http://dx.doi.org/10.14236/ewic/HCI2021.10

MailTrout: A Machine Learning Browser Extension for Detecting Phishing Emails

Paul Boyle
School of Design and Informatics
Division of Cyber Security
Abertay University
Bell Street, Dundee, DD1 1HG, UK
1600301@abertay.ac.uk

Lynsay A. Shepherd
School of Design and Informatics
Division of Cyber Security
Abertay University
Bell Street, Dundee, DD1 1HG, UK
lynsay.shepherd@abertay.ac.uk

The onset of the COVID-19 pandemic has given rise to an increase in cyberattacks and cybercrime, particularly with respect to phishing attempts. Cybercrime associated with phishing emails can significantly impact victims, who may be subjected to monetary loss and identity theft. Existing anti-phishing tools do not always catch all phishing emails, leaving the user to decide the legitimacy of an email. The ability of machine learning technology to identify reoccurring patterns yet cope with overall changes complements the nature of anti-phishing techniques, as phishing attacks may vary in wording but often follow similar patterns. This paper presents a browser extension called MailTrout, which incorporates machine learning within a usable security tool to assist users in detecting phishing emails. MailTrout demonstrated high levels of accuracy when detecting phishing emails and high levels of usability for end-users.

Phishing. Usable Security. Machine Learning. Browser Extension. Socio-Technical Security.

1. INTRODUCTION

Phishing emails generally attempt to persuade the recipient to reveal private or confidential information such as passwords or bank details and may deliver malware to infect the victim's machine. Information gained via phishing emails is used for fraudulent purposes by the sender, placing users at risk of identity theft, fraud, and significant financial loss.

Since the beginning of the COVID-19 pandemic, incidences of cyberattacks and cybercrime have increased considerably, including a sharp rise in phishing attempts (Lallie et al., 2021; Horgan et al., 2021). The pandemic has caused a fundamental shift in working practices and social interactions, creating an enhanced dependence on technology. Thus, users require additional support to identify potentially malicious emails.

Successful phishing scams can be costly for victims; in the UK, it is estimated that it takes 20 days and £960,000 to address the consequences of a single phishing or social engineering attack (Graham, 2018). To combat phishing attempts, email clients make use of spam filters to quarantine suspicious emails. However, these filters are not always successful; consequently, users require additional assistance to help them detect phishing emails in the form of anti-phishing tools and security education.

Anti-phishing tools may take the form of browser extensions, which can augment the users' browsing experience. These tools can identify different forms of phishing attacks; 'GoldPhish' is an Internet Explorer extension used to identify phishing webpages (Dunlop et al., 2010), while 'PhishAri' is a Google Chrome extension designed to detect phishing attempts on Twitter (Aggarwal et al., 2012). Anti-phishing tools may also make use of machine learning (ML), allowing systems to learn from existing data to make decisions without the need for human interaction. Previous work by Fette et al. (2007) was able to detect phishing emails based on features such as the number of hyperlinks present and the use of JavaScript.

This paper presents a prototype browser extension to detect phishing emails, which harnesses the power of machine learning to assist users in identifying phishing attempts, protecting them from becoming a victim of cybercrime. TensorFlow was used to develop and train an ML model using a dataset of fraudulent and legitimate emails. The model was evaluated for accuracy and converted for use in a prototype Google Chrome browser extension. The extension parses email text and evaluates sentiment and language to determine legitimacy. The extension was also tested with participants to evaluate its usability.

Proceedings of the BCS 34th British HCI Conference 2021, UK.

The remainder of the paper is organised as follows; Section 2 explores related work in phishing detection and machine learning. Section 3 describes the methodology. Results are presented in Section 4 and are discussed in Section 5. Section 6 presents conclusions and considers future work.

2. RELATED WORK

2.1 Existing anti-phishing tools

Phishing emails are not a new problem; however, attempts have increased in the wake of the COVID-19 pandemic attempts (Lallie et al., 2021). Usable security research has investigated anti-phishing tools to protect users and increase the awareness of risks associated with phishing attempts. A vital consideration when developing security tools – especially those aimed at non-technical individuals – is ensuring that they are accessible and user-friendly. Kumaraguru et al. (2010) developed two anti-phishing tools: the embedded email-based 'PhishGuru' and the online game 'Anti-Phishing Phil'. To ensure the tools provided effective education, the developers followed a series of design principles, including 'learning-by-doing', which states that people learn better when they practice their skills. In PhishGuru, instructional materials are embedded into the user's everyday tasks, such as checking their emails. Implanting the materials increases the prevalence of 'teachable moments' – optimal opportunities to convey a point or idea – increasing the tool's educational potential. 'Anti-Phishing Phil' and the concept of embedding phishing content into games has been explored by Dixon et al. (2019). The work highlighted that users prefer integrated tools to help them learn, and they would not seek out a game for the sole purpose of learning about phishing.

Embedded tools may take the form of browser extensions. GoldPhish, developed by Dunlop et al. (2010), was an extension for the now deprecated Internet Explorer browser, allowing it to access the sites viewed by the user to identify phishing.

Aggarwal, et al. (2012) developed 'PhishAri', a Chrome browser extension used to detect phishing attempts on Twitter. The researchers found that phishing attacks carried out through social media sites have risen, and a common technique used is the obfuscation of malicious web links through URL shortening. The extension uses ML techniques to classify phishing URLs and tweets through characteristics of the URL, the tweet, and the author. The tool applies a red indicator to phishing tweets and a green indicator to safe tweets.

2.2 Machine Learning (ML)

The language patterns commonly reused in phishing attacks have generated interest in how ML can identify and protect users from phishing attacks due to its ability to classify data by identifying trends.

ML models require input data stored in a numerical format for processing. Data can come from a variety of sources, including images and text converted to numerical vectors. In the field of natural language processing (NLP), structured collections of text referred to as 'corpora' are used as datasets for training. Converted data can make predictions on non-numerical information, using qualities such as its visual appearance or use of language.

Fu et al. (2006) proposed a method for detecting phishing webpages by assessing visual similarities between a potential phishing site and a set of protected sites known to be legitimate. The research interpreted the colour and location of each pixel on a webpage as data when making a prediction. However, this method only detects phishing pages that look similar to those in the protected set, with less success at detecting phishing web pages outside of this set. Fu et al. (2006) cited natural language analysis to enhance the project, improving detection accuracy.

The GoldPhish browser extension by Dunlop et al. (2010) uses optical character recognition (OCR) to detect a company logo on a webpage and converts it to text. Google PageRank is then used to compare the top domains with that name to the current webpage. However, one potential issue with this method is that webpage logos may be highly stylised, rendering them difficult for OCR to interpret.

The aforementioned research has explored phishing webpages, which contain more graphical content than phishing emails. Image-based phishing detection is less flexible than text-based detection. It can only detect images similar to those used during model training and is dependent on the accuracy of external technologies, such as OCR. Thus, it is essential to focus on the text content using sentiment analysis, which has been applied to other contexts.

Tao and Fang (2020) proposed a multi-label sentiment analysis method to determine the sentiment of online reviews for restaurants, wines and films. This method allows the sentiment towards specific aspects of a sample to be analysed, rather than producing a prediction for the overall sentiment. For example, a review for a restaurant may express a positive sentiment towards the food but a negative sentiment towards the atmosphere.

While emails may contain some common features, such as greetings and sign-offs, these are not present in all emails. Also, compared to descriptions of specific features of an object, such as a wine's variety or country of origin, these email features are more abstract and may be more difficult for an ML model to identify. However, this method used a multi-class approach, allowing

samples to be classified as positive, negative, neutral or conflicted (both positive and negative). Such an approach may be applicable when identifying phishing emails, as it may produce more accurate results, considering different types of phishing attempts.

Other important factors to consider relating to the dataset used in training are its quality, size, and format. Halgaš et al. (2019) proposed a phishing classifier that uses a recurrent neural network (RNN) to evaluate an email's text and structure. Researchers highlighted the ability of phishing emails to avoid filters due to their changing nature and suggest that ML may be able to identify trends in phishing emails. Two datasets comprised of legitimate and phishing emails sourced from existing email corpora were used to train the model. Of the two datasets used, the RNN classified emails more accurately when trained with the smaller and less balanced of the two datasets, demonstrating that both quantity and quality of a corpus impact a model's accuracy. This method classified emails as either 'ham' (legitimate) or phishing. However, this binary classification system may have impacted the model's accuracy, given the many differences in language used in the numerous types of phishing attacks, such as extortion compared to unexpected money fraud.

Prusa et al. (2015) investigated the correlation between the size of a training dataset and the accuracy of a sentiment analysis classifier, explicitly studying the number of instances required to train a tweet sentiment classifier. The researchers found that as the size of the dataset used for training increased, the accuracy of the machine learning model improved. However, there was no significant improvement in the accuracy of this classifier after the use of a dataset containing 81,000 instances. The sentiments of tweets were classified as either positive or negative, which are very general terms (Prusa et al., 2015).

ML techniques can be applied to the field of usable security. Given the increased need for usable, anti-phishing tools and the ability of ML to detect patterns in data, this highlights the potential for these research areas to be combined, thus protecting users and enhancing phishing detection. In the following section, the methodology behind the research is outlined, explaining how an ML model was integrated into an anti-phishing browser extension to support end-users.

3. METHODOLOGY

An ML model was trained to classify emails as phishing or legitimate and was designed to produce a classification prediction based on an email's text contents. The browser extension operated by reading and processing selected text to generate an output in a popup window.

The browser extension and the ML model were integrated into a single extension named MailTrout (Figure 1). The browser extension selected and read text from the browser window and converted the text into a numerical sequence for processing. The ML model then generated a prediction based on the sequence. Finally, the browser extension displayed an output based on the prediction of the ML model.

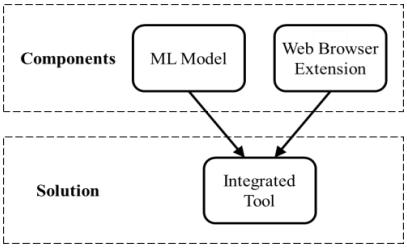

Figure 1: *Components within MailTrout*

3.1 Machine learning model

The ML model was developed using Python 3, the Python deep-learning library Keras (Chollet, 2015) and the open-source ML library, TensorFlow (Google, 2020a).

3.1.1. Algorithm selection
Artificial neural networks (ANNs) are computational algorithms based on the model of biological neurons in the human brain. ANNs can be used in ML to process input data and produce an output, such as a classification or prediction (Chen et al., 2019). Recurrent neural networks (RNNs) are variants of ANNs. The results of previous items in a sequence – such as words in a text – are stored to provide contextual information and produce results based on both the current and previous input. This method is ideal for NLP as it can evaluate the sentiment of text overall by evaluating words individually as well as in their context by considering the impact of the previous text (Lai et al., 2015).

Long Short-Term Memory networks (LSTMs) are an RNN architecture designed to cope with long-range dependences. As the distance between previous information and present input data grows, traditional RNNs become less effective at connecting this information to apply context. However, LSTMs are more capable of learning long-term dependencies, as they use multiple neural network layers to pass the neuron's output value and a memory cell state along the network, providing contextual information that can influence the output value at each stage. Due to this technique, LSTMs are shown to outperform standard RNNs at learning context-free and context-sensitive language (Gers & Schmidhuber, 2001; Sak et al., 2014).

Bidirectional Long Short-Term Memory networks (BLSTMs) further improve the ability to learn long-term dependencies. BLSTMs operate on the input sequence from both directions, allowing the network to incorporate context from before and after the present item in a sequence. This method has proven to be powerful in tasks involving NLP, including sentiment analysis and classification (Wang, et al., 2015). For these reasons, the ML model was designed to use a BLSTM layer to process data.

3.1.2. Classification

Binary-class models allow data to be classified as one of two categories, typically 'positive' or 'negative'. While this approach could be applied in this research, the issue of the many differences in phishing email patterns and vectors had to be considered. Avanan's Global Phish Report (2019) classified the phishing emails reviewed into four vectors: spearphishing, extortion, credential harvesting and malware phishing.

Spearphishing attacks - phishing targeted at a specific individual, such as a high-level employee in an organisation - were commonly found to impersonate senior employees such as CEOs. Spearphishing uses social engineering to urge their victim to complete a task, such as granting the attacker access to company information or finances. This form of attack is known as business email compromise.

Extortion attacks use threats to pressure their victim, e.g. threatening to share compromising information, holding them to a cryptocurrency-based ransom. These emails often use email spoofing techniques and passwords uncovered from data leaks to add credibility to their claims.

Credential harvesting attacks aim to steal sensitive information from their victims, such as passwords or bank details. These attacks commonly impersonate trusted brands and lead the victims to phishing webpages, using social engineering to create a sense of urgency.

Malware phishing attempts seek to install harmful software on a victim's device. These exhibit characteristics similar to the aforementioned attacks.

Postolache and Postolache (2010) also identified numerous phishing vectors, including extortion and the impersonation of legitimate organisations and individuals. However, they also identified numerous vectors not covered by these terms, including advance-fee, lottery and investment fraud. These are examples of unexpected money and winnings scams, in which a scammer attempts to make a victim believe that they can receive a financial or material reward by following their instructions, such as by sharing their bank details or paying an upfront fee (Australian Competition & Consumer Commission, 2015). These investigations highlight the broad range of phishing email vectors in use and pose an issue for an ML model; as classification predictions are most accurate when items of a class have more similarities, a model's accuracy may be hindered by large differences in the data.

To reduce issues with accuracy, a multi-class approach was chosen for the ML model, in which text could either be classified as legitimate (HAM) or one of four classes of phishing: impersonation phishing (IMP), business email compromise (BEC), extortion (EXT) or unexpected money/winnings scams (UNX). This method ensured that data used for training could be sorted into classes of as little variance as possible. The approach helped ensure the ML model's accuracy, allowing the finished product to produce information specifically relevant to the type of phishing email that the user had likely received.

Finally, the ML model used the softmax function to output results as a probability distribution. Softmax normalises output by converting a vector of numbers to values between 0 and 1 that have a sum of 1, allowing each result to be interpreted as a probability (Goodfellow et al., 2016). This approach allows the model to output the certainty of its result, which may be helpful to a user when considering if they should follow the actions recommended by the browser extension in response to an email message they have received.

3.1.3. Datasets

The Fraud Email Dataset published by Verma (2018) was included in the final dataset used to train the ML model. Verma's dataset contains fraud emails described as 'Nigerian fraud' (advance-fee scam) taken from the CLAIR collection of fraud email (Radev, 2008), and legitimate emails taken from the dataset of Hillary Clinton's emails released by the US Department of State (Kaggle, 2019).

Verma's dataset was chosen as it required little formatting or review; the data did not contain any email header information, only the body content, which the ML model was designed to process. Also, all items were labelled as either fraud (1) or legitimate (0), allowing for easy relabelling to UNX and HAM respectively, for, compatibility with the ML model's multi-class system. Additionally, the Python Reddit API Wrapper (PRAW) was used to collect extortion phishing emails posted on a series of Reddit threads titled "The Blackmail Email Scam" (EugeneBYMCMB, 2019). Suitable entries were labelled as EXT and added to the final dataset.

Online records of phishing emails are commonly presented in the format of screenshots rather than plaintext copies. In response to this, a script was developed to use OCR technology to read and store the text of saved images of emails. The Python script 'Google Images Download' was used

to download results of online image searches for examples of phishing emails. This script allowed for multiple prefix and suffix terms to be added to a keyword for individual searches. The approach allowed for greater automation of image acquisition by appending names of well-known banks and commerce platforms to a search of "impersonation phishing email". The script also allowed for colour filters to be applied to searches, which was used to specify black-and-white images for forms of phishing that were unlikely to include colours or images (Vasa, 2019).

The image results required manual review as many were not suitable, including infographics and images on the subject of email phishing. The suitable images were then compiled into folders manually, separated by their classifications. The free OCR engine Tesseract was used to interpret the text from the images (Google 2020b). The Python wrapper tool PyTesseract was used to include Tesseract as part of a Python script (Lee, 2020).

All emails (Table 1) were compiled into one large CSV file.

Table 1: *Different types of phishing email in the dataset.*

Email Category	**Count**
Business Email Compromise (BEC)	*391*
Extortion (EXT)	*1427*
Legitimate (HAM)	*5287*
Impersonation (IMP)	*541*
Unexpected Money/Winnings (UNX)	*3581*
Grand Total	***11227***

3.1.4. Training

The email text from the dataset was split into portions for training and validation of 80% and 20%, respectively, following the commonly used Pareto Principle (McRay, 2015).

High-frequency words that consume processing time but do not contribute to sentiment were filtered from the dataset. These words are known as stop words. The Natural Language Toolkit (NLTK) is a Python library used for NLP and includes a corpus of stop words, including "the", "a", and "also" (Bird et al., 2009). This corpus was used as part of the ML training script to find and remove all stop words present in the training data.

Words in the dataset were converted to integers using a tokenizer, which converts text into meaningful data or 'tokens'. The tokenizer used was included in the Keras Python library and was created using the 10,000 most reoccurring words in the dataset vocabulary. The tokenizer was exported as a JSON file so that it could be used later. Both the training and validation sequences were tokenized, and a separate tokenizer was used to convert the data labels to integers (Google, 2020c).

The sequences used to train the model had to be equal in size, meaning that sequences had to be padded or truncated to fit a set length. Sequence padding involves adding zeros to a sequence until it is the desired length. This can be done from the beginning (pre-padding) or the end (post-padding. The sequences were pre-padded, as this method has been shown to produce the most accurate results when used with an LSTM model (Dwarampudi & Reddy, 2019).

The standard sequence length chosen was 500 words. The mean word count of the emails used in training was 201, and the standard deviation approximately 266. The sequence length was calculated as the mean plus one standard deviation rounded to the nearest hundred. On inspecting the distribution of word counts of the emails, it was confirmed that this length was suitable, as most emails were within this range. Emails with a word count greater than 500 were truncated. Truncation can be carried out from the beginning (pre-truncation) or the end (post-truncation) of the sequence. There is no widely accepted best practice for sequence truncation for LSTMs; therefore, post-truncation was used to avoid removing keywords or phrases commonly located near the beginning of phishing emails, e.g. "Dear Customer".

3.2 Web browser extension

The browser extension was designed to help the user classify an email as legitimate or phishing, using the ML model. The extension was developed for use in Google Chrome, which has the largest share (StatCounter, 2021).

Figure 2: *Screenshot of the browser extension.*

The browser extension uses a complementary colour palette of turquoise and gold, ensuring that

text and buttons are high contrast and easy to distinguish visually. The extension also uses green and red icons to highlight what the user should and should not do in response to receiving a potential phishing email. Green is commonly associated with safety, while red is associated with danger. The extension uses this recognisable colour scheme to make the meaning of its messages clear. Screenshots of the prototype were tested using 'Coblis', an online tool that allows the user to view an uploaded image as a person would see it with a colour vision deficiency (CVD) or colour-blindness (Wickline, 2001).

3.2.1. Implementation

In order to select an email to be evaluated by the extension, the user highlights text using their cursor and then selects a button on the extension popup. This method was deemed the most transferable for use in different web email clients as it did not require email contents to be automatically detected, ensuring that the extension processed only the text a user wanted to evaluate. The extension popup displays on the right side of the page; this is common practice for web browser extensions and would be familiar to the user. This also prevents disruption to the user's browsing experience; given that the user is likely to have left-aligned text on a page, the popup will not cover any important parts of the text.

3.2.2. Readability

To ensure that users could easily understand the instructions, certainty of classification, and information given by the extension, the Python package 'TextStat' was used to evaluate the readability and complexity of the text in the extension's instructions and results (Bansal & Aggarwal, 2020).

TextStat can be used to produce a readability score using numerous established readability formulas. The Dale-Chall readability formula was used to calculate the US grade level of text, which was then used to determine the average age level. According to Begeny and Greene (2014), the Dale-Chall formula outperforms other commonly used readability formulas as a consistent and accurate indicator of text difficulty. Text within the extension was written to be understandable by those aged 18 majority – the legally recognised threshold of adulthood – in most countries (UN General Assembly, 1989).

3.3 Integration of model into extension

The ML model was converted from a Hierarchical Data Format Files to the TensorFlow.js Layers format, allowing for use with JavaScript as part of the web browser extension. The Layers formatted model consisted of a JSON file of the model architecture and a binary weights file. The JSON file was loaded into JavaScript using TensorFlow.js, allowing the browser extension to make and output predictions using the ML model.

The browser extension was designed to use the ML model to make classifications on the client-side. This approach ensured the analysis of emails would be faster compared to loading the model from a server (Figure 3). The model was loaded from the JSON file stored by the extension and generated a prediction based on the sequence. The prediction consisted of an array of probabilities that the sequence was one of the potential email categories. The browser extension then displayed a result based on the classification with the highest probability.

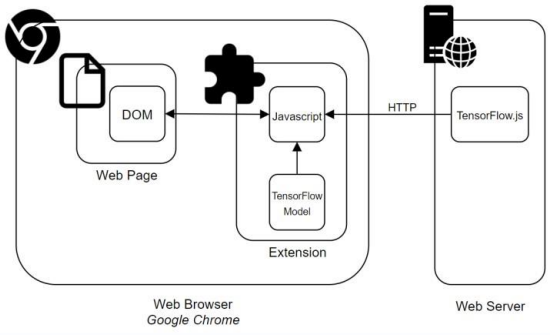

Figure 3: *System architecture.*

3.4 User testing

User acceptance testing was carried out to evaluate the browser extension's usability, with a total of 44 participants. Testing was conducted remotely owing to the COVID-19 pandemic and lockdown. Participants emailed the researchers indicating interest and were provided with a copy of the extension, along with an instructional YouTube video containing installation details.

An online survey used to record participants' feedback on the extension's usability was developed. Participants had to agree to an informed consent statement before proceeding with the experiment and were asked to provide demographic information. Participants were also asked about their familiarity with the terms used to describe the four categories of phishing emails. They were then given a fuller description of each category and asked to rate how likely they would be to identify an email of that category.

A scenario was given to participants to add a level of realism to the testing environment. This scenario stated that the participant was working for an organisation and had been asked to review their boss's email inbox using the MailTrout extension to identify phishing emails received. A webmail-style sandbox environment created using HTML, CSS, JavaScript, and PHP was developed and displayed

to participants in the browser. The inbox randomly selected 10 test emails out of a possible 30 and displayed these to the participant one at a time, moving to the next email once the participant marked each as either legitimate or phishing. Once they completed the task, they were asked to consider how usable they found the extension, using an all-positive version of the System Usability Scale (SUS) as discussed by Sauro & Lewis (2011).

After using the extension, participants were again asked how likely they would be to identify phishing emails of each category. Additional questions explored how helpful they found the instructions provided for using the extension and how likely they would be to recommend the extension to someone looking to protect themselves against phishing emails. Participants were also asked to provide feedback on how the extension catered to any conditions they had which may impact their ability to use a browser extension, such as a specific learning difficulty (SpLD), CVD, or visual impairment. Participants were also given the opportunity to provide any other feedback they had about the extension overall.

4. RESULTS

Results showed that overall, the ML model classified emails accurately, and test participants were content with the usability of the extension. Additionally, they found it simple to use and felt it educated them on the techniques commonly used in phishing emails.

4.1 Model accuracy

The model was trained with a sequence size of 500 words, using pre-padding and post-truncation to reach this standard size. The size of 500 words was chosen because the majority of emails in the dataset fell within this range. Overall, the dataset contained 11227 records.

The model produced: 5930 true positives (TPs), 5287 true negative (TNs), 0 false positives (FPs), and 10 false negatives (FNs). The true categories of emails were recorded when counting FNs to understand the model's accuracy when classifying categories of phishing emails (Table 2).

Table 2: *Total number of emails in the dataset and FNs.*

Email Category	Total	No. of FNs
Unexpected Money/Winnings (UNX)	*3581*	*0*
Extortion (EXT)	*1427*	*0*
Impersonation (IMP)	*541*	*7*
Business Email Compromise (BEC)	*391*	*3*

4.2 User acceptance

To evaluate the usability of the extension, 44 participants (23 male, 21 female) over the age of 18 years were recruited for the pilot study. Participants ranged in age from 18-69 (with 59% falling into the 18-24 bracket), and varied in level of education, field of study, and country of residence.

Overall, participants' answers to the SUS questionnaire gave the MailTrout extension a score of 87.5 out of 100. Notably, younger participants gave the extension a higher usability rating than older participants. Only one participant reported being in the age range of 40-54. The two highest age ranges (40-54 and 55-69) were combined into one range of 40-69 to aid in presenting and interpreting the results. Participants aged 18-24 gave the extension the highest usability score on average, while those aged 40-69 gave the extension the lowest usability score. While the ratings received overall were positive, these findings demonstrate that older participants may have found the tool less usable (Figure 4).

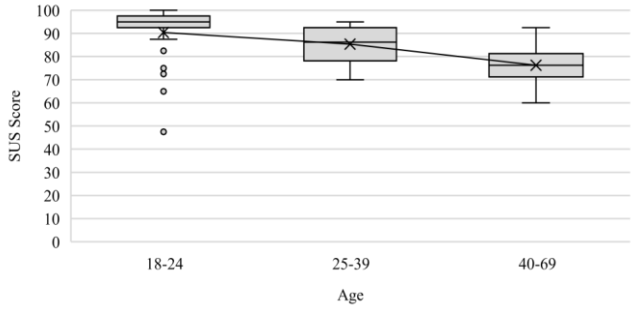

Figure 4: *SUS score by age bracket.*

Many participants remarked on how easy they found the extension to use and understand, describing it as refined and straightforward. Participants also found the speed and ability of the extension impressive. A common view among participants was that the extension was well designed, and the text was easy to read and understand.

Several participants expressed that the extension would be helpful to those who are less confident online and perhaps more vulnerable to phishing emails, such as the elderly. Another emerging theme was that participants said they would recommend MailTrout to people they knew who commonly receive phishing emails. Overall, participants felt they would be very likely to recommend the extension to someone looking to protect themselves against phishing emails, scoring their likelihood an average of 4.59 out of 5.

Before testing the extension, users were asked to rate their familiarity with terms describing the types of phishing emails. They were then given a description of each category and asked to rate the likelihood that they would be able to identify an email of each category. After using the extension, they were asked to rate the likelihood again, exploring if

their level of knowledge increased. Results showed that participants knowledge of phishing emails and the associated categories improved post-test.

Some participants raised issues with the extension's functionality. Some reported that the result produced was more accurate if more text was selected for analysis. Therefore, users could receive less accurate results if they omitted some words when highlighting an email's text. Participants also expressed issues with interaction, notably the need to highlight text and click the extension icon. Others argued the extension often flagged emails as phishing where there were no typical characteristics of phishing attacks present in the text, such as requests for information or money or when the email appeared to have been sent by a trusted individual.

The results of the user testing were recorded to evaluate the accuracy of participants' classification of emails either as legitimate or phishing. These results are displayed as a confusion matrix – a table of the number of correct and incorrect predictions generated (Figure 5). Confusion matrices can be used to visualise accuracy and can include performance metrics. Using the FP and FN rates calculated, a confusion matrix was developed to present the participants' accuracy.

	Actual Positive 542	**Actual Negative** 125
Predicted Positive 525	**True Positive** *(TP)* 504	**False Positive** *(FP)* 21
Predicted Negative 142	**False Negative** *(FN)* 38	**True Negative** *(TN)* 104

Figure 5: *Confusion matrix of participant classifications.*

The specific categories of phishing emails shown during experiments were also recorded to determine the number of phishing emails erroneously marked legitimate (false negatives), as shown in Table 3. These results showed that BEC emails had the largest number of FNs, suggesting they may have been the category detected with the least accuracy.

Table 3: *FNs per category during user testing*

Email Category	**No. of Emails**	**No. of FNs**
Unexpected Money/Winnings (UNX)	*124*	*1*
Extortion (EXT)	*97*	*1*
Impersonation (IMP)	*143*	*5*
Business Email Compromise (BEC)	*178*	*31*

5. DISCUSSION

This section discusses the accuracy and success of the ML model used by MailTrout and the usability of the integrated solution as a security tool.

5.1 Model

Using the FP and FN rates of PILFER and SpamAssassin as shown in Table 1 (Fette et al., 2007), categories were devised to rank the success of the ML model.

Table 4: *Categories of ML Model FP and FN Rates***.**

Category	**False Positive (FP) Rates**	**False Negative (FN) Rates**
'Excellent'	≤ 0.0012	≤ 0.036
'Good'	> 0.0012, ≤ 0.00135	> 0.036, ≤ 0.0715
'Average'	> 0.00135, ≤ 0.0022	> 0.0715, ≤ 0.13
'Poor'	> 0.0022	> 0.13

$$FP\ Rate = \frac{FP}{FP+TN} \qquad FN\ Rate = \frac{FN}{FN+TP}$$

Figure 6: *Formulae for false positive (FP) and false negative (FN) rates.*

Using the formula shown in Figure 6, the FP of the MailTrout model was 0.0, with an FN of 0.00168, demonstrating the model's accuracy.

In comparison to existing research, the model outperformed other ML-based phishing detection methods. As shown in Table 4, the email classifier PILFER combined with a feature using the spam filter SpamAssassin developed by Fette et al. (2007) had an FP rate of 0.0013 and an FN rate of 0.036, while the trained SpamAssassin filter alone had an FP rate of 0.0012 and an FN rate of 0.130. The most accurate RNN phishing classifier developed by Halgaš et al. (2019) had FP and FN rates of 0.0126 and 0.0147, respectively.

However, these findings are somewhat limited by issues with the dataset. Firstly, due to the lack of data available, 80% of emails from the same dataset were used for training, with 20% for testing, following the Pareto Principle (McRay, 2015). Since the training and testing emails were from the same dataset, the model's familiarity may cause it to produce more seemingly accurate results than it would on unseen data. Models may learn the details of the training data with such specificity that they cannot make more general predictive rules that can be applied to new datasets, in an issue known as 'overfitting' (Dietterich, 1995). Due to the use of the same dataset for training and testing, the results of

this study may suggest that the model is more accurate than it would be in a practical setting.

The dataset's quality may have been negatively impacted by the methods used to collect data or issues with the existing corpora. The Fraud Email Dataset (Verma, 2018) used as part of the training dataset contained some email metadata such as the date and time that emails were sent and encoded text for use with older email servers. This data was not valuable for training and may have caused the ML model to overfit or fail to identify words and phrases correctly. The dataset also had a lack of variety of legitimate emails; as the legitimate emails used all came from the released dataset of Hillary Clinton's emails (Kaggle, 2019), they may not have been reflective of the average email user's inbox.

When using the Python Reddit API Wrapper to extract comments from a Reddit thread of extortion emails (EugeneBYMCMB, 2019), some unrelated comments were extracted and added to the dataset. This was due to the thread containing general comments from users introducing or discussing the emails shared. Also, the OCR technology used to extract text from images of phishing emails may have produced inaccurate results due to an inability to understand stylised text or navigate unusual text layouts. The presence of text added to images to highlight common signs of phishing attacks may also have been picked up by OCR technology.

5.2 User acceptance

Considering the SUS adjective ratings proposed by Bangor et al. (2009), the SUS score of 87.5 given to the extension can be described as 'excellent', and highlights that the extension met its aim of being a usable security tool.

Participants reported that they found the extension easy to use and understand. One participant suggested that users would be more likely to keep using MailTrout due to the extension's accessibility and embedded nature.

- *"I like how easy it is to use, it's always in the corner so it isn't a complicated process that people will give up on easily"*

Participants also remarked how impressed they were with the functionality of the extension.

- *"There are certain things in the tone of an email that I would not have flagged had it not been for the extension"*

Commenting on the design and layout of the extension, one participant with strong colour vision deficiency (CVD) reported the colour scheme provided a high level of contrast and therefore had no issues using it. Other participants with specific learning difficulties (SpLDs) found the extension accessibly designed with a simple layout, colour-coding, and succinct information.

- *"I am extremely colour blind (strong deuteranopia) and had absolutely no issues using the web extension and found each colour to clearly stand out from its surroundings"*

- *"I have dyslexia which makes using some text-based extensions difficult, this extension and the colour coded nature of the help box layout made it very accessible to use. Additionally the lists of what to look out for were to the point and easy to understand"*

Participants also suggested that the extension could educate people on identifying phishing emails themselves, reporting that the information on what to look out for, what to do and what not to do was a particularly good feature.

- *"The Look Out/Do/Don't is a really good feature, as the user is learning as they use [the extension] rather than just relying on a traffic light system."*

The responses to each statement in the SUS survey were very positive overall, generally averaging between 'Agree' (4) and 'Strongly Agree' (5). However, the average response to the first statement was found to be lower than that of all others. The first statement read *"I think that I would like to use this extension frequently*".

A possible explanation is that while participants provided positive feedback on the extension overall, they did not feel that they needed it themselves due to their ability to identify phishing emails unaided. This can be understood further using the theory of diffusion of innovations (DOI), which explores how new ideas and technologies are adopted. One of the characteristics of an innovation is its 'relative advantage', meaning the degree to which the innovation is perceived as better in comparison to existing measures. If a user perceives the relative advantage of an innovation as low, they will be less likely to adopt it (Rogers, 2003). Therefore, if participants believed they were able to identify phishing emails themselves with high levels of accuracy, they may have felt the relative advantage of using the extension was low. Therefore, they felt less likely to adopt the extension.

Participants' ratings of their ability to spot phishing emails (where 1 is poor and 5 is excellent) were compared to their answers for SUS statement 1 to understand this finding further. As hypothesised, participants who answered that they would not use the extension frequently reported they had a strong ability to spot phishing emails, suggesting that they would find using the extension unnecessary.

Another characteristic of innovation is 'observability', meaning the degree to which the effects of the innovation are visible. If a user perceives the visible

results of innovation as low, they will be less likely to adopt it (Rogers, 2003). In DOI theory, 'preventative innovations' aim to lower the probability of an unwanted future event. Preventative innovations tend to take longer to be adopted by users due to the lack of observable impact of their use. However, if a user experiences a 'cue-to-action' – an event that causes them to undergo a behavioural change – then this can result in a more favourable attitude towards an innovation (Xiao, et al., 2014).

Security tools such as MailTrout may be considered preventative innovations as they aim to lower the probability of security failures, such as a user falling victim to phishing emails. Therefore, users may be more likely to adopt the extension if they have experienced a cue-to-action, such as becoming the victim of a phishing attack.

Participants were shown to have an increased knowledge of types of phishing email after using the extension. Average familiarity ratings for each phishing email category increased as participants used the extension to learn what to look for in phishing emails. These findings demonstrate the extension's potential ability to educate users about identifying phishing emails in the long term.

The results also demonstrated a correlation between a participant's SUS rating and their demographic characteristics. Firstly, participants studying or working in a formal sciences subject, such as computing science, found the extension more usable than those in other subject areas.

A potential explanation for this result may be that those employed in formal science fields are more frequent users of computers and are therefore more comfortable learning how to use new tools. Participants in formal sciences may have had more experience, specifically with web browser extensions and would find learning to use the extension far less challenging than someone who has never used a browser extension before. They may also have had more exposure to phishing emails, especially if they are involved in cybersecurity, which may also have given them an advantage over users who are less familiar with the terms and techniques often associated with email phishing. Participants in formal sciences demonstrated an overall higher familiarity with categories of phishing emails than those in other fields throughout the experiment. However, it is important to note that the average SUS scores of each subject field were all in the 'excellent' category. Hence, differences between subject fields are a minor concern.

Younger participants found the extension more usable than older participants. The average SUS scores of the 18-24 and 25-39 age ranges were in the 'excellent' category, while that of the 40-69 range was in the 'good' category. While these ratings are positive, it demonstrates that older participants found the tool less usable. A potential reason for this may be that participants who were born after the 1980s – commonly referred to as 'digital natives' – are more likely to have grown up around digital technology and so have been familiar with computers from an early age. Conversely, users born before the 1980s – commonly referred to as 'digital immigrants' – grew up before the widespread use of digital technology and have not had the same experience, thus making it harder for them to learn how to use new technologies (Prensky, 2001). Younger participants may also have had more experience using browser extensions and dealing with phishing emails - this group demonstrated an overall higher familiarity with categories of phishing emails throughout the experiment than older participants. However, consideration should be given to the limited number of older participants who took part in the research, thus further work is required to explore this result.

6. CONCLUSION AND FUTURE WORK

The research showed that due to the presence of common words and sentiment patterns across phishing emails, and the ability of ML algorithms to classify data by detecting recurring patterns, ML technology is well-suited to the task of identifying phishing emails. Additionally, the web browser extension format provided a suitable way to create an embedded learning tool, providing users with an opportunity to use the extension while completing everyday tasks, such as checking their emails. The extension demonstrated high levels of usability and accuracy when detecting phishing emails.

These findings also indicate that browser extensions can act as accessible security tools, requiring limited technical knowledge to use and can easily be incorporated within a person's routine online activities. Due to their simplicity and embedded nature, browser extensions may be beneficial for those with less experience of using the internet.

Furthermore, the COVID-19 pandemic has caused a fundamental shift in our lives, heightening the pace at which society adopts and utilises digital technologies. The increased reliance on digital communications means that more people may be likely to encounter phishing emails. Thus, more research is required in this field to protect potentially vulnerable citizens.

7. REFERENCES

Australian Competition & Consumer Commission. (2015) Types of scams. https://www.scamwatch.gov.au/types-of-scams (Retrieved 13 March 2019).

Aggarwal, A., Rajadesingan, A., and Kumaraguru, P. (2012) PhishAri: automatic realtime phishing detection on twitter. 2012 eCrime Researchers Summit, Las Croabas, PR, USA, 23-24 October 2012, pp. 1-12. IEEE.

Avanan. (2019) Global Phish Report. https://www.avanan.com/global-phish-report (Retrieved 7 February 2020).

Bangor, A., Kortum, P. & Miller, J. (2009). Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *Journal of Usability Studies,* 4(3), pp. 114-123.

Bansal, S. & Aggarwal, C. (2020) *textstat 0.6.0.* https://github.com/shivam5992/textstat (Retrieved 3 March 2020).

Begeny, J. C. & Greene, D. J. (2014). Can Readability Formulas Be Used to Successfully Gauge Difficulty of Reading Materials?. *Psychology in the Schools,* 51(2), pp. 198-215.

Bird, S., Klein, E. & Loper, E. (2009). 2.4.1 Wordlist Corpora. In: J. Steele, 1st ed. *Natural Language Processing with Python.* Sebastopol: O'Reilly Media, Inc., pp. 60-62.

Chen, Y.Y., Lin, Y.H., Kung, C.C., Chung, M.H., Yen, I. (2019). Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in DemandSide Management for Smart Homes. *Sensors (Basel),* 19(9).

Chollet, F. (2015) *Keras Documentation.* https://keras.io/ (Retrieved 14 March 2020).

Dietterich, T. (1995) Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR),* 27(3), pp. 326-327.

Dixon, M., Gamagedara Arachchilage, N.A. and Nicholson, J. (2019) Engaging users with educational games: The case of phishing. In Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems (pp. 1-6).

Dunlop, M., Groat, S. and Shelly, D. (2010) Goldphish: using images for content-based phishing analysis. 2010 Fifth international conference on internet monitoring and protection, Barcelona, Spain, 9-15 May 2010, pp. 123-128. IEEE.

Dwarampudi, M. & Reddy, N. V. S. (2019) Effects of padding on LSTMs and CNNs. https://arxiv.org/pdf/1903.07288.pdf (Retrieved 5 February 2020).

EugeneBYMCMB. (2019) The Blackmail Email Scam (part 3) : Scams. https://www.reddit.com/r/Scams/comments/biv65o/the_blackmail_email_scam_part_3/ (Retrieved 21 January 2020).

Fette, I., Sadeh, N. & Tomasic, A. (2007) Learning to detect phishing emails. 16th International World Wide Web Conference, Banff, Canada, May 2007, pp. 649–656. ACM.

Fu, A. Y., Wenyin, L. & Deng, X. (2006). Detecting Phishing Web Pages with Visual Similarity Assessment Based on Earth Mover's Distance (EMD). *IEEE Transactions on Dependable and Secure Computing,* 3(4), pp. 301-311.

Gers, F. A. & Schmidhuber, E. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks,* 12(6), pp. 13331340.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). 6.2.2.3 Softmax Units for Multinoulli Output Distributions. In: *Deep Learning.* Cambridge: MIT Press, pp. 180-184.

Google. (2020a) Tensorflow. https://www.tensorflow.org/ (Retrieved 15 January 2020).

Google. (2020b). tesseract-ocr / tesseract: Tesseract Open Source OCR Engine (main repository). https://github.com/tesseract-ocr/tesseract (Retrieved 6 February 2020).

Google. (2020c). *tf.keras.preprocessing.text .Tokenizer.* https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer (Retrieved 23 March 2020).

Graham, A. (2018) The cost of a cyber attack. IT Governance. https://www.itgovernance.co.uk/blog/the-cost-of-a-cyber-attack (Retrieved 7 February 2020).

Halgaš, L., Agrafiotis, I. & Nurse, J. R. C. (2019). *Catching the Phish: Detecting Phishing Attacks using Recurrent Neural Networks (RNNs).* Jeju Island: 20th World Conference on Information Security Applications, Springer.

Horgan, S., Collier, B., Jones, R., Shepherd, L. (2021) Re-territorialising the policing of cybercrime in the post-COVID-19 era: towards a new vision of local democratic cyber policing. Journal of Criminal Psychology, *Accepted/In Press*.

Kaggle. (2019) Hillary Clinton's Emails. https://www.kaggle.com/kaggle/hillary-clinton-emails/ (Retrieved 15 March 2020).

Kumaraguru, P., Sheng, S., Acquisti, A., Cranor, L.F., Hong, J. (2010). Teaching Johnny not to fall for phish. *ACM Transactions on Internet Technology (TOIT),* 10(2), pp. 1-31.

Lai, S., Xu, L., Liu, K. & Zhao, J., (2015). Recurrent convolutional neural networks for text classification. *Proceedings of the National*

Conference on Artificial Intelligence, Volume 3, pp. 2267-2273.

Lallie, H. S., Shepherd, L. A., Nurse, J. R. C., Erola, A., Epiphaniou, G., Maple, C., & Bellekens, X. (2021) Cyber security in the age of COVID-19: a timeline and analysis of cyber-crime and cyber-attacks during the pandemic. Computers & Security, 105. 102248.

Lee, M. (2020) madmaze/pytesseract: A Python wrapper for Google Tesseract https://github.com/madmaze/pytesseract (Retrieved 6 February 2020).

McRay, J., 1st ed., (2015). Pareto principle. In: *Leadership glossary: Essential terms for the 21st century.* Santa Barbara: Mission Bell Media.

Postolache, F. & Postolache, M. (2010). Current and Ongoing Internet Crime Tendencies and Techniques. Preventive Legislation Measures in Romania. *EIRP Proceedings,* 5(1), pp. 35-43.

Prensky, M., (2001). Digital Natives, Digital Immigrants. *On the Horizon,* 9(5), pp. 1-6.

Prusa, J., Khoshgoftaar, T. M. & Seliya, N. (2015). *The Effect of Dataset Size on Training Tweet Sentiment Classifiers.* Miami: IEEE 14th International Conference on Machine Learning and Applications (ICML), IEEE.

Radev, D. (2008). CLAIR collection of fraud email, ACL Data and Code Repository, ADCR2008T001.https://aclweb.org/aclwiki/CLAIR_collection_of_fraud_email_(Repository) (Retrieved 15 February 2020).

Rogers, E. M. (2003). *Diffusion of Innovations.* 5th ed. New York City: Simon and Schuster.

Sak, H., Senior, A. & Beaufays, F. (2014). *Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling.* 15th Annual Conference of the International Speech Communication Association, Singapore, ISCA Archive.

Sauro, J. & Lewis, J. (2011). When designing usability questionnaires, does it hurt to be positive?. *Proceedings of the SIGCHI Conference on human factors in computing systems*, 7 May, pp. 2215-2224.

StatCounter. (2021) Browser market share worldwide. https://gs.statcounter.com/browser-market-share (Retrieved 6 May 2021).

Tao, J. & Fang, X. (2020). Toward multi-label sentiment analysis: a transfer learning based approach. *Journal of Big Data,* 7(1), pp. 1-26.

UN General Assembly. (1989). Convention on the Rights of the Child. *United Nations, Treaty Series,* Volume 1577, p. 3.

Vasa, H. (2019) Google Images Download. https://github.com/hardikvasa/google-images-download (Retrieved 15 March 2020).

Verma, A. (2018). Fraud Email Dataset | Kaggle. https://www.kaggle.com/llabhishekll/fraud-email-dataset (Retrieved 28 January 2020).

Wang, P., Qian, Y., Soong, F.K., He, L., Zhao, H. (2015). Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network. *ArXiv [Preprint].* (Retrieved 15 March 2020).

Wickline, M. (2001) Coblis - Color Blindness Simulator. https://www.color-blindness.com/coblis-color-blindness-simulator/ (Retrieved 10 March 2020).

Xiao, S., Witschey, J. & Murphy-Hill, E., (2014). Social Influences on Secure Development Tool Adoption: Why Security Tools Spread. *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing,* pp. 1095-1106.