INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2023, VOL. 69, NO. 3, PP. 485–491
Manuscript received July 7, 2023; revised July, 2023. DOI: 10.24425/ijet.2023.146496

# Email Phishing Detection with BLSTM and Word Embeddings

Rafał Wolert, and Mariusz Rawski

***Abstract*—Phishing has been one of the most successful attacks in recent years. Criminals are motivated by increasing financial gain and constantly improving their email phishing methods. A key goal, therefore, is to develop effective detection methods to cope with huge volumes of email data. In this paper, a solution using BLSTM neural network and FastText word embeddings has been proposed. The solution uses preprocessing techniques like stop-word removal, tokenization, and padding. Two datasets were used in three experiments: balanced and imbalanced, whereas in the imbalanced dataset, the effect of maximum token size was investigated. Evaluation of the model indicated the best metrics: 99.12% accuracy, 98.43% precision, 99.49% recall, and 98.96% f1-score on the imbalanced dataset. It was compared to an existing solution that uses the DL model and word embeddings. Finally, the model and solution architecture were implemented as a browser plug-in.**

***Keywords*—phishing; BLSTM; word embeddings**

## I. Introduction

PHISHING is a well-known type of social engineering attack that exploits human trust [1]. Phishing emails is one of the most popular types of such attacks. Research shows that 96% of phishing attacks originate from emails, while 3% come from malicious websites and 1% through phone calls [2].

A phishing email is a type of spam message that uses deception to appear as if it is from a legitimate company or bank. The email contains a link that redirects users to a fake website designed to fraudulently obtain sensitive financial information such as usernames, passwords, and credit card numbers. The name phishing is reminiscent of fishing – catching fish. Criminals, like fishermen, use appropriately prepared ”bait”.

Despite over 10 years of research into phishing, there has been no significant progress in reducing the prevalence of phishing attacks. This may be due to several reasons: phishing is more complex than human perception, practical techniques used by researchers may have overlooked key problem parameters, and phishing attacks exploit human unawareness which may not be addressed solely by technical methods and may require human interventions such as training and awareness [3].

There are several ways to combat phishing. One of them is the use of artificial intelligence (AI) in cybersecurity [4]. AI has improved email security by providing speed, accuracy, and the ability to conduct in-depth investigations. With the help of pre-existing datasets, AI can identify various types of attacks, including spam, phishing, spear phishing, and more.

The remainder of the paper is organized as follows: *Section II* highlights existing work that apply machine and deep learning techniques with NLP. *Section III* presents the methodology used in this research: used dataset, preprocessing, feature extraction, deep learning model, and proposed tool. *Section IV* describes experiments conducted in this research along with described metrics, two dataset tasks, and summarized results. *Section V* discusses model performance with comparison study and the solution's limitations along with future work.

Authors are with the Institute of Telecommunications, Faculty of Electronics and Information Technology, Warsaw University of Technology, Poland (e-mail: {rafal.wolert.stud, mariusz.rawski}@pw.edu.pl)

## II. Related Work

In recent years, several reviews have been published on the topic. In [5] authors make an update to previous systematic literature surveys with a focus on the latest trends in phishing detection techniques. According to the review, most research papers propose approaches based on Machine Learning (ML) techniques. Most studies used Random Forest Classifier, but Convolution Neural Network (CNN) achieved the highest accuracy of 99.98%.

In [6] the development of Phish Responder, a detection solution to address the challenge of phishing and spam emails is discussed. The authors propose a solution that uses a hybrid machine learning approach combining natural language processing (NLP). This Python-based command line solution has presented an average accuracy of 94% with the MLP model for numerical-based datasets.

The authors of the paper [7] developed MailTrout, a browser extension that incorporates machine learning within a usable security tool to assist users in detecting phishing emails. TensorFlow was used to develop and train an ML model using a dataset of fraudulent and legitimate emails. Proposed solution demonstrated high levels of accuracy when detecting phishing emails and high levels of usability for end-users.

The paper [8] discusses the need for an efficient mechanism to detect phishing emails to provide better security against such attacks to the common user. The authors created a real-time in-house corpus of phishing and legitimate emails and proposed efficient techniques to detect phishing emails using word embedding and machine learning algorithms. The proposed system uses only four email header-based heuristics for the classification of emails.

## III. Methodology

This section presents the methodology used in this research. In *A. Overview*, a general overview is presented. In *B. Dataset*, the used dataset, is described. In *C. Preprocessing*, the preprocessing is explained along with *D. Feature extraction*, where feature extraction with FastText embeddings is brought up. In *E. Deep learning model*, the deep learning model is explained. Finally, in *F. Proposed tool*, the proposed browser extension is demonstrated.

### A. Overview

The general overview of the architecture and pipeline used in this work is presented in Fig. 1.

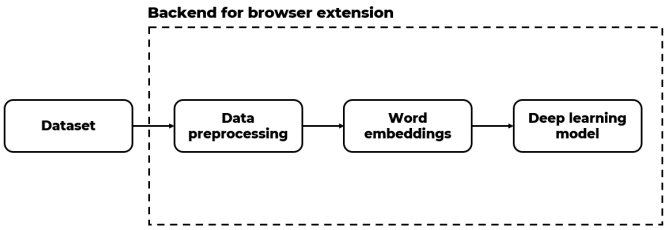


Fig. 1. Overview of the used methodology

This figure represents four stages used in our approach: dataset, data preprocessing, word embeddings, and deep learning model. The last three blocks represent elements used as a browser extension's backend.

### B. Dataset

The two datasets used in the research consist of combined email bodies from the Enron email corpus [9] and Jose Nazario's phishing corpus [10]. Both datasets are well-known for their usage in phishing detection by providing real emails received or sent by users.

The Enron email corpus consists of 619 446 messages sent by 158 employees of Enron Corporation [11]. In our work, we have loaded all messages with omitting directories such as: *discussion_threads*, *notes_inbox* due to the fact stated by the Authors that such folders do not appear to be used by users and are instead a computer generated or already present in the users' other folders. We have also observed that folder, *_sent_mail*, is irrelevant for the same reasons. An initial preprocessing task was performed on the whole corpus before selecting the random sample. The initial preprocessing task was used to remove records with null email bodies or email bodies that cannot be decoded. This ensured the selected sample could be used in later pipeline processes. Then, we randomly selected a sample of loaded emails and labeled them benign ones, which will be further described in this section.

Jose Nazario's phishing corpus is a private corpus comprising phishing emails sent to its creator from 2005 to 2022. Our work has focused on the data's quantity and quality. Due to that reason, we have included all emails ranging from 2005 to 2022 in our dataset, 10 706 total in raw format, before preprocessing. By doing so, we are increasing the dataset's size and providing more examples of the newest phishing attacks in the emails.

In provided work, two datasets were created for imbalanced and balanced classification tasks. Table I presents the ratio of phishing and ham emails in created datasets.

TABLE I
SUMMARY OF DATASETS

| Task | Phishing | Ham | Total count |
|---|---|---|---|
| Imbalanced | 10 568 | 14 971 | 25 539 |
| Balanced | 10 568 | 10 568 | 21 136 |

As presented in Table I, created datasets have 138 fewer emails than in the original Jose Nazario's phishing corpus. This is because preprocessing was done on the initial corpus, which will be described later in the next section.

A subset of ham emails for imbalanced tasks was created by randomly selecting 15 000 samples from the initial-preprocessed Enron email corpus, of which 29 emails were discarded in the later preprocessing phase. The same action was applied to the subset of ham emails for balanced tasks. An equal amount of 10 568 phishing emails was used as the random sample size in the ham subset.

### C. Preprocessing

This section is divided into two subsections containing information about the preprocessing of email bodies and data analysis of the whole dataset.

*1) Email preprocessing:* Preprocessing was done on datasets created for imbalanced and balanced tasks for phishing and ham emails. It involves steps in the order described below:

1) Extract messages from HTML if necessary.
2) Lowercase all characters.
3) Replace all URLs with: *fixedstringurl*.
4) Replace all emails with: *fixedstringsemails*
5) Remove all nonalphanumeric characters; include only characters (a-z or A-Z) and numbers (0-9).
6) Remove additional whitespaces like tabs and multiple spaces to ensure only one whitespace between words.
7) Tokenize words based on whitespaces and remove stop words.

If any steps above failed, the record was marked with a fixed string as *to_manual_extraction* and looked at manually.

Step 7) involves tokenization and stop word removal, which was conducted with the usage of spaCy Python library, specifically *en_core_web_sm-3.5.0* pipeline [12]. This model is described as a general-purpose model with components such as *tok2vec*, *tagger*, *parser*, *senter*, *attribute_ruler*, *lemmatizer*, *ner*. Our work focused only on tokenization and stop word removal, so we have used attributes such as: "text" and "is_stop" from Token class [13] to ensure that the pipeline is explicitly narrowed to those two NLP techniques.

Tokenization and stop-word removal are the most common and basic NLP techniques. As mentioned by authors of [14], in their systematic review of 100 research studies about

phishing detection using NLP techniques, 59 studies used NLP techniques such as stop-word removal, punctuations, special characters, stemming, and tokenization.

Tokenization is the process of breaking sentences into single words, in our case, based on single whitespaces. Words created from tokenization are referred to as tokens. Tokens were discarded if they were a stopword, which are words that do not carry significant information in the sentences such as: “a”, “the”, “or”.

Additionally, records with emails that contained empty bodies after preprocessing were dropped from both phishing and ham datasets in both tasks.

*2) Dataset analysis:* To ensure that the feature extraction task will be conducted in reasonable time with average resource usage, we have limited the token list length in the whole dataset in both tasks to less than 20 000. Only two records, one phishing email, and one ham email were found to be above 20 000 tokens in the imbalanced task; they contained 46 436 and 24 345 tokens, respectively.

### D. Feature extraction

Feature extraction task was achieved using word embeddings. Word embeddings are a group of models and methods used to represent a word as a real-valued vector in dense, low-dimensional space [8], typically ranging from 100 to 300. Each word has its vector, and words used in similar contexts have similar vector representations; thus, their meaning is captured.

In our work, we are focusing on using word embeddings with FastText. FastText is a library developed by Facebook AI Research. Models used in FastText are learning word representations for character *n*-grams and represent words as the sum of the *n*-gram vectors and are referred to as subword information [15].

FastText provides two architectures for creating word embeddings: SkipGram and Continuous-Bag-Of-Words (CBOW). The main difference is that SkipGram uses a target word to predict the context (surrounding words), whereas CBOW uses the context, a window of words, to predict the target word.

The main advantage of FastText over other word embedding models, such as Word2Vec, is its capability to handle Out-Of-Vocabulary (OOV) words, which are words that do not appear in the training set of the model. The embedding vector of OOV words is represented as an average of the vector representation of their *n*-grams. This is a crucial concept and motivation for choosing FastText in the phishing detection task.

According to [16], phishing emails often contain spelling, grammar, and punctuation mistakes that result from an attacker’s lack of English proficiency. Our approach eliminates that problem so that words with grammar or spelling mistakes still have a decent embedding representation.

In our work, we have used one of the pre-trained word vectors from the FastText library, named *cc.en.300.bin*, which was trained for the English language. That model was trained on *Common Crawl* [17], an open repository of web crawl data using CBOW with position-weights, in dimension 300, with character *n*-grams of length 5, a window of size 5 and 10 negatives [18]. Table II summarizes the parameters.

TABLE II
SUMMARY OF FASTTEXT MODEL WITH POSITION WEIGHTS

| parameter | value |
|---|---|
| dimension size | 300 |
| *n*-grams | 5 |
| window size | 5 |
| negative sample size | 10 |

After creating the embeddings for tokenized words in the dataset, the fixed size of vectors was set by padding the sequences using the Tensorflow *pad_sequences* method [19]. Table III summarizes the parameters used in this approach.

TABLE III
SUMMARY OF PAD_SEQUENCES METHOD

| parameter | value |
|---|---|
| maxlen | 300 |
| dtype | float32 |
| padding | post |
| value | 0.0 |

*Maxlen* is the parameter that is the maximum length of all sequences. It was set as a value from 0.90 to 0.95 percentile of tokens length in both datasets: imbalanced and balanced. Additionally, we have experimented with *maxlen* parameter set to 324 as it represented 0.95 percentile of the used balanced dataset. Due to memory requirements and resource usage, the same experiment could not be performed with the imbalanced dataset. *Padding* parameter was set to *post* so that the parameter *value* set to 0.0 is added at the end of token sequences.

### E. Deep learning model

This section is divided into two subsections containing information about data manipulations used on the datasets and chosen model architecture.

*1) Data transformations:* Before feeding data to a neural network, the train-test split method was performed on both datasets for imbalanced and balanced tasks, in the ratio of 80% for training and 20% for testing.

Data manipulations were performed on both datasets for imbalanced and balanced tasks to optimize the model pipeline [20] as summarized in Table IV.

TABLE IV
SUMMARY OF DATA MANIPULATIONS

| parameter | value |
|---|---|
| train test ratio | 80:20 |
| batch size | 32 |
| prefetch buffer size | AUTOTUNE |

*Batch size* parameter was set to 32 as a compromise between memory usage and model performance. *Prefetch* transformation overlaps the preprocessing and model execution of a

training step to speed up the process of training [20]. Since *prefetch* has *buffer_size* attribute that should be equal to the number of batches consumed in a single training step, we used *buffer_size* of *AUTOTUNE*. By doing so, the value is tuned dynamically at the runtime.

*2) Model architecture:* The model was built with the usage of TensorFlow [21], and Keras [22].

The used model architecture is built upon the Bidirectional network. It is a type of recurrent neural network (RNN) network that can learn long-term dependencies from past states and future states [23], as opposed to Long Short-Term Memory (LSTM), a similar network, which can learn dependencies only from past context. This approach has been successful in many NLP tasks, such as Part-of-speech (POS) tagging in classification task [23].

Additionally, as stated in [14], after 2019, more deep learning techniques were utilized in phishing email detection. This suggested that further work, for example, in LSTM and CNN models, is necessary to provide sufficient deep learning tools.

Fig. 2 represents the used model architecture with corresponding layers. Besides every layer, there is a set of parameters associated with it.

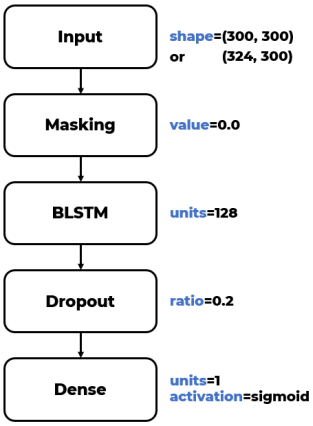


Fig. 2. Model architecture with used parameters

The first layer used in model architecture is *Input* layer. It expects a numerical date of: *maxlen* by *dimension_size*, which, as described in *D Feature extraction*, was set to 300 by 300 on both imbalanced and balanced dataset or 300 by 324 only on the balanced dataset.

The next layer is *Masking* layer. Its core function informs the neural network that some data were padded with *0.0*, as described in *D Feature extraction*, and therefore should be ignored.

The third layer consists of *BLSTM* layer, with 128 *units*. After that, *Dropout* layer is inserted with *ratio* set to 0.2. This layer is used as a regularization technique to prevent overfitting to the present data.

The last layer consists of *Dense* layer with one *unit* and sigmoid *activation function* as our work focuses on the binary classification task. The defined loss function is *binary cross-entropy*, and the *optimizer* is set to Adam.

The model's configuration is summarized in Table V.

TABLE V
SUMMARY OF MODEL CONFIGURATION

| parameter | value |
|---|---|
| Input shape | (300,300) or (324,300) |
| Masking value | 0.0 |
| BLSTM units | 128 |
| Dropout ratio | 0.2 |
| Dense units | 1 |
| Dense activation function | sigmoid |
| loss function | binary cross-entropy |
| optimizer | Adam |

### F. Proposed tool

The model pipeline can be easily integrated into a browser extension. This approach combines the accuracy of a deep learning detection system and the usability of a browser extension. The proposed tool has been tested with Gmail API [24] for email access, and a browser extension was installed on Google Chrome. The model pipeline with preprocessing and embeddings was deployed using the FastAPI library [25].

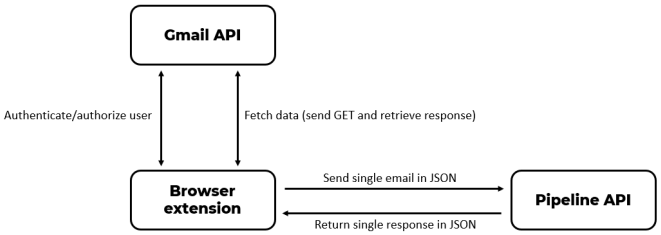


Fig. 3. Proposed phishing detection tool

Fig. 3 presents the overview of the proposed tool. Core elements and connections between them are described below:

- **Gmail API**: Used to authenticate/authorize the extension to fetch email data within the user inbox using OAuth 2.0 provided in the Gmail API library.
- **Browser extension**: Used to fetch data using Gmail API, forward the response in JSON format to Pipeline API, and output the model's response on the screen.
- **Pipeline API**: Used to perform all architecture tasks on a given sample and return the model's prediction probability in JSON.

The extension was deployed as Proof-of-Concept. Fig. 4 a) shows a successful authentication view using Gmail API while b) shows the detection of opened e-mail in the inbox. Fig. 5 a), b), and Fig. 6 c) show views of corresponding: email with low phishing rate, medium phishing rate, and high phishing rate. The detection score is the model's prediction output rounded to three decimal places and scaled to 100%. The three compartments are designated in thirty percent increments, and each compartment has its own designated background color.

## IV. EXPERIMENTAL RESULTS

This section is divided into four subsections containing information about experiments conducted with imbalanced

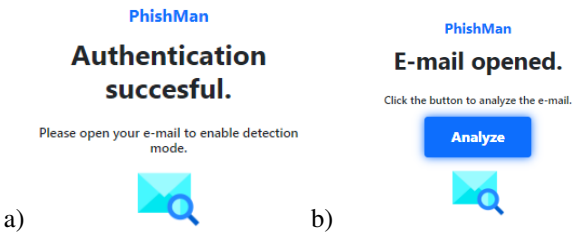


Fig. 4. View of Gmail authentication and opened e-mail

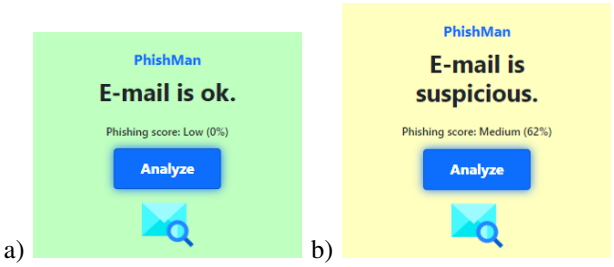


Fig. 5. Views of low, medium and high phishing rate

(subsection *B. Imbalanced dataset task*) and balanced tasks (*C. Balanced dataset task*) as well as metrics used within (subsection *A. Metrics*). The last subsection, *D. Summarized results* summarizes all results.

In all experiments, the dataset presented in *III. Methodology B. Dataset* was split into 80% for training and 20% for validation. Also, *early stopping* technique was used with a target metric to loss function on the validation dataset. As a result, training is stopped after a certain number of epochs, named *patience*, and the best weights based on target value are restored for the final model. For all experiments, *patience* was set to 10.

Experiments were performed with the usage of Google Colab with GPU acceleration.

Three experiments were conducted: one considering **Imbalanced dataset task** and two considering **Balanced dataset task** as summarized in Table VI.

### A. Metrics

Four metrics were chosen: *accuracy*, *precision*, *recall*, *f1-score* for evaluating model's performance. Metrics are defined in the equations (1), (2), (3), (4), where:

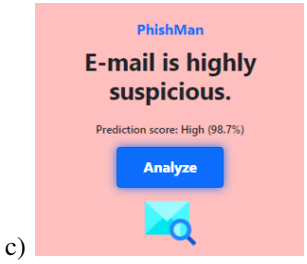


Fig. 6. Views of low, medium and high phishing rate

TABLE VI
SUMMARY OF EXPERIMENTS CONFIGURATION

| dataset | input shape | early stopping patience | epochs |
|---|---|---|---|
| imbalanced | (300,300) | 10 | 17/100 |
| balanced | (300,300) | 10 | 22/100 |
| balanced | (300,324) | 10 | 21/100 |

- *TP* is defined as the number of phishing e-mails classified as phishing,
- *TN* is defined as the number of ham e-mails classified as ham,
- *FP* is defined as the number of phishing e-mails classified as ham,
- *FN* is defined as the number of ham e-mails classified as phishing.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} * 100 \quad (1)$$

$$precision = \frac{TP}{TP + FP} * 100 \quad (2)$$

$$recall = \frac{TP}{TP + FN} * 100 \quad (3)$$

$$f1 - score = \frac{2 * TP}{2 * TP + FP + FN} * 100 \quad (4)$$

### B. Imbalanced dataset task

An imbalanced dataset task was performed on the imbalanced dataset. The training process stopped at 17 epochs out of 100 epochs set due to early stopping. The best set of parameters was found at the six epochs, achieving 99.12% accuracy, 98.43% precision, 99.49% recall, and 98.96% f1-score on the validation set.

### C. Balanced dataset task

A balanced dataset task was performed on the balanced dataset. The first experiment was conducted with a data input shape of (300, 300) and stopped at 22 epochs out of 100 epochs set due to early stopping. The best set of parameters was found at 11 epochs, achieving 98.77% accuracy, 98.96% precision, 98.59% recall, and 98.77% f1-score on the validation set.

The second experiment was conducted with a data input shape of (300, 324) and stopped at 21 epochs out of 100 epochs due to early stopping. The best set of parameters was found at 10 epochs, achieving 98.58% accuracy, 98.41% precision, 98.68% recall, and 98.54% f1-score on the validation set.

### D. Summarized results

The summarized results of all experiments are presented in Table VII.

TABLE VII
SUMMARY OF EXPERIMENTS

| task | imbalanced | balanced | balanced |
|---|---|---|---|
| input size | (300,300) | (300,300) | (300,324) |
| best epoch | 6 | 11 | 10 |
| accuracy | **99.12%** | 98.77% | 98.58% |
| precision | 98.43% | **98.96%** | 98.41% |
| recall | **99.49%** | 98.59% | 98.68% |
| f1-score | **98.96%** | 98.77% | 98.54% |

## V. EVALUATION AND DISCUSSION

This section is divided into three subsections. Subsection *A. Model performance* describes model performance in terms of the experiments conducted. In Subsection *B. Comparison study* presented model is compared to existing solutions. Subsection *C. Limitations and future work* discuss the limitations of a featured solution and future work to conduct.

### A. Model performance

The model with the highest metrics: accuracy, recall, and f1-score, was trained on the imbalanced dataset with an input shape of 300 by 300. The highest precision was achieved on a balanced dataset task with an input shape of 300 by 300. The last experiment with an increased maximum token length to 324 did not improve model performance compared to the imbalanced dataset with a maximum token size set to 300.

### B. Comparision study

We have compared the provided architecture with existing solution utilizing deep learning and word embeddings. We selected the imbalanced task model, which achieved the highest three metrics out of four from all experiments. In the article [26] Word2Vec embedding was used with LSTM architecture achieving 0.876 accuracy, 0.957 precision, 0.90 recall, and 0.928 f1-score on the dataset without email headers. A detailed comparison has been presented in table VIII.

TABLE VIII
COMPARISION WITH DEEPANTI-PHISHNET

| | our approach | DeepAnti-PhishNet |
|---|---|---|
| dataset | 10568 phishing 14971 ham | 612 phishing 5088 ham |
| train test ratio % | 80:20 | 73:27 |
| type of embeddings | FastText | Word2Vec |
| embedding vector size | 300 | 200 |
| type of NN | BLSTM | LSTM |
| max epochs | 100 | 1000 |
| accuracy % | 99.12 | 86.00 |
| precision % | 98.43 | 96.20 |
| recall % | 99.49 | 87.60 |
| f1-score % | 98.96 | 91.70 |

As presented in Table VIII, there are some major differences between the used approaches. Firstly, our approach utilizes more email phishing samples which are crucial in the neural network training process. By giving a more balanced dataset, we avoid model overfitting, which was the case with DeepAnti-PhishNet, as mentioned by the authors. Also, authors of DeepAnti-PhishNet used preprocessing that included: conversion of all characters to lower case, ignoring punctuation marks and special characters, and assigning a unique number for the unknown word [26], where our solution utilizes tokenization and stop-word removal along with padding the token sequences to gain fixed size of vectors.

Another difference was the type of embeddings used. Authors of DeepAnti-PhishNet used Word2Vec embeddings which has a disadvantage compared to FastText embeddings that can produce OOV embeddings for words that, for example, are misspelled in phishing emails. Additionally, our approach used a vector size of 300 compared to a 200 vector size in DeepAnti-PhishNet, which can capture more semantic information.

One of the main differences is the neural network architecture used in both approaches. Our proposed solution uses BLSTM, which, compared to LSTM, can learn dependencies from past and future states, which can be beneficial when dealing with email bodies.

Our work also implemented the Early Stopping technique that stopped model training after not improving the validation's loss. On the contrary, the authors of DeepAnti-Phishnet used 10-fold cross-validation, which in our case was not implemented.

Although both approaches have differences in feature extraction methods and neural network architectures, our approach outperforms DeepAnti-PhishNet in all used metrics.

### C. Limitations and future work

Our presented work on phishing email detection has its limitations. The main point to consider is the dataset. We have created datasets for both imbalanced and balanced tasks, yet no deep data analysis was done on both datasets to ensure their representativity in phishing detection. This is a crucial point, as a lack of variety in provided samples could lead to model overfitting or underfitting. Also, the email headers can be considered as they carry another source of information that could improve the overall model score.

In addition, one must consider the word embeddings used as a feature extraction task. We have used a pre-trained FastText model, which can be improved by fine-tuning it with phishing-related texts, to capture semantic information from email bodies better. Due to computation limitations, we have not used an increased maximum token size of 324 with balanced dataset tasks, which could potentially improve the model's performance.

Besides dataset and feature extraction tasks, hyperparameter tuning must be provided to ensure that the neural network has the best performance on a given dataset. Additionally, k-fold cross-validation could be used to prevent overfitting.

Moreover, our solution could potentially be used for different phishing detection tasks. One thing to consider is the multilanguage model for phishing emails written in different

languages. FastText library provides pre-trained models for different languages, and used with different datasets could lead to multilanguage phishing detection solutions. Furthermore, using Tensorflow, our model could be compiled to its lightweight version to detect phishing sent over SMS, called *smishing*.

Using word embeddings combined with BLSTM, we can efficiently detect phishing emails and apply that solution to real-world scenarios using the browser extension. Further work must be conducted to explore the usage of deep learning models combined with word embeddings in phishing detection.

## VI. Conclusions

The paper explores the use of deep learning and word embeddings in detecting phishing. We used a BLSTM neural network architecture with FastText embedding and conducted experiments on two datasets: one imbalanced and one balanced. The model performed best on the imbalanced dataset, achieving 99.12% accuracy, 98.43% precision, 99.49% recall, and 98.96% f1-score.

We also developed a proof-of-concept tool that uses our model as the backend and can be easily deployed as a browser extension. Our approach was compared to another method that used an LSTM neural network and Word2Vec embeddings. We discussed the limitations of our work and potential future improvements, as well as other phishing detection tasks that our solution could be applied to.

## References


[1] A. Almomani, B. B. Gupta, S. Atawneh, A. Meulenberg, and E. Almomani, "A survey of phishing email filtering techniques," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 2070–2090, 2013. [Online]. Available: https://doi.org/10.1109/SURV.2013.030713.00020

[2] F. Labs, "2020 phishing and fraud report," 2020, [Accessed: 14 June 2023]. [Online]. Available: https://www.f5.com/labs/articles/threat-intelligence/2020-phishing-and-fraud-report

[3] A. Das, S. Baki, A. El Aassal, R. Verma, and A. Dunbar, "Sok: A comprehensive reexamination of phishing research from the security perspective," *IEEE Communications Surveys Tutorials*, vol. 22, no. 1, pp. 671–708, 2020.

[4] S. Salloum, T. Gaber, S. Vadera, and K. Shaalan, "A systematic literature review on phishing email detection using natural language processing techniques," *IEEE Access*, vol. 10, pp. 65 703–65 727, 2022. [Online]. Available: https://doi.org/10.1109/ACCESS.2022.3183083

[5] A. Safi and S. Singh, "A systematic literature review on phishing website detection techniques," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 2, pp. 590–611, 2023. [Online]. Available: https://doi.org/10.1016/j.jksuci.2023.01.004

[6] 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, 2022. [Online]. Available: https://doi.org/10.3390/asi5040073

[7] P. Boyle and L. Shepherd, "Mailtrout: a machine learning browser extension for detecting phishing emails," in *34th British Human Computer Interaction Conference 2021 proceedings*, ser. Electronic Workshops in Computing, J. Nocera, H. Petrie, G. Sim, T. Clemmensen, and F. Spyridonis, Eds. BCS Learning Development Ltd., Jul. 2021, pp. 104–115, 34rd British Human Computer Interaction Conference : Post-Pandemic HCI – Living digitally ; Conference date: 19-07-2021 Through 21-07-2021. [Online]. Available: https://doi.org/10.14236/ewic/HCI2021.10

[8] S. M. and A. R. Pais, "Classification of phishing email using word embedding and machine learning techniques," *Journal of Cyber Security and Mobility*, vol. 11, no. 03, p. 279–320, May 2022. [Online]. Available: https://doi.org/10.13052/jcsm2245-1439.1131

[9] Enron email dataset. [Accessed: 14 June 2023]. [Online]. Available: https://www.cs.cmu.edu/~enron/

[10] Jose nazario phishing email corpus. [Accessed: 14 June 2023]. [Online]. Available: https://monkey.org/~jose/phishing/

[11] B. Klimt and Y. Yang, "Introducing the enron corpus," in *First Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, 2004, [Accessed: 14 June 2023]. [Online]. Available: https://www.ceas.cc/papers-2004/168.pdf

[12] spacy python model. [Accessed: 14 June 2023]. [Online]. Available: https://github.com/explosion/spacy-models/releases/tag/en_core_web_sm-3.5.0

[13] spacy token class attributes. [Accessed: 14 June 2023]. [Online]. Available: https://spacy.io/api/token#attributes

[14] S. V. K. S. Said Salloum, Tarek Gaber, "A systematic literature review on phishing email detection using natural language processing techniques," *IEEE Access*, vol. 10, pp. 2169–3536, June 2022. [Online]. Available: https://doi.org/10.1109/ACCESS.2022.3183083

[15] A. J. T. M. Piotr Bojanowski, Edouard Grave, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–147, June 2017. [Online]. Available: https://doi.org/10.1162/tacl_a_00051

[16] S. J. H. Marcus Butavicius, Ronnie Taib, "Why people keep falling for phishing scams: The effects of time pressure and deception cues on the detection of phishing emails," *Computers and Security*, vol. 123, December 2022. [Online]. Available: https://doi.org/10.1016/j.cose.2022.102937

[17] Common crawl. [Accessed: 14 June 2023]. [Online]. Available: https://commoncrawl.org/

[18] G. P. J. A. M. T. Grave Edouard, Bojanowski Piotr, "Learning word vectors for 157 languages," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[19] Tensorflow pad sequences method. [Accessed: 14 June 2023]. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/utils/pad_sequences

[20] Tensorflow data performance. [Accessed: 14 June 2023]. [Online]. Available: https://www.tensorflow.org/guide/data_performance

[21] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[22] F. Chollet *et al.*, "Keras," https://keras.io, 2015, [Accessed: 14 June 2023].

[23] P. Wang, Y. Qian, F. K. Soong, L. He, and H. Zhao, "Part-of-speech tagging with bidirectional long short-term memory recurrent neural network," 2015. [Online]. Available: https://doi.org/10.48550/arXiv.1510.06168

[24] Gmail api. [Accessed: 14 June 2023]. [Online]. Available: https://developers.google.com/gmail/api/guides

[25] Fastapi. [Accessed: 14 June 2023]. [Online]. Available: https://fastapi.tiangolo.com/

[26] P. P. M. A. K. S. K. Vinayakumar Ravi, Barathi Ganesh Hb, "Deepanti-phishnet: Applying deep neural networks for phishing email detection cen-aisecurity@iwspa-2018." Tempe AZ USA, March 2018, 1st AntiPhishing Shared Pilot at 4th ACM International Workshop on Security and Privacy Analytics (IWSPA 2018) [Accessed: 14 June 2023]. [Online]. Available: https://ceur-ws.org/Vol-2124/paper_9.pdf