Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 18 No. 10 (2024) | OPEN ACCESS

https://doi.org/10.3991/ijim.v18i10.45725

PAPER

NDLP Phishing: A Fine-Tuned Application to Detect Phishing Attacks Based on Natural Language Processing and Deep Learning

Eduardo Benavides-Astudillo[1,2](✉), Walter Fuertes[2], Sandra Sanchez-Gordon[1], Daniel Nuñez-Agurto[2]

[1]Department of Informatics and Computer Science, Escuela Politécnica Nacional, Quito, Ecuador

[2]Department of Computer Sciences, Universidad de las Fuerzas Armadas ESPE, Sangolquí, Ecuador

diego.benavides@epn.edu.ec

ABSTRACT

Phishing is a cyberattack that aims to deceive and harm users socially or economically. The most elaborate method to carry out this type of attack is through phishing web pages. For an untrained eye, it is not easy to distinguish whether a page is phishing. Different solutions combat this type of attack, such as those using deep learning (DL). Still, they need to be more aligned with the body text of web pages, taking into account their linguistic characteristics, or they will only exist as a model without providing practical application. This study aims to develop a lightweight tool, an extension for installation in the Google Chrome web browser that enables the detection of phishing attacks using DL and natural language processing (NLP) techniques. This proposed tool is NDLP Phishing (NDLP is a combination of the acronyms NLP and DL). First, we selected and adjusted the hyperparameters of BiGRU layers, dropout, batch Size, epochs, BiGRU neurons, and GloVe dimension of a BiGRU detection model based on DL and NLP. Second, an extension was developed for Google Chrome based on the fine-tuned model. The results of our experiments show a set of optimal hyperparameters to train the model. Subsequently, we applied these hyperparameters and achieved a mean accuracy of 98.55%. The code for the algorithms that generated the prediction model and the code for the Google Chrome extension are shared on GitHub.

KEYWORDS

phishing, deep learning (DL), natural language processing (NLP), application, chrome extension, BiGRU, fine-tuning

1 INTRODUCTION

People have become dependent on electronic devices and Internet services. The use of devices has grown exponentially since the COVID-19 pandemic. Due to mobility restrictions, online transactions multiplied during the pandemic for

Benavides-Astudillo, E., Fuertes, W., Sanchez-Gordon, S., Nuñez-Agurto, D. (2024). NDLP Phishing: A Fine-Tuned Application to Detect Phishing Attacks Based on Natural Language Processing and Deep Learning. *International Journal of Interactive Mobile Technologies (iJIM)*, 18(10), pp. 173–190. https://doi.org/10.3991/ijim.v18i10.45725

Article submitted 2023-10-09. Revision uploaded 2024-01-08. Final acceptance 2024-01-26.

purchasing food, clothing, and other items, as well as paying for services. However, as the number of users utilizing Internet platforms for transactions has increased, so has the number of attackers attempting to obtain confidential information from users to commit fraud against them [1]. Computer systems may have software and hardware that prevent computer attacks; however, the chain is only as strong as its weakest link. This weak link is often the user themselves, who are tricked using social engineering techniques into revealing confidential information.

Social engineering is a practice that enables attackers to acquire sensitive or confidential information from a user of a system or organization by exploiting specific human characteristics [2], such as behavior [3], or their susceptibility to attacks [4]. The primary type of social engineering cyberattack is phishing, in which the attacker utilizes electronic devices to deceive users through emails and web pages, among other methods. Figure 1 illustrates the impacted business sectors in 2021 [5]. The primary technique to detect if a web page is phishing is to check if the page is on a blacklist [6]. However, the blacklisting procedure fails when a new phishing page appears. Several methods, such as machine learning, can be used to combat this new attack involving phishing web pages [7]. Machine learning techniques, such as deep learning (DL), analyze new text using the characteristics of previously trained DL algorithms.

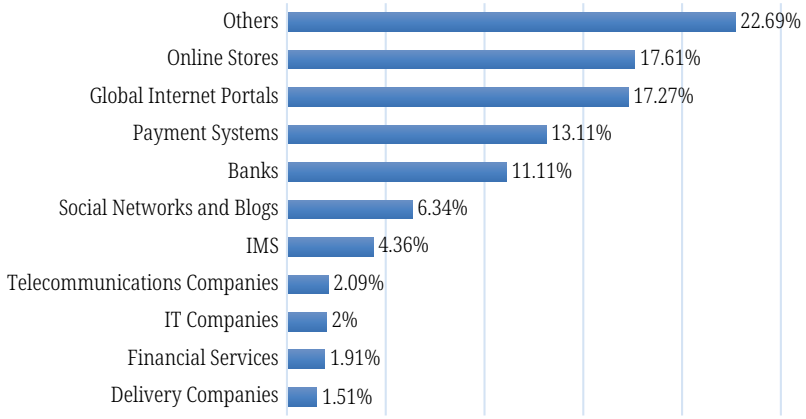

Fig. 1. Percentage of phishing attacks by type of organization

Currently, most web page solutions focus on applying DL algorithms to the URLs of these pages, with only a few solutions targeting the text within the web pages [8]. These solutions fail to leverage the rich semantics and syntax present in the analyzed text. Some researchers leverage semantics and syntax through natural language processing (NLP) to counter phishing attacks [7]. Still, they target the text in phishing emails rather than phishing web pages. Furthermore, solutions that apply NLP to the text of phishing emails result in excessive computational processing. In contrast, the literature review did not identify any solution that provides an application to alert users when a web page has a high likelihood of phishing based on its textual content.

Due to the importance of obtaining a solution to detect this type of attack on web pages, the main goal of this study is to offer a refined and lightweight application that alerts the user if a page intended to be opened has a high probability of being phishing. The following secondary objectives must be carried out to meet this main goal: 1) Develop a refined DL model that captures the semantic and syntactic characteristics of text extracted from web pages; and 2) Create a browser extension that promptly and accurately alerts users to potential phishing web pages.

The hyper-parameters were turned using a manual search method, evaluating BiGRU layers, dropout, batch size, epochs, BiGRU neurons, and GloVe dimension. The traditional cascade method was used to develop the application, consisting of requirements analysis, design, coding and testing, and implementation.

The following are the main contributions of this study:

- Provide a dataset with clean information, free from the words and characters present in the HTML code of the web pages.
- Share the complete code for tuning and training the detection algorithm using BiGRU, as well as the trained algorithm file.
- Display the appropriate hyperparameters for model tuning.
- Share the complete code required to create the extension that can detect a new phishing page.
- Share an extension that can be installed and tested in the Google Chrome browser.

In the next section, the related works will be reviewed. Section 3 will present the methodology and tools used. Section 4 will show the results. Section 5 will discuss the study. Finally, in Section 6, the conclusions and future work will be presented.

2 RELATED WORK

This study aims to develop an application for detecting phishing attacks using NLP and DL based on [9]. In addition, the analysis should be conducted on the content of the phishing web pages. That is why the search for related published work has been conducted on the Web of Science, Scopus, IEEE, and ACM scientific databases.

The search was performed with the following search string: [Phishing and (NLP) and (DL)]. Initially, this search yielded 62 research articles. Duplicates, papers not in English, papers lacking a model or application, or failing to demonstrate a solution combining NLP and DL were excluded. Finally, we selected the documents presented in Table 1.

Table 1. Related work

Id	Title
[10]	A deep learning model with hierarchical LSTMs and supervised attention for anti-phishing.
[11]	Phishing Detection Using Deep Learning Attention Techniques.
[12]	Machine Learning and Deep Learning for Phishing Email Classification Using One-Hot Encoding.
[13]	Spam email detection using deep learning techniques.
[14]	Applying machine learning and natural language processing to detect phishing emails.
[15]	Intelligent Deep Learning Based Cybersecurity Phishing Email Detection and Classification.
[16]	Phish Responder: A Hybrid Machine Learning Approach to Detect Phishing and Spam Emails.
[17]	Lightweight URL-based phishing detection using natural language processing transformers for mobile devices
[18]	Phishing Email Detection Model Using Deep Learning
[19]	Cyberattack Detection in Social Network Messages Based on Convolutional Neural Networks and NLP Techniques
[9]	A Phishing-Attack-Detection Model Using Natural Language Processing and Deep Learning.

The study [10] proposes a model to detect phishing attacks by applying the LSTM algorithm to the header and body text extracted from emails. For this purpose, they establish a hierarchical structure, with sentences at the upper level and words at the lower level. The ratio between the words appearing on a legitimate web page, (legitimate rank (word)) and a phishing web page, (phishing rank (word)) is calculated to analyze the text and each term. Finally, a precision of 98.16%, recall of 95.96%, and F1 score of 97.05% are obtained. Neither in this study nor in any of the subsequent studies was an application developed.

In article [11], the email data corpus is first preprocessed. Two models based on attention mechanisms, BERT and XLNet, analyze this corpus. XLNet and BERT were utilized to comprehend the context of incoming emails and offer effective and accurate filtering. The final accuracy of the learned BERT was 99.66%, while XLNet's achieved an absolute accuracy of 99.62%.

The research [12] is a comparative study of ML and DL algorithms. It explores one-hot encoding and word embedding for preprocessing email body data. This study applies Naive Bayes, SVM, Decision Tree, LSTM, and CNN algorithms in combination with Word Embedding using one-hot encoding. The study also varies the hyperparameters of each algorithm. The DL algorithms yielded superior results compared to the ML algorithms, but the computational time was excessively high. The best performing algorithm was the combination of CNN with one-hot embedding, achieving an accuracy of 95.70%; however, CNN with Word Embedding achieved 96.34% accuracy.

The study [13] examines the text found in spam emails using BERT, a model that utilizes layers of attention to grasp the context of the text. In this study, it is concluded that BERT contextual word embedding's yield better results compared Keras word embedding. The text preprocessed by BERT is fed into a BiLSTM algorithm, which achieves 96.43% accuracy and a 96% F1 score.

The study [14] uses body text to improve phishing detection accuracy. The email body is analyzed using NLP, DL algorithms, and GCN to determine if an email is phishing or legitimate (ham). After cleansing the dataset, the next step is to construct a single large graph from the entire email corpus, with the words and emails used as nodes. The edges connecting the word nodes depend on the co-occurrence information between two words. The boundaries between a word and an email are determined by the word frequency and the email frequency of the word. The classifier proposed in this research performs well on a balanced and labeled dataset. The accuracy exceeds 98%, and all evaluation metrics surpass those of existing detection models.

In the article [15], an analysis is made of the body of the emails using n-grams. This NLP technique captures the relationships between the words in the email. The clean data is then passed through the GRU algorithm to detect whether an email is phishing. A unique aspect of this work is the optimization of the GRU algorithm using the Cuckoo Search algorithm. The application of these techniques together resulted in an accuracy of 99.72%.

Phish Responder [16] is a solution that utilizes DL and NLP to detect phishing and spam email attacks. For this, an NLP analysis of the body of the emails was re-performed using only tokenization and TF-IDF, without employing techniques such as stemming, lemmatization, and POS tagging. For the LSTM model applied to text-based datasets, 99 % accuracy rate was achieved.

Other techniques do not necessarily require NLP preprocessing with a dictionary, such as GloVe, before being input into a DL algorithm. For example, BERT and ELECTRA are combined with artificial neural networks for phishing attack detection [17]. Unlike our proposal, this work is oriented towards applying algorithms to the URLs and HTML code of web pages.

Since the primary vector of phishing attacks is email, many studies already utilize DL and NLP to detect phishing attacks. We have included the study conducted by [18] in our related work because it employs techniques and methods similar to those used in our study. In our work, the analysis is performed on web pages and not on emails.

While email phishing and webpage phishing attacks are more prevalent, attacks through social network messages are also on the rise due to the expansion of networks such as WhatsApp, Facebook, Twitter, and Instagram, among others. The authors of the article [19] propose using NLP and convolutional neural networks to detect social network attacks and classify the type of cyberattack (malware, phishing, spam, or boot).

The authors [9] characterized the detection of phishing attacks based on the text extracted from web pages. For this, to begine with, the text of the web page is captured. For this, first, a capture of the web page is made. The extracted text is preprocessed, and Keras embedding is used with the GloVe dictionary to capture the semantic and syntactic characteristics of the text. Subsequently, the preprocessed data is input into a BiGRU algorithm to predict whether a page is phishing, achieving up to 97.39% mean accuracy.

The publications [10] to [19] in Table 1 propose models for detecting phishing attacks using NLP and DL. However, they only focus on features of phishing emails or social networks, neglecting phishing web pages. Thus, the only study in Table 1 that identifies phishing web pages is [9]. None of the reviewed papers present an application capable of real-time detection to determine if a webpage is phishing or not when attempting to access it. See Table 2 for a summary of the revised characteristics of the related work.

Table 2. Techniques used in related articles and its scope

Id	NLP	DL	Phishing Web Page	Application
[10]	Yes	Yes	No	No
[11]	Yes	Yes	No	No
[12]	Yes	Yes	No	No
[13]	Yes	Yes	No	No
[14]	Yes	Yes	No	No
[15]	Yes	Yes	No	No
[16]	Yes	Yes	No	No
[17]	No	Yes	Yes	No
[18]	Yes	Yes	No	No
[19]	Yes	Yes	No	No
[9]	Yes	Yes	Yes	No
Our proposal NDLP	Yes	Yes	Yes	Yes

3 METHODOLOGY AND TOOLS

This section is structured into two parts. In the first part of the methodology, the steps to follow to achieve two objectives are outlined: first, developing a refined

algorithm to detect attacks using DL and NLP, and second, the procedures to implement and install a Chrome browser extension for the refined algorithm. The following subsection presents the hardware and software tools used to implement this project.

3.1 Methodology

The phases and steps conducted in this study are detailed below.

Phase I. Determine the optimal hyper-parameters to fine-tune the NLP and DL phishing attack detection application. There are three main strategies to determine the hyperparameters of machine learning algorithms: grid search, random search, and manual search. For our study, we followed the manual search method [20]. Although grid search and random search can achieve higher hyperparameter resolution, the computational cost is very high. Furthermore, according to [21], most hyperparameters that are analyzed will not significantly increase the accuracy of the analyzed algorithm. Therefore, we have adopted the manual hyperparameter search approach based on expert knowledge, intuition, and experience. The steps followed in the manual search for hyperparameters are [22]:

1. Understanding the model and hyperparameters
2. Definition of the search space
3. Data division
4. Training, evaluation, analysis of results, and manual adjustment
5. Cross validation
6. Test on the test set
7. Documentation

1. *Understanding the model and hyperparameters*
   One of our research objectives is to develop an algorithm that is fast, efficient, and highly accurate in detecting phishing attacks [23]. We previously selected the BiGRU algorithm utilizing embeddings from the GloVe dictionary [9] [24].
2. *Definition of the search space*
   To enhance the performance of the model presented in [9] and develop a lightweight application without compromising its accuracy, it was essential to fine-tune hyperparameters in the preprocessing stage and the design of the DL algorithm [21]. For this purpose, the algorithms are executed with the hyperparameters, and their various values are shown in Table 3.

Table 3. Hyper-parameters to evaluate

Hyper-Parameter	Values			
Number of words	100	200	500	1000
BiGRU Neurons	32	64	128	256
Dimension of the GloVe embedding dictionary		50	100	200
Number of epochs		5	10	20
Batch size	32	64	128	256
Dropout value		0.1	0.5	0.7
BiGRU layers			1	2

3. *Data division*
   Out of 9761 records, the dataset was divided into 80% for training and 20% for testing. This split was done implicitly using K-fold = 5 with K-fold cross-validation.
4. *Training and evaluation, analysis of results, and manual adjustment.*
   In this step, the training and evaluation are carried out with each of the hyperparameters in the search space. The assessment of the results of the algorithm's execution is conducted, followed by manual adjustments to each value of every hyperparameter. Figure 2 illustrates the sequence of analysis for each hyperparameter.

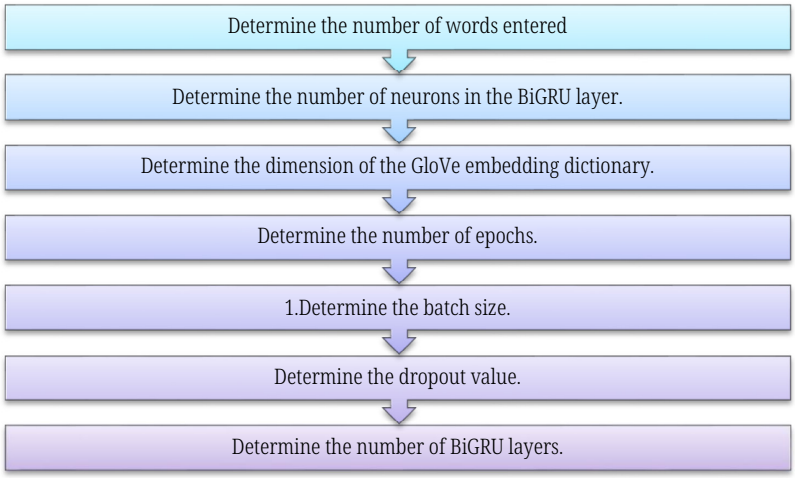

Fig. 2. Steps to determine the optimal hyper-parameters of NDLP Phishing

5. *Cross validation*
   A K-fold cross-validation algorithm [25] with five folds is used for each model run. We utilized K-fold cross-validation due to the imbalance in the data used for training, testing, and tuning our BiGRU algorithm. The metric used to compare the algorithms' performance was mean accuracy.
6. *Test on the test set*
   Finally, we executed the model with the optimal hyperparameters obtained in the previous steps and determined the mean accuracy achieved. It aims to emphasize average accuracy and the duration required to train and test the algorithm.
7. *Documentation*
   The effects of the experiments are presented in the results section.

Phase II. Make an application based on the fine-tuned attack detection model using NLP and DL. Once we have obtained a lightweight algorithm with high accuracy in the previous phase, our next objective is to develop an application based on that algorithm. This application can be installed as an extension in Chrome, the most widely used web browser. We followed the following steps to carry out the application:

1. Requirements analysis
2. Design
3. Coding, testing
4. Implementation

1. *Requirements analysis*
   The developed application should consider the following requirements:
   - It would be best if you took advantage of the characteristics of NLP.
   - It must be able to capture the text of analyzed web pages.
   - It must preprocess and provide clean text to a DL algorithm.
   - It must have an accuracy rate of over 95%.
   - It must be fast and consume minimal computational resources.
   - It must be able to run in the primary web browser.
2. *Design of the application*
   First, users open a web page. If they want to determine whether a page is a phishing website, they can click on the extension installed in the body of the web browser. The application captures the text on the web page at that moment. This text is then pre-processed with NLP, obtaining data without stop words, POS-tagged, lemmatized, pad-sequenced, and tokenized. Subsequently, this pre-processed data is sent to our pre-trained and fine-tuned algorithm. Finally, the end user is informed whether the page they are trying to open is phishing or legitimate (ham). See the architectural design in See Figure 3.

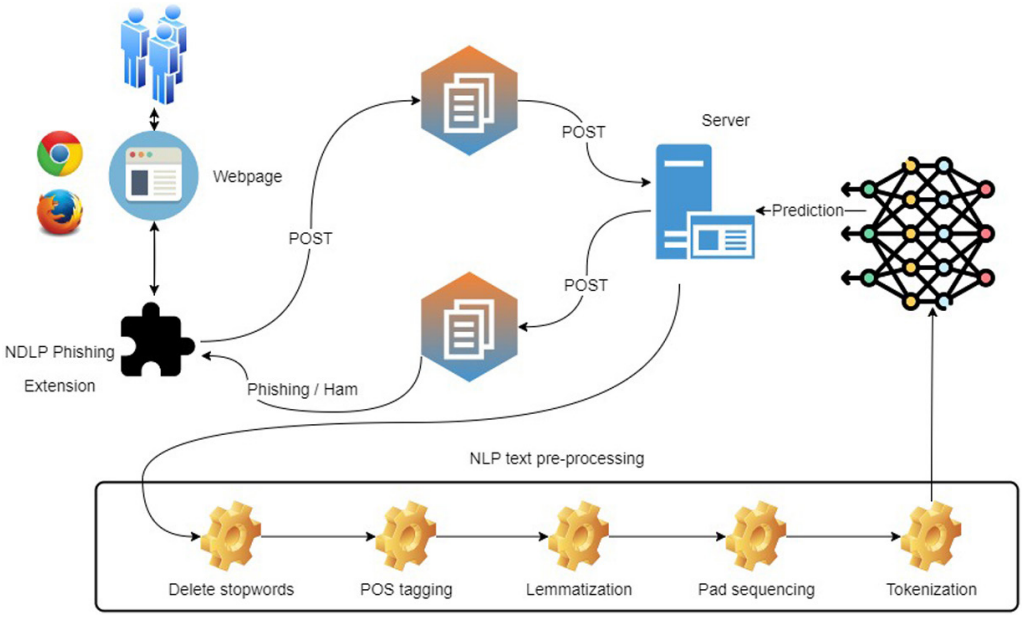

Fig. 3. Design of the proposed application NDLP phishing

3. *Coding and testing*
   The coding and testing procedures are detailed in the results section.
4. *Implementation*
   The implementation procedure is detailed in the results section.

3.2 Tools

Hardware environment tuning the model's hyperparameters required a high-processing server and a laptop with standard specifications to run the application. Table 4 shows the specifications of the two pieces of equipment used.

Table 4. Hardware features

Component	Server Features	Client Features
Processor	AMD Ryzen Threadrippe 2920X	Intel Core I7, 10th generation
RAM	16 GB Crucial Ballistix DDR4-3000	8GB
Video card	16 GB Phantom Gaming X Radeon VI	NVidia GeForce MX230
SSD	500 GB Crucial SSD M.2 NVMe	256 GB
HD	3 TB Western Digital HDD Purple	Not apply
Main board	ASUS ROG Zenith Extreme Alpha	Not apply

Software environments. For the execution of the tuning process, a RIG server was utilized with Python 3.5.2 on Jupyter Notebook 6.0.2, along with the libraries: Keras, NLTK, NumPy, pandas, requests, scikit-learn, and TensorFlow.

Dataset. The dataset used for the experiments was obtained from the Phishload page [26]. The dataset consists of 10,488 rows. Since the original data contained many unusable characters for our algorithm analysis, it was preprocessed before being used in our experiment. Finally, only 9761 valid rows were obtained, of which 8589 are phishing rows and 1172 are from ham. The data, filtered and cleaned by us, can be downloaded from our website.

4 RESULTS

This section is composed of two different parts. The first part presents the results obtained from model tuning and identifies the optimal hyperparameters for this process. The second part elaborates on the steps to create the extension based on the optimized algorithm and demonstrates its functionality.

4.1 Determine the optimal hyper-parameters to fine-tune the NLP and DL phishing attack detection application

The entire tuning experiment was performed in Jupyter Notebook and can be reproduced with the code shared in the following GitHub link: https://github.com/debenavides/NDLP-hyper-parameter-tuning/.

In the comments of the shared code above, each hyperparameter and its corresponding values are displayed.

The tuning was performed on the model proposed in the study [9] because it aligns with the assumptions of predicting a phishing attack using NLP and DL on the text extracted from the body of the web pages. Before starting, it should be mentioned that in the study [9], several aspects were analyzed. These details will serve as the basis for our experiment. It was determined in [9] that the algorithm that yielded the best results in this type of problem was BiGRU, surpassing the average accuracy of LSTM, BiLSTM, and GRU.

The aim of this work is not only to provide a highly accurate model but also to ensure it is lightweight enough for the application to be installed on a regular computer or mobile device. So, our tuning aims to provide a lightweight application without compromising accuracy.

Determine the number of words entered. The algorithms were tested in the article [9] using four text inputs of 100, 200, 500, and 1000 words. It was conducted that the optimal number of words to enter was 200.

Determine the number of neurons in the BiGRU layer. Based on the experiment conducted by [9], where 128 neurons were utilized in the BiGRU layer, our current study assesses the mean accuracy achieved with 32, 64, 128, and 256 neurons. The results obtained can be seen in Table 5.

Table 5. BiGRU neurons

BiGRU Neurons	K-Fold Mean Accuracy
32	0.9680
64	0.9713
128	0.9737
256	0.9757

Table 5 shows that the K-fold mean accuracy obtained with 32 neurons significantly differs from that obtained with 64, 128, and 256 neurons. This difference occurs because the mean accuracy curve tends to stabilize as the number of neurons increases. On the other hand, with 256 neurons, the mean accuracy increases slightly, but at a high computational cost. As a result, 32 and 256 neurons are excluded from further experiments, and only 64 and 128 neurons are considered.

Determine the dimensions of the GloVe embedding dictionary. The next tuning step involved determining the optimal dimension of the GloVe dictionary [27], taking into account the available GloVe dimensions of 50, 100, 200, and 300. The results of the tests with vectors of 50, 100, and 200 dimensions are shown in Table 6.

Table 6. GloVe dimensions

BiGRU Neurons	GloVe Dimension	K-Fold Mean Accuracy
64	50	0.9685
64	100	0.9703
64	200	0.9749
128	50	0.9681
128	100	0.9736
128	200	0.9758

As shown in Table 5, the algorithm's performance was evaluated with 50, 100, and 200 dimensions of the GloVe dictionary (300 dimensions were not necessary), both with 64 and 128 BiGRU neurons. It is observed that the value obtained with 100 and 200 GloVe dimensions does not vary significantly from that obtained with a 50-dimensional vector. This may be because the model has little complexity. Since a lightweight model is required, a size of 50 GloVe dimensions is adopted as the optimal value.

Determine the number of epochs. So far, ten epochs have been used for testing. To enhance the mean accuracy, the model is evaluated with 5, 10, and 20 epochs. The results of the model execution with varying epochs are presented in Table 7.

Table 7. Epochs

Epochs	BiGRU Neurons	GloVe Dimension	Elapsed Time (Min)	Mean Accuracy
5	64	50	0:12	0.9905
10	64	50	0:23	0.9896
20	64	50	0:46	0.9844

Table 7 shows that the accuracy decreases as the number of epochs increases, possibly due to overfitting. Even with five epochs, a mean accuracy of 0.9905 is achieved. Five epochs are adopted for the model.

Determine the batch size. A larger batch size can lead to greater generalization; however, this requires more memory capacity and longer training time. Therefore, batch size values of 32, 64, 128, and 256 are analyzed. See the results in Table 8.

Table 8. Batch size

Batch Size	Epochs	BiGRU Neurons	GloVe Dimension	Elapsed Time (Min)	Mean Accuracy
32	5	64	50	0:51	0.9851
64	5	64	50	0:26	0.9896
128	5	64	50	0:13	0.9844
256	5	64	50	0:09	0.9611

Table 8 shows that the mean accuracy value obtained for a batch size of 256 is considerably lower than the other values. In addition, it has been determined that the value obtained with a batch size of 64 offers better results than the other batch size.

Determine the dropout value. It is essential to determine an appropriate dropout rate to reduce overfitting without compromising performance or speed [28]. Therefore, an assessment is conducted at low, medium, and high dropout values. The values obtained with dropouts of 0.1, 0.5, and 0.7 can be seen in Table 9.

Table 9. Dropout size

Dropout	Batch Size	Epochs	BiGRU Neurons	GloVe Dimension	Elapsed Time (Min)	Mean Accuracy
0.1	64	5	64	50	0:12	0.9635
0.5	64	5	64	50	0:12	0.9454
0.7	64	5	64	50	0:11	0.9384

Table 9 shows that the elapsed time in the execution with the three dropout values analyzed practically remains unchanged; however, the mean accuracy obtained with the 0.1 dropout is better than that obtained with the other two dropout values.

Determine the number of BiGRU layers. We added an additional layer to test for improved accuracy after determining the optimal hyperparameters for a single BiGRU layer. The results for one and two layers of BiGRU are presented in Table 10.

Table 10. BiGRU layers

BiGRU Layers	Dropout	Batch Size	Epochs	BiGRU Neurons	GloVe Dimension	Elapsed Time (Min)	Mean Accuracy
1	0.1	64	5	64	50	0:11	0.9635
2	0.1	64	5	64	50	0:19	0.9454

Based on the results presented in Table 10, we observed that increasing the number of layers in our DL model did not enhance the mean accuracy. Furthermore, the training time increased by seven minutes compared to the previous model. We concluded that adding more layers to the model is unnecessary and may lead to increased complexity, longer processing times, and decreased accuracy. This decrease may be because the data processed by our model is not necessarily very complex.

After tuning the model, Table 11 displays the values of the hyperparameters obtained in our proposal.

Table 11. Tuned hyper-parameters for the NDLP application

	BiGRU Layers	Dropout	Batch Size	Epochs	BiGRU Neurons	GloVe Dimension
Our tuned model NDLP	1	0.1	64	5	64	50

4.2 Make an application based on the fine-tuned attack detection model using NLP and DL

Coding and testing. The code files used to develop the Google extension can be found at the following link on the GitHub site: https://github.com/debenavides/NDLP-Phishing/.

The following describes the content of each file in the shared repository:

Preprocessed_dataset.csv. This dataset contains 9761 valid rows, with 8589 rows classified as phishing and 1172 as ham. This dataset contains three columns: "context," which includes the text obtained from each web page, and the categorical columns "phishing" and "ham," which indicate whether a web page is a phishing site or not. This data is ready to be entered into the NLP and DL algorithms.

NDLP phishing tuned Model.ipynb. This file contains the model that has already been tuned with all the parameters optimized. In this model, variations of all the hyperparameters have been analyzed. The experiment was performed on the data contained in the Preprocessed_dataset.csv file. Finally, the trained model NDLP_model.h5 is generated and saved in the model.

NDLP_model.h5. This file is the saved model of the NDLP Phishing Tuned Model.ipynb, which is then sent to production to detect a new phishing page.

One webpage analysis.ipynb. This code contains two parts: in the first part, the HTML text obtained from a new web page is cleaned, and in the second part, a prediction is made by calling the trained model *NDLP_model.h5*.

Ext_NDLP.zip. This folder contains all the Chrome extension files. Before running this extension, it must be added to the Chrome browser. This folder includes the code for pre-processing the web page and the fine-tuned *NDLP_model* for analyzing the pre-processed text.

How the Google Chrome extension works. The browser extension is configured using the manifest.json file, which serves as a configuration map defining essential properties such as name, version, and required permissions. The popup interface, popup.html, provides a simple user interface with a button that, when activated by the popup.js script, sends a message to the content script content.js to start capturing text on the current web page. The content.js script runs in the web page context and utilizes the browser API to capture the page text document.body.innerText.

Then, the fetch function sends an HTTP request to the Flask server with the captured text as JSON data.

Flask is used on the server side. Flask acts as a server and an environment to load and run trained DL models. The deploy.py script configures a Flask server and defines routes, including the POST/processText path. When a request is received on this route, it invokes the process_text function, which performs processing operations on the text. The process_text function is responsible for processing the received text and performing specific operations such as cleaning, tokenization, and selecting 200 words. Once the above process is complete, it uses a phishing DL model NDLP Phishing.h5 to make the prediction.

Once the Flask server has processed the text and made the prediction, it sends the result back to the client. The content.js script on the client side receives the server's response, which includes the prediction result. Based on the prediction result, the script displays an alert to the user in the browser interface. This comprehensive interaction flow between the client and the server ensures that the browser extension can capture, process, and evaluate the text of the current web page, enabling a response to potential phishing attempts.

Implementation. To use this application, first open the web page to be analyzed in the Google Chrome browser. Then, click on the extension icon as shown in Figure 4. Subsequently, the interface depicted in Figure 5 will appear.

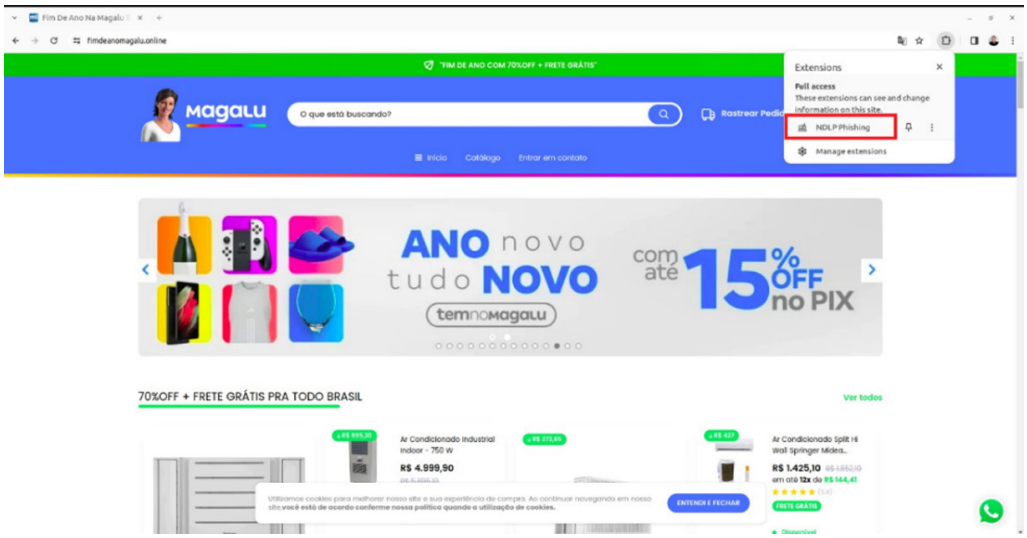

Fig. 4. NDLP extension icon

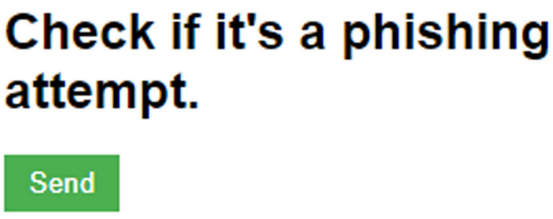

Fig. 5. NDLP Message to check if it is a phishing attack attempt

When the user clicks on the interface in Figure 5, the NDLP program is executed. It analyzes the text of the web page and reports in Figures 6 and 7 the percentage probability that the page under analysis is phishing or not.

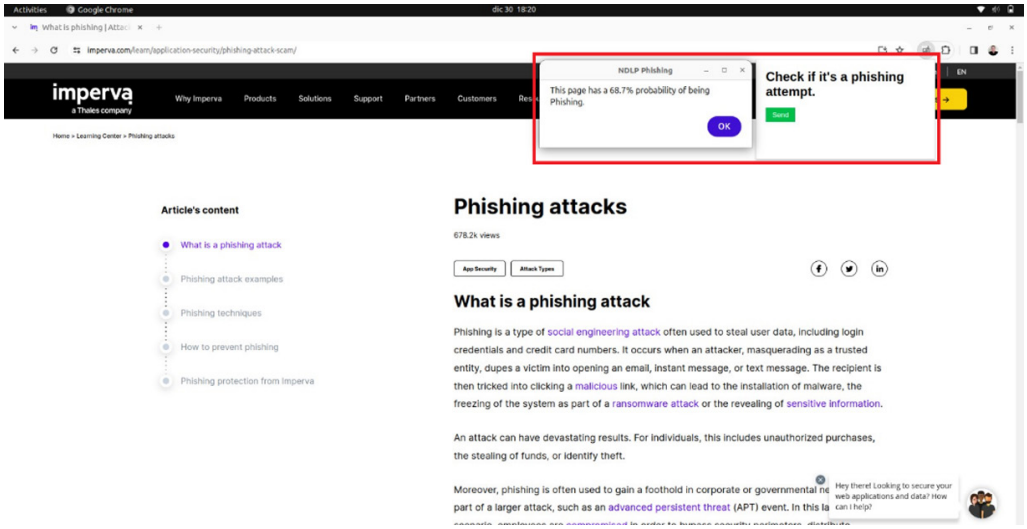

Fig. 6. NDLP execution

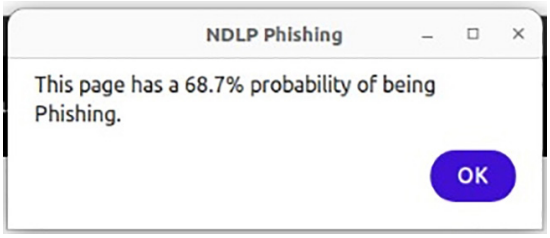

Fig. 7. Message showing the percentage of a webpage being phishing or not

5 DISCUSSION

Although articles [10–19] and [9] address the problem of detecting phishing attacks through DL and NLP, none of them develops an application as proposed in our study. In addition, articles that detect phishing attacks using DL and NLP on web pages were also searched, but only article [9] was found. In contrast, the others were only focused on phishing emails or social networks, leaving much to be investigated in the content of web pages. For this reason, the model proposed in the article [9] was taken as the starting point of our study. It detects phishing attacks using DL and NLP on the text found in web pages.

To conduct a comparative analysis between the initial algorithm of the [9]-article model and our proposed optimized algorithm, NDLP, we replicated the experiment from [9] on a personal computer. Subsequently, we executed the experiment using our proposed optimized algorithm (https://github.com/debenavides/NDLP_Comparative). Table 12 displays the configuration of the hyperparameters for [9] and our proposal.

Table 12. Tuned hyper-parameters for the NDLP application

	BiGRU Layers	Dropout	Batch Size	Epochs	BiGRU Neurons	GloVe Dimension	Elapsed Time	Mean Accuracy
[9]	1	0.1	128	10	128	100	5h 32min	0.9682
Our tuned model NDLP	1	0.1	64	5	64	50	0h 26min	0.9855

After running the experiment, it can be seen in Table 12 that the model proposed by [9] achieved a mean accuracy of 0.9682. In comparison, our tuned proposal achieved a mean accuracy of 0.9855, surpassing the algorithm proposed in [9] by 1.7%, with in a processing time 12 times shorter.

Figure 8 illustrates the performance of two algorithms: one untuned with 10 epochs and the other tuned with five epochs. Also, in Figure 8, it can be observed that the non-tuned algorithm, after epoch seven, starts to decline, dropping even below 99.86%, while the tuned algorithm reaches epoch five up to 99.90%.

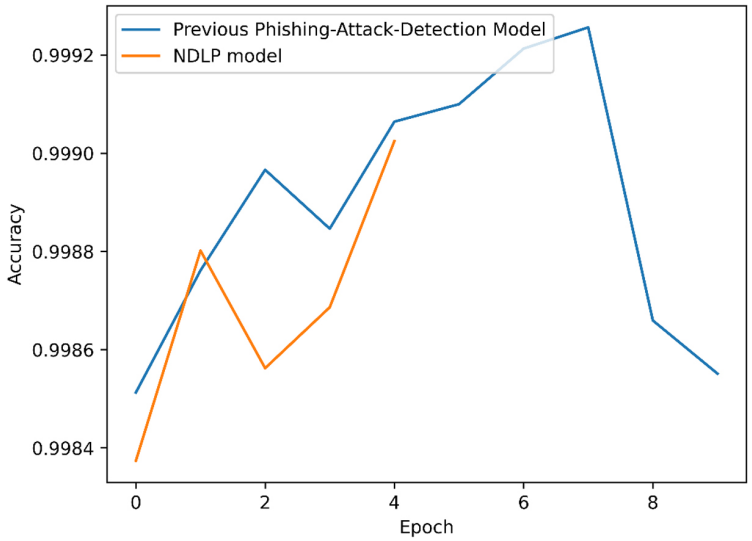

Fig. 8. Comparison between the accuracy of the algorithms of the proposed model in [9] and the refined NDLP

It can be seen in Table 11 that after conducting our experiments, the hyperparameters that remained unchanged from those initially used in the model [9] were BiGRU layers and dropout. Table 11 also indicates a reduction in the hyperparameters such as batch size, epochs, BiGRU neurons, and GloVe dimensions, which were set at 0.9855.

Thus, it can be determined that the values obtained with the tuned hyperparameters only decrease the mean accuracy of the proposal from [9] to our proposal by 1%. On the other hand, the training time with our proposal is four times faster than the proposal [9]. Therefore, we have accepted the hyperparameters presented in Table 10 for our model as the basis for the application developed as part of this study.

With our proposal, we aim to address the research gap by developing an application that can detect phishing attacks on web pages using cutting-edge technologies such as DL and NLP-GloVe. In addition, our application can only capture the text content of any web page. It preprocesses the text with NLP-GloVe and analyzes it with DL, achieving a mean accuracy of over 99% in predicting whether a page is not phishing. The prediction is made using the NDLP Phishing.h5 model that was previously obtained. This application was installed as a Google Chrome extension and is user-friendly.

6 CONCLUSIONS

The primary goal of this study was to create a user-friendly application capable of effectively detecting phishing attacks using NLP and DL on the text within web page bodies. This approach leverages the semantic and syntactic content of the text.

Reviewing the existing literature, we identified a significant research gap in web page text analysis using NLP and DL. At the same time, we discovered numerous studies

that address phishing using NLP and DL, with most focusing on phishing emails. The emphasis on web pages was predominantly on URL analysis, neglecting the text within the pages. Moreover, we found no evidence of these models being implemented in web browsers in the literature we examined, highlighting the novelty of our approach.

We refined the initial model based on the text of the web pages and 98.55% achieved a mean accuracy of 98.55%. This was accomplished with a batch size of 64, five epochs, 64 neurons, and GloVe dimension of 50. The new solution is 1.7% higher than the initial proposal in [9] and trains and tests 12 times faster.

We have developed a user-friendly extension for Google Chrome. This extension, built on our refined model, not only detects phishing attacks by analyzing the text of web pages using NLP and applying the BiGRU algorithm but also ensures a secure browsing experience. It accomplishes this without introducing technical complexities, making it accessible to all users, regardless of their technical proficiency. This seamless integration of advanced technology into everyday browsing is a significant step towards safer Internet use.

This work was based on refining the BiGRU DL algorithm, which conducts text analysis by NLP using embeddings from the GloVe dictionary. Initially developed for NLP tasks, Transformer algorithms have revolutionized the DL field. Thus, as part of our future work, we will develop an algorithm that can detect phishing attacks using the four most commonly used Transformer algorithms.

In future research involving the application of various Transformer algorithms to detect phishing attacks, we will develop an extension that can be installed not only in the Chrome browser but also in Mozilla Firefox, Edge, and Opera browsers.

7 REFERENCES

[1] M. Elsadig, A. O. Ibrahim, S. Basheer, M. A. Alohali, S. Alshunaifi, H. Alqahtani, N. Alharbi, and W. Nagmeldin, "Intelligent deep machine learning cyber phishing URL detection based on BERT features extraction," *Electronics*, vol. 11, no. 22, p. 3647, 2022. https://doi.org/10.3390/electronics11223647

[2] W. Fuertes *et al.*, "Impact of social engineering attacks: A literature review," *Smart Innovation, Systems and Technologies*, vol. 255, pp. 25–35, 2021. https://doi.org/10.1007/978-981-16-4884-7_3

[3] E. Benavides-Astudillo *et al.*, "A framework based on personality traits to identify vulnerabilities to social engineering attacks," *Communications in Computer and Information Science*, vol. 1535, pp. 381–394, 2022. https://doi.org/10.1007/978-3-031-03884-6_28

[4] E. Benavides-Astudillo *et al.*, "Analysis of vulnerabilities associated with social engineering attacks based on user behavior," *Communications in Computer and Information Science*, vol. 1535, pp. 351–364, 2022. https://doi.org/10.1007/978-3-031-03884-6_26

[5] statista, "Worldwide organizations most targeted by phishing attacks in 2023, by industry," 2023. [Online]. https://www.statista.com/statistics/420442/organizations-most-affected-byphishing/

[6] A. Raja Saleem, S. Balasubaramanian, P. Ganesan, J. Rajasekaran, and R. Karthikeyan, "Weighted ensemble classifier for malicious link detection using natural language processing," *International Journal of Pervasive Computing and Communications*, 2023.

[7] X. Zhang, Y. Zeng, X.-B. Jin, Z.-W. Yan, and G. G. Geng, "Boosting the phishing detection performance by semantic analysis," in *Proceedings – 2017 IEEE International Conference on Big Data, Big Data 2017*, 2017, pp. 1063–1070. https://doi.org/10.1109/BigData.2017.8258030

[8] E. Benavides-Astudillo *et al.*, "Comparative study of deep learning algorithms in the detection of phishing attacks based on HTML and text obtained from web pages," *Communications in Computer and Information Science*, vol. 1755, pp. 386–398, 2022. https://doi.org/10.1007/978-3-031-24985-3_28

[9] E. Benavides-Astudillo, W. Fuertes, S. Sanchez-Gordon, D. Nuñez-Agurto, and G. Rodríguez-Galán, "A phishing-attack-detection model using natural language processing and deep learning," *Applied Sciences 2023*, vol. 13, no. 9, p. 5275, 2023. https://doi.org/10.3390/app13095275

[10] M. Nguyen and T. Nguyen, "A deep learning model with hierarchical lstms and supervised attention for anti-phishing," *ArXiv Preprint*, 805.01554, 2018.

[11] Y. Safonov, "Phishing detection using deep learning attention techniques," vol. 1, no. 1, pp. 131–135, 2021. https://10.13164/eeict.2021.131

[12] S. Bagui, D. Nandi, S. Bagui, and R. J. White, "Machine learning and deep learning for phishing email classification using one-hot encoding," *Journal of Computer Science*, vol. 17, no. 7, pp. 610–623, 2021. https://doi.org/10.3844/jcssp.2021.610.623

[13] I. AbdulNabi and Q. Yaseen, "Spam email detection using deep learning techniques," *Procedia Computer Science*, vol. 184, pp. 853–858, 2021. https://doi.org/10.1016/j.procs.2021.03.107

[14] A. Alhogail and A. Alsabih, "Applying machine learning and natural language processing to detect phishing email," *Computers & Security*, vol. 110, p. 102414, 2021. https://doi.org/10.1016/j.cose.2021.102414

[15] R. Brindha *et al.*, "Intelligent deep learning based cybersecurity phishing email detection and classification," *Computers, Materials & Continua*, vol. 74, no. 3, pp. 5901–5914, 2022. https://doi.org/10.32604/cmc.2023.030784

[16] M. Dewis and T. Viana, "Phish responder: A hybrid machine learning approach to detect phishing and spam emails," *Applied System Innovation*, vol. 5, no. 4, p. 73, 2022. https://doi.org/10.3390/asi5040073

[17] K. Haynes, H. Shirazi, and I. Ray, "Lightweight URL-based phishing detection using natural language processing transformers for mobile devices," *Procedia Computer Science*, vol. 191, pp. 127–134, 2021. https://doi.org/10.1016/j.procs.2021.07.040

[18] S. Atawneh and H. Aljehani, "Phishing email detection model using deep learning," *Electronics*, vol. 12, no. 20, p. 4261, 2023. https://doi.org/10.3390/electronics12204261

[19] J. E. Coyac-Torres, G. Sidorov, E. Aguirre-Anaya, and G. Hernández-Oregón, "Cyberattack detection in social network messages based on convolutional neural networks and NLP techniques," *Machine Learning and Knowledge Extraction*, vol. 5, no. 3, pp. 1132–1148, 2023. https://doi.org/10.3390/make5030058

[20] E. Lee, "A gentle introduction to hyperparameter tuning with scikit learn and python," 2023. https://drlee.io/a-gentle-introduction-to-hyperparameter-tuning-with-scikit-learn-and-python-835139d55ef

[21] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[22] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. United States of America: O'Reilly Media, Inc., 2022.

[23] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, 2018. https://doi.org/10.1007/978-3-030-05318-5_3

[24] S. Giri, S. Banerjee, K. Bag, and D. Maiti, "Comparative study of content-based phishing email detection using global vector (GloVe) and bidirectional encoder representation from transformer (BERT) word embedding models," in *2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*, 2022, pp. 1–6. https://doi.org/10.1109/ICEEICT53079.2022.9768612

[25] F. Pedregosa, *et al.*, "Cross-validation: Evaluating estimator performance," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. https://scikit-learn.org/stable/modules/cross_validation.html

[26] Phishload. [Online]. https://www.medien.ifi.lmu.de/team/max.maurer/files/phishload/download.html

[27] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," 2014. https://nlp.stanford.edu/projects/glove/

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[29] J. Heaton, "Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning," *Genetic Programming and Evolvable Machines*, vol. 19, pp. 305–307, 2017. https://doi.org/10.1007/s10710-017-9314-z

8 AUTHORS

Eduardo Benavides-Astudillo is a Researcher and Professor at the Department of Computer Science at Universidad de las Fuerzas Armadas ESPE in Ecuador since 2006. Eduardo has a Master's degree in Systems Management from Universidad de las Fuerzas Armadas ESPE, Ecuador, obtained in 2015. He is doing Ph.D. in Informatics at Escuela Politécnica Nacional in Ecuador. His main research interests include machine learning, deep learning, and cybersecurity. He has published several research papers in scientific journals indexed in SJR and JCR rankings (E-mail: diego.benavides@epn.edu.ec; ORCID: https://orcid.org/0000-0003-4543-0082).

Walter Fuertes is a Full Professor in the Department of Computer Science at the Universidad de las Fuerzas Armadas ESPE in Sangolquí, Ecuador. He holds an Engineering degree in Computer Systems, a Master's degree in Computer Networking, and a Ph.D. (Hons.) degree in Computer Science and Telecommunications (E-mail: wmfuertes@espe.edu.ec).

Sandra Sanchez-Gordon is a Researcher and Professor in the Department of Informatics and Computer Science at Escuela Politécnica Nacional in Ecuador since 1994. Sandra holds a Ph.D. in Applied Informatics from the University of Alicante, Spain, in 2017, and a Master's in Software Engineering from Drexel University, USA, in 2001. With 30 years of experience, she has expertise in developing and implementing software solutions in Ecuador, Panama, and the USA. Her research interests include Human-Computer Interaction, Usability, Accessibility, and Applications of Artificial Intelligence (E-mail: sandra.sanchez@epn.edu.ec).

Daniel Nuñez-Agurto is a Researcher and Professor in the Department of Computer Science at the Universidad de las Fuerzas Armadas – ESPE Sede Santo Domingo since 2017. Daniel holds a Master's degree in Systems Management from the Universidad de las Fuerzas Armadas – ESPE in Ecuador, earned in 2015. He is in the researching phase to obtain his Doctorate in Computer Science from the Universidad Nacional de La Plata, Argentina. Daniel has over 18 years of experience in developing and implementing data networks and telecommunications solutions. His primary research interests include Software-Defined Networking, Data Networks, Machine Learning, and Cybersecurity (E-mail: adnunez1@espe.edu.ec).