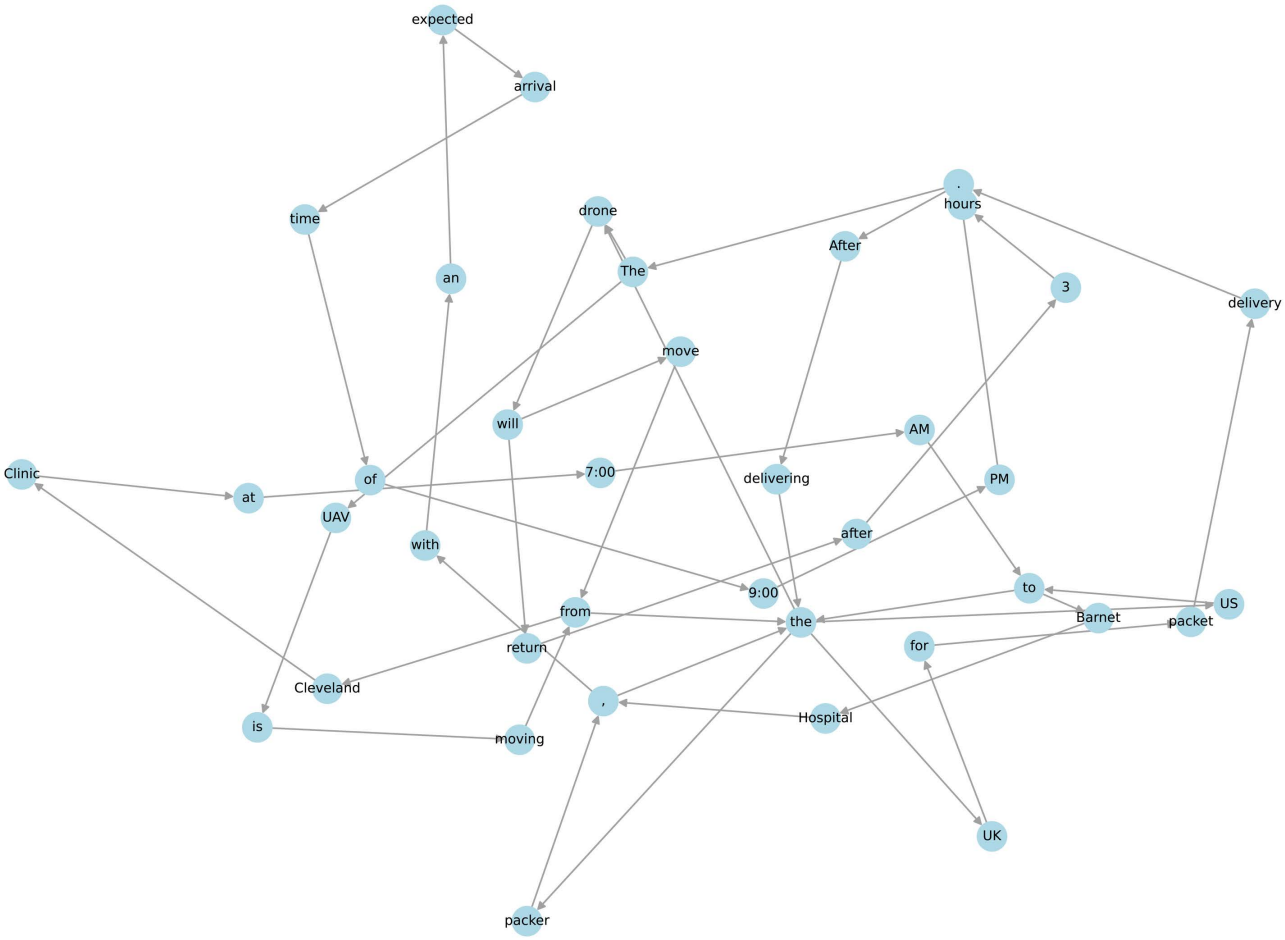

Figure 9. Input text graph.

```
The Widget is moving from the Country to the Destination
for parcel delivery. The drone will move from City Medical
Center at Morning AM to Facility Medical Facility, with an
expected arrival time of Evening PM. After shipment the
parcel, the drone will return after 3 hours.
```

Figure 10. Text with sensitive information replaced.

The widget is being transported from the Country to the Destination for secure item transfer. The autonomous device will depart from City Medical Center early in the day and reach the Facility Medical Facility by the late evening. Following the shipment of the package, the device will make its way back after a brief period of 3 hours.

Figure 11. Final text generated.

Recall: Recall measures how well the model captures all instances of sensitive information. It calculates the ratio of true positive replacements to the total number of actual sensitive instances in the text. A higher recall indicates a lower rate of false negatives.

F1 Score: The F1 score provides a balanced measure of precision and recall. It is the harmonic mean of precision and recall, giving equal weight to both metrics. The F1 score helps evaluate the overall performance of the model in identifying and replacing sensitive information.

Human Evaluation: Human evaluation can help identify any nuanced errors or improvements that may not be captured by automated metrics. We have used five users to make human evaluations.

4.5. Evaluation

The evaluation metric can be presented in Figure 12 as:

For the human test, we asked five users to provide us with data and manually identify their keywords. We ran the data through our model and obtained the following results present in Table 1.

By using data in Table 1, we can plot the graph in Figure 13.

4.6. Discussion

Babu [12] proposes a deception model by encrypting the context message using NLP. The pattern involves replacing part of the input text. The output can be inconsistent, and we can also see the critical information in the output. Kushwaha *et al.* [13] encrypt relevant information in order to keep the information. The problem is that after changing the sensitive information, the attacker can read the message and deduce the sensitive one. Tanmoy [14] proposes a model that, for each document on a server, generates n documents and keeps them on the same server. The problem is that the attacker may be lucky and directly have

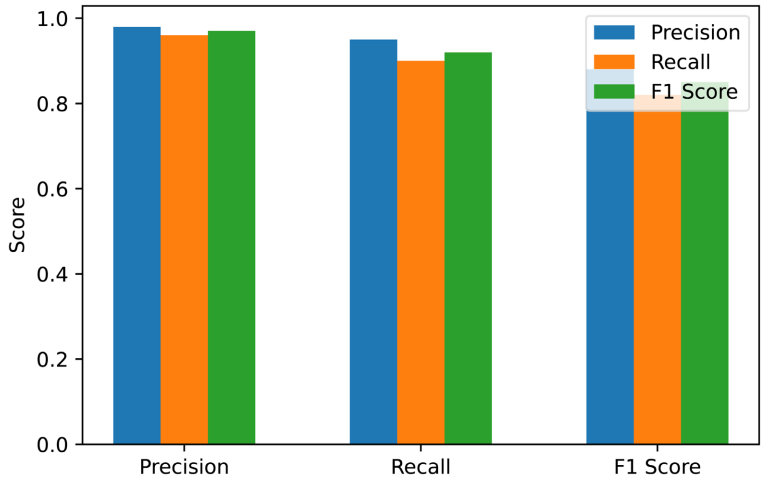

Figure 12. Evaluation metrics.

Table 1. Evaluation scenarios.

User	User 1	User 2	User 3	User 4	User 5
Precision	0.85	0.92	0.88	0.98	0.91
Recall	0.78	0.85	0.82	0.95	0.76

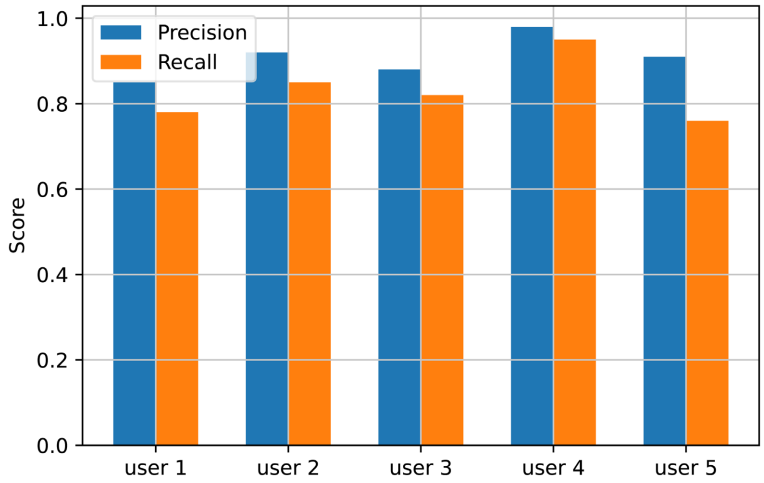

Figure 13. Precision and recall across evaluation scenarios.

the good document. Moreover, the model uses too much space to save some duplicate documents. Prakruthi *et al.* [15] suggest that for a given text, include more text to manipulate text comprehension. But the output shows all the real messages with the added message. The attacker can read all sensitive information. The authors of [25]-[31] have worked on the proposal to generate texts with a size that matters. However, they encountered problems of consistency and missing words in the generated text.

Just by focusing on these models, we can see that each of them presents at least one major problem in the case where we would like to disappoint the attacker. However, in the model, the segmentation introduced in Module 1 enables us to solve the problem linked to the size of the input text. Indeed, by segmenting a text into several segments, we can obtain inputs that can be easily manipulated by the model. One of the key problems in generating receptive text lies in the ability to hide information sensitive to the attacker in the output text. Unlike [12] and [13], which replace words in the input text to modify the context, our model searches for sensitive words and replaces them in such a way as to preserve the coherence of the text after replacement. What's more, the output text is still generated before being transmitted to the output. So, contrary to the literature, it is not possible to read or deduce sensitive information from the generated text.

4.7. Applicability

The multi-level deception model proposed in this article can be used in any information system, but it needs to be optimized for use in embedded tools such as sensors. However, the text generation model applied to it can only be used by systems handling textual data, as the model does not handle numerical data or formulas. To be effective in the medical field and produce good results, the model will require further training with more data sets, as the model is not effective when it comes to abbreviations. Ultimately, the model cannot be used in banking systems, information systems manipulating image or video data or geo-

graphic coordinates, such as drones, and the like.

5. Conclusions

Information systems face multiple attacks, usually targeting data. This can range from modification, deletion, or the unavailability of data. In this paper, we propose a multi-level deception system consisting of three states: the black state containing malicious users, the gray state containing uncertain users, and the white state containing legitimate users. Data manipulated by black-state users is generated using real data.

Text generation involves several steps, including the identification of sensitive words, the replacement of sensitive words, and text generation. We applied this to test data and users, and the results obtained validated the model.

However, there are a few problems with the model, such as the handling of synonymous words.

In the text generated above, the sensitive words have been replaced with alternative terms to obfuscate the original meaning. For example, "packet delivery" has been replaced with "secure item transfer," "Cleveland Clinic" has been replaced with "City Medical Center," "Hospital" has been replaced with "Facility Medical Facility," and "parcel" has been replaced with "package." These changes aim to deceive potential attackers by altering the context and making it harder to discern the true meaning of the text.

However, the word drone, which refers to UAV, has not been identified, nor has the word delivery.

Acknowledgements

Research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-21-1-0326. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

The work of the second author is partially supported by the EPSRC grant EP/V049038/1 and the Alan Turing Institute under the EPSRC grant EP/N510129/1.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Kouam Kamdem, I.G. and Nkenlifack, M.J.A. (2021) Data Security in Health Systems: Case of Cameroon. In: Arai, K., Ed., *Intelligent Computing*, Lecture Notes in Networks and Systems, Vol. 285, Springer, Cham, 48-57.

https://doi.org/10.1007/978-3-030-80129-8_4

[2] Mohammed, M.E. (2016) La sécurité dans les systèmes de santé. PhD Thesis, Diss, Université mohamed boudiaf' des sciences et de la technologie d'oran, Algerie.

[3] Katt, B. (2014) A Comprehensive Overview of Security Monitoring Solutions for E-Health Systems. 2014 *IEEE International Conference on Healthcare Informatics*, Verona, 15-17 September 2014, 364. https://doi.org/10.1109/ICHI.2014.59

[4] Akinyele, J., Pagano, M., Peterson, Z., Lehmann, C. and Aviel, D.R. (2011) Securing Electronic Medical Records Using Attribute-Based Encryption on Mobile Devices Report 2010/565. *Proceedings of the* 1*st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, Chicago, 17 October 2011, 75-86. https://doi.org/10.1145/2046614.2046628

[5] Anwar Ahmed, H., Charles, K. and Nandi, L. (2019) A Game-Theoretic Framework for Dynamic Cyber Deception in Internet of Battlefield Things. *Proceedings of the* 16*th EAI International Conference on Mobile and Ubiquitous Systems*: *Computing, Networking and Services*, 2-14 November 2019, 522-526. https://doi.org/10.1145/3360774.3368204

[6] Xing, J.C., Yang, M.L., Zhou, H.F., *Et Al.* (2019) Hiding and Trapping: A Deceptive Approach for Defending against Network Reconnaissance with Software-Defined Network. 2019 *IEEE* 38*th International Performance Computing and Communications Conference* (*IPCCC*), London, 29-31 October 2019, 1-8. https://doi.org/10.1109/IPCCC47392.2019.8958776

[7] Pawlick, J. and Zhu, Q.Y. (2021) Game Theory for Cyber Deception. Springer International Publishing, Berlin. https://doi.org/10.1007/978-3-030-66065-9

[8] Chen, Y.L., Wei, Y.J., Yu, Y.F., Xue, W. and Qin, X.Y. (2018) Cyber Security NLP: Machine-Based Text Analytics of National Cybersecurity Strategies. https://github.com/ychen463/cyber

[9] Egon, K., Stevanovic, M. and Pedersen, J.M. (2018) Detection of Malicious Domains through Lexical Analysis. 2018 *International Conference on Cyber Security and Protection of Digital Services* (*Cyber Security*), Glasgow, 11-12 June 2018, 1-5. https://doi.org/10.1109/CyberSecPODS.2018.8560665

[10] L'Huillier, G., *Et Al.* (2010) Latent Semantic Analysis and Keyword Extraction for Phishing Classification. 2010 *IEEE International Conference on Intelligence and Security Informatics*, Vancouver, 23-26 May 2010, 129-131. https://doi.org/10.1109/ISI.2010.5484762

[11] Mokhov, S.A., Paquet, J. and Debbabi, M. (2014) The Use of NLP Techniques in Static Code Analysis to Detect Weaknesses and Vulnerabilities. *Canadian Conference on Artificial Intelligence*, Montréal, 6-9 May 2014, 326-332. https://doi.org/10.1007/978-3-319-06483-3_33

[12] Priyavrat, B. (2014) Context Encryption Using Natural Language Processing. Amdocs.

[13] Kushwaha, A., Sharma, H.R. and Ambhaikar, A. (2018) Selective Encryption Using Natural Language Processing for Text Data in Mobile Ad Hoc Network. In: Vasant, P., Litvinchev, I. and Marmolejo-Saucedo, J.A., Eds., *Modeling, Simulation, and Optimization*, Springer, Cham, 15-26. https://doi.org/10.1007/978-3-319-70542-2_2

[14] Chakraborty, T., *Et Al.* (2019) A Fake Online Repository Generation Engine for Cyber Deception. *IEEE Transactions on Dependable and Secure Computing*, **18**, 518-533. https://doi.org/10.1109/TDSC.2019.2898661

[15] Karuna, P., *Et Al.* (2020) Fake Document Generation for Cyber Deception by Manipulating Text Comprehensibility. *IEEE Systems Journal*, **15**, 835-845.

https://doi.org/10.1109/JSYST.2020.2980177

[16] Egozi, G. and Verma, R. (2018) Phishing Email Detection Using Robust NLP Techniques. 2018 *IEEE International Conference on Data Mining Workshops* (*ICDMW*), Singapore, 17-20 November 2018, 7-12. https://doi.org/10.1109/ICDMW.2018.00009

[17] Siddiqui, S., *Et Al.* (2019) Ontology Driven Feature Engineering for Opinion Mining. *IEEE Access*, **7**, 67392-67401. https://doi.org/10.1109/ACCESS.2019.2918584

[18] Mokhov, S.A., Paquet, J. and Debbabi, M. (2014) The Use of NLP Techniques in Static Code Analysis to Detect Weaknesses and Vulnerabilities. *Canadian Conference on Artificial Intelligence*, Montréal, 6-9 May 2014, 326-332. https://doi.org/10.1007/978-3-319-06483-3_33

[19] Sarker, I.H., Furhad, M.H. and Nowrozy, R. (2021) Ai-Driven Cybersecurity: An Overview, Security Intelligence Modeling and Research Directions. *SN Computer Science*, **2**, Article No. 173. https://doi.org/10.1007/s42979-021-00557-0

[20] Cho, K., Van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014) Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation.

[21] Bahdanau, D., Cho, K. and Bengio, Y. (2014) Neural Machine Translation by Jointly Learning to Align and Translate.

[22] Tang, J.H., Zhao, T.C., Xiong, C.Y., Liang, X.D., Xing, E.P. and Hu, Z.T. (2019) Target-Guided Open-Domain Conversation. *Proceedings of the* 57*th Annual Meeting of the Association for Computational Linguistics*, Florence, July 2019, 5624-5634. https://doi.org/10.18653/v1/P19-1565

[23] Su, H., Shen, X.Y., Zhao, S.Q., Zhou, X., Hu, P.W., Zhong, R., Niu, C. and Zhou, J. (2020) Diversifying Dialogue Generation with Non-Conversational Text. *Proceedings of the* 58*th Annual Meeting of the Association for Computational Linguistics*, July 2020, 7087-7097. https://doi.org/10.18653/v1/2020.acl-main.634

[24] Shen, L., Zhan, H.L., Shen, X., Song, Y.H. and Zhao, X.F. (2021) Text Is Not Enough: Integrating Visual Impressions into Open-Domain Dialogue Generation. *Proceedings of the* 29*th ACM International Conference on Multimedia*, 20-24 October 2021, 4287-4296. https://doi.org/10.1145/3474085.3475568

[25] Liu, Y. and Lapata, M. (2018) Learning Structured Text Representations. *Transactions of the Association for Computational Linguistics*, **6**, 63-75. https://doi.org/10.1162/tacl_a_00005

[26] Guo, J.X., Lu, S.D., Cai, H., Zhang, W.N., Yu, Y. and Wang, J. (2018) Long Text Generation via Adversarial Training with Leaked Information. *Proceedings of the AAAI Conference on Artificial Intelligence*, **32**, 5141-5148. https://doi.org/10.1609/aaai.v32i1.11957

[27] Li, Z., Jiang, X., Shang, L. and Li, H. (2017) Paraphrase Generation with Deep Reinforcement Learning. https://doi.org/10.48550/arXiv.1711.00279

[28] Noraset, T., Demeter, D. and Downey, D. (2018) Controlling Global Statistics in Recurrent Neural Network Text Generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, **32**, 5333-5341. https://doi.org/10.1609/aaai.v32i1.11993

[29] Graves, A. (2013) Generating Sequences with Recurrent Neural Networks.

[30] Le, Q. and Mikolov, T. (2014) Distributed Representations of Sentences and Documents. *International Conference on Machine Learning*, Beijing, 21-26 June 2014, 1188-1196.

[31] Li, J.W., Luong, M.-T. and Jurafsky, D. (2015) A Hierarchical Neural Autoencoder

for Paragraphs and Documents.

[32] Agrawal, R., Chakraborty, S., Gollapudi, S., Kannan, A. and Kenthapadi, K. (2012) Empowering Authors to Diagnose Comprehension Burden in Textbooks. *Proceedings of the* 18*th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Beijing, 12-16 August 2012, 967-975. https://doi.org/10.1145/2339530.2339682

[33] Opsahl, T., Agneessens, F. and Skvoretz, J. (2010) Node Centrality in Weighted Networks: Generalizing Degree and Shortest Paths. *The Social Network*, **32**, 245-251. https://doi.org/10.1016/j.socnet.2010.03.006

[34] Mihalcea, R., *Et Al*. (2006) Corpus-Based and Knowledge-Based Measures of Text Semantic Similarity. *Proceedings of the* 21*st National Conference on Artificial Intelligence*, **1**, 775-780.

[35] Saviour, M.P.A. and Samiappan, D. (2023) IPFS Based Storage Authentication and Access Control Model with Optimization Enabled Deep Learning for Intrusion Detection. *Advances in Engineering Software*, **176**, Article ID: 103369. https://doi.org/10.1016/j.advengsoft.2022.103369

[36] Bisaillon, C. (2019) Fake and Real News Dataset. https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset?select=true.csv

[37] Mamlin, B. (2022) Demo Data. https://wiki.openmrs.org/display/res/demo+data