Implementing a Browser Extension to Detect Phishing Emails Using Natural Language Processing

Paul Boyle

BSc (Hons) Ethical Hacking, 2020

School of Design and Informatics

Abertay University

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

ACKNOWLEDGEMENTS

I would first like to thank my supervisor, Dr Lynsay Shepherd, for her expertise, guidance and encouragement throughout this project.

I would also like to thank all the participants who took part in my study and made this research possible.

In addition, I would like to thank Randstad Student Support for their valuable feedback and assistance.

Finally, I would like to thank my family and friends who have supported me during this project and throughout the past four years.

ABSTRACT

CONTEXT

Phishing scams can cause huge financial and reputational damage for organisations and individuals. The unpredictability of phishing emails often results in them not being detected by email client spam filters. Natural language processing may be used to identify manipulative language typical of such attacks in order to identify phishing emails and educate users.

AIM

To develop a web browser extension for Google Chrome to use machine learning to detect phishing emails displayed on the user's web browser based on their sentiment and language.

METHOD

TensorFlow was used to develop and train a machine learning model using a dataset of fraudulent and legitimate emails. This model was tested to evaluate its accuracy and converted for use in a web browser.

A Chrome browser extension was developed and tested to ensure that it could read highlighted text on a webpage and that its text was readable by the end user. This extension was integrated with the machine learning model and tested with end users to evaluate its usability.

RESULTS

The machine learning model was found to successfully detect and classify phishing emails with a good level of accuracy.

The integrated extension demonstrated that it was usable; testing participants identified that they would recommend the tool to anyone looking to protect themselves from phishing emails, and considered the tool to be especially helpful for those less confident with technology.

CONCLUSION

The project demonstrated that a machine learning model can be integrated with a web browser extension to use natural language processing to identify phishing emails based on their content. Future work for the project may include the use of a dataset with a higher quantity and variety of emails to create a more accurate machine learning model with a greater ability to classify new data.

ACRONYMS AND ABBREVIATIONS

ANN	*Artificial Neural Network*
BEC	*Business Email Compromise (CEO Fraud)*
BLSTM	*Bidirectional Long Short-Term Memory*
CVD	*Colour Vision Deficiency (Colour Blindness)*
DOI	*Diffusion of Innovations*
EXT	*Extortion Phishing*
FN	*False Negative*
FP	*False Positive*
HAM	*Legitimate Emails (Not Spam/Phishing)*
HDF	*Hierarchical Data Format*
IE	*Internet Explorer*
IMP	*Impersonation Phishing*
LSTM	*Long Short-Term Memory*
ML	*Machine Learning*
MODI	*Microsoft Office Document Imaging*
NLP	*Natural Language Processing*
OCR	*Optical Character Recognition*
PILFER	*Phishing Identification by Learning on Features of Email Received*
PRAW	*Python Reddit API Wrapper*
RNN	*Recurrent Neural Network*
SA	*SpamAssassin*
SpLD	*Specific Learning Difficulty*
SUS	*System Usability Scale*
TN	*True Negative*
TP	*True Positive*
UNX	*Unexpected Money/Winnings Scam*

1. INTRODUCTION

1.1. BACKGROUND

'Phishing' is a form of fraud in which an attacker sends emails claiming to be from a reputable individual or organisation with a view to committing malicious acts, such as stealing money or sensitive information (such as passwords or bank details) from their victims, spreading malware or causing disruption.

Successful phishing scams can be costly for their victims; a 2016 study found that the average spear-phishing attack – a form of phishing attack specifically targeted at an individual or organisation – costs an organisation $1.6 million (Cloudmark Security Blog, 2016).

To combat phishing scams, email clients make use of spam filters to send suspicious emails to quarantined folders rather than the user's main inbox. However, these filters are not always successful; worryingly, out of over 555,000 phishing emails analysed by the cloud-based security company Avanan for their 2019 Global Phish Report, 25% bypassed Office 365's security measures, resulting in them being delivered to their potential victim's inbox (Avanan, 2019). Due to the fallibility of these server-side filters, users may take precautions into their own hands through anti-phishing tools and educational materials.

Web users may make use of browser extensions to allow them to personalise their browsing experience. Browser extensions are add-ons to the user's browser of choice that can make client-side changes to alter the way webpages are displayed or give the user tools to help them when browsing the internet. Browser extensions can be used to 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 make use of 'machine learning' (ML) – a technology which allows computer systems to learn from data they are provided with in the form of datasets in order to make their own decisions without the need for human interaction. For example, machine learning has been used to determine the sentiment expressed in online reviews (Tao & Fang, 2020), as well as to detect phishing emails based on features such as the number of hyperlinks present and the use of JavaScript (Fette, et al., 2007).

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 to attempt to exploit their victims.

1.2. RESEARCH QUESTION

This project set out to investigate how a machine learning model could be integrated with a Google Chrome web browser extension to develop a security tool that could detect potential phishing emails by analysing the sentiment of their text content.

1.3. AIMS

The aims of this project were:

- To develop a ML model that can detect phishing emails with a high level of accuracy
- To integrate this model with a web browser extension to create an accessible and usable security tool for people of all levels of technical experience and confidence
- To educate users on how to protect themselves from phishing attacks by providing detailed descriptions of the model's results, such as signs that an email may not be legitimate and instructions on how users should respond if they consider an email to be suspicious.

This project does not examine the impact of using different network architectures as part of the ML model, nor that of integrating a ML model with an extension for web browsers other than Google Chrome.

1.4. STRUCTURE

Chapter 2 explores previous research related to this project, including the incorporation of usability into the design of existing anti-phishing tools, the use of ML technology to detect phishing emails, the impact of dataset quality on a ML model's accuracy, and the use of metrics to assess and compare the accuracy of a ML model.

The methodology discussed in Chapter 3 includes the process used to design, implement and test the two components of the project – the ML model and the web browser extension – followed by the approaches used to integrate these components into the final tool and carry out user acceptance testing to evaluate the tool's usability.

Chapter 4 states the results of the testing, including the readability of the text displayed by the browser extension, the accuracy of the ML model in classifying emails correctly and the usability of the integrated tool as reported from user acceptance testing.

The results presented in Chapter 4 are evaluated in Chapter 5. The findings of the project are compared to existing research to discuss the tool's success in meeting its aims.

Finally, Chapter 6 summarises the main findings of this project and sets out recommendations for further research in the field of ML and anti-phishing tools.

2. LITERATURE REVIEW

The language patterns commonly reused in phishing attacks have generated much interest in how machine learning can be used to identify and protect users from phishing attacks due to its ability to classify data by identifying trends. Previous work in this field has examined the effects of how training data is supplied to a machine learning model, how a model's accuracy can be assessed using common metrics, and how anti-phishing measures can be made into usable security tools.

This section covers existing anti-phishing tools and how they incorporate usability into their design, as well as how machine learning technology can be used to detect phishing emails and how datasets and metrics can impact and assess a machine learning (ML) model's accuracy respectively.

2.1. EXISTING ANTI-PHISHING TOOLS

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 education systems: the embedded email-based 'PhishGuru', and the online game 'Anti-Phishing Phil'. To ensure that the projects developed were effective educational materials, the developers followed a series of design principles, including the principle of 'learning-by-doing', which states that people learn better when they practise the skills that they are learning. An example of this is 'embedded training', a method used in the development of PhishGuru, in which instructional materials are integrated into the user's everyday tasks, such as checking their emails. By making the use of this tool a part of the user's everyday lives, this increases the prevalence of 'teachable moments' – optimal opportunities to convey a point or idea – increasing the tool's educational potential.

PhishGuru acts as an embedded training tool by sending simulated phishing emails to users and, in the event that they follow the links in the email, displaying advice on how they can protect themselves from phishing attacks (Kumaraguru, et al., 2010).

While tools such as PhishGuru may be especially useful in an organisational environment, such as where management wish to educate their employees on identifying and protecting themselves from phishing emails, it may not be as useful for an individual wishing to protect themselves from phishing emails with immediate results, rather than with education over an extended period.

Other embedded tools may take the form of browser extensions. GoldPhish, developed by Dunlop et al. (2010), is an extension for Internet Explorer, allowing it to easily access the sites viewed by the user (Dunlop, et al., 2010). While Internet Explorer (IE) was the most widely used internet browser at the time of this research, it is not anymore, meaning the market for this tool is now very limited.

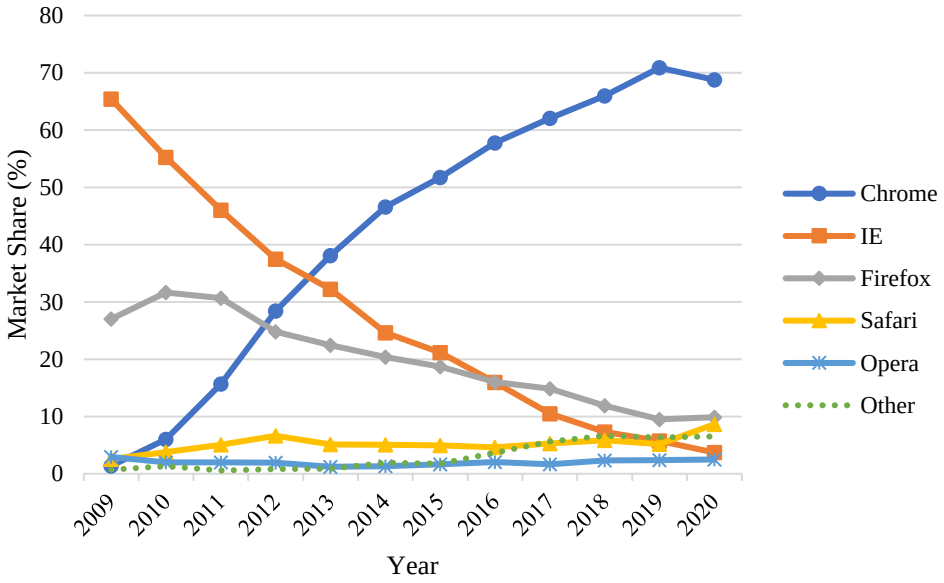


Figure 1 - Worldwide Desktop Browser Market Share (StatCounter, 2020)

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 online social media sites are on the rise, and a common technique used in these attacks is the obfuscation of malicious web links through URL shortening. The extension uses machine learning techniques to classify phishing URLs and tweets through characteristics of the URL, the tweet, and the account the tweet came from. The tool applies a red indicator to phishing tweets and a green indicator to safe tweets, using recognisable methods to display information to its users conveniently and understandably (Aggarwal, et al., 2012).

2.2. MACHINE LEARNING

2.2.1. Datasets

While machine learning (ML) models require input data to be stored as a numerical format in order to be processed, this data can come from a variety of sources, including images and text that have been converted to numerical vectors. In the field of natural language processing (NLP), structured collections of text referred to as 'corpora' (singular 'corpus') can be used as datasets for training. This converted data can be used in order to 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 the visual similarities between a potential phishing site and a set of protected sites known to be legitimate. This method works at the pixel level of webpages, interpreting the colour and location of each pixel as data to be used when making a prediction. However, this method is only able to detect phishing pages that look similar to those in the protected set, with less success at detecting phishing webpages that impersonate websites outside of this set. Fu et al. cited using natural language analysis as potential future work to this project, as this may improve the method's detection accuracy (Fu, et al., 2006).

Dunlop et al. (2010) proposed an Internet Explorer browser extension to be used to detect phishing websites. The extension – GoldPhish – uses optical character recognition (OCR) technology to detect the company logo on a webpage and convert it to text, then Google PageRank to compare the top domains of that name to the current webpage. GoldPhish was trained with 100 phishing sites and 100 verified sites, and was able to classify them with lower false positive and false negative rates than all other tools investigated by the researchers. However, one potential issue with this method is that webpage logos may be highly stylised and abstract, rendering them difficult for an OCR to interpret. The OCR software used by GoldPhish is Microsoft Office Document Imaging (MODI), which is included as part of Microsoft Office (Microsoft Support, 2019). Dunlop et al. admit that MODI is not as accurate as other commercial tools, and that the accuracy of GoldPhish depends on the accuracy of the OCR software available (Dunlop, et al., 2010).

An important consideration is that the aforementioned projects are used to detect phishing webpages, which may contain more graphical content than phishing emails and therefore may be more suitable to analyse at a pixel level than by the text present. However, image-based phishing detection in either scenario is less flexible than text-based detection, as it is only able

to detect images similar to those used during model training, and may be dependent on the accuracy of external technologies, such as OCR software.

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 simply producing a prediction for the sentiment of the text overall. For example, a review for a restaurant may be found to express a positive sentiment towards the food, but a negative sentiment towards the atmosphere. However, this approach required a team of three coders to review and label 10,000 sentences in order for this data to be used to train the machine learning model. The developers also used web-scraping techniques to collect reviews of wines and films, however, this data also required manual reviewing as some items were not suitable for labelling. Of the training sets collected, 80% were used for training the model, while 20% were used for testing (Tao & Fang, 2020).

Overall, the collection and processing of data for use in training the machine learning model proved a resource-intensive procedure, and the developers may have simplified the process by using pre-processed datasets where possible. While this method analysed online reviews, some concepts used apply to the field of analysing phishing emails, such as the 80%/20% split of data used to train and test the model respectively. However, the multi-label aspect of this method may not apply to emails, which are less easy to divide into categories. While emails may contain some common features, such as greetings, propositions 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 a machine learning 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 to group similar types of phishing email together in one class, rather than training a model to identify patterns across various styles of phishing email.

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. The researchers highlighted the ability of phishing emails to avoid filters due to their changing nature and suggest that machine learning 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. In order to ensure the greatest flexibility of the model, all the data used in training was converted to lowercase text. While this may produce more accurate results, it requires any tools that use the RNN to convert any text input to lowercase before it can be processed. 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 (Halgaš, et al., 2019).

Prusa et al. (2015) investigated the correlation between the size of a training dataset and the accuracy of a sentiment analysis classifier, specifically 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 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). The findings of this study may vary if a multi-class approach was tested rather than a binary classifier, and if the classifications of sentiment used were more specific, such as 'sad', 'angry' or 'afraid' rather than simply 'negative'.

2.2.2. Metrics

Various standard metrics can be used to evaluate the accuracy of a machine learning model. Fette et al. (2007) developed the email classifier 'PILFER' – "Phishing Identification by Learning on Features of Email Received" – which uses machine learning to detect phishing emails. The formulas presented in Figure 2 were used to classify the false positive (FP) and false negative (FN) rates of PILFER and related spam filters. TP and TN refer to the number of true positives and true negatives respectively, which are results that have been correctly identified by a classifier.

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

Figure 2 - Formulas used to find false positive (FP) and false negative (FN) rates

Using these formulas, the success rates of PILFER, as well as the spam filter SpamAssassin (SA), were calculated as shown in Table 1.

Classifier	False Positive (FP) Rate	False Negative (FN) Rate
PILFER, with SA feature	0.0013	0.036
PILFER, without SA feature	0.0022	0.085
SpamAssassin (Untrained)	0.0014	0.376
SpamAssassin (Trained)	0.0012	0.130

Table 1 - Accuracy of PILFER classifier compared with SpamAssassin (Fette, et al., 2007)

The use of these metrics allows PILFER to be easily compared to other similar classifiers, showing for example that PILFER produced more accurate results overall than SpamAssassin (Fette, et al., 2007).

In addition to FP and FN rates, Halgaš et al. (2019) measured the results of their RNN in various other metrics, namely accuracy, precision, recall and F-measure. The developers identified these as "the most popular metrics in email classifications", and used these to demonstrate that the developed classifier compared in accuracy to modern models (Halgaš, et al., 2019).

3. METHODOLOGY

This section discusses the approach used to design, implement and test the machine learning model, the web browser extension and the integrated tool as a whole. The project was comprised of the development of two components: the machine learning (ML) model and the Google Chrome web browser extension.

The ML model was trained to classify emails as phishing or legitimate and was designed for use in the project to produce a classification prediction based on a given email's text contents.

The extension was designed to allow the project to operate on a user's web browser, such as by reading and processing selected text in order to generate an output in a popup window.

These components were integrated into a single tool. The browser extension was used to select and read text from the browser window, convert the text into a numerical sequence for processing, then use the ML model to generate a prediction based on the sequence. The browser extension then displayed an output based on the prediction of the ML model.

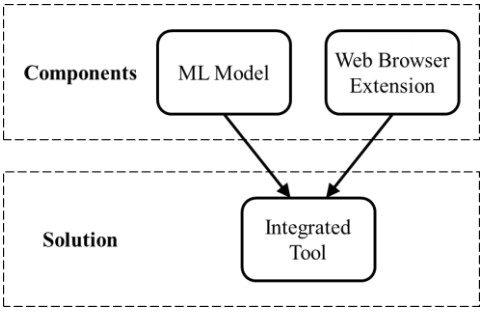


Figure 3 – Project Components and Integration

The development of the tool was carried out using the iterative waterfall model, in which each stage of development – such as the design, implementation and testing – was completed sequentially and without overlapping. This development methodology combines the traditional waterfall model – in which each stage is passed through once – with the iterative development model, allowing for phases to be revisited in future cycles of development (Trivedi & Sharma, 2013). This methodology provided a suitable approach as it was necessary to return to stages such as design and implementation following later stages of testing, such as if the machine learning model required further training following the results of system integration.

A model that avoided overlapping tasks or multiple resources being managed was suitable for this project as only one developer was working on the project, so it was not possible to delegate tasks or resources beyond this. Furthermore, stages of development – such as the integration of the ML model and browser extension – were dependant on the completion of the previous stages. The clear requirements of the project allowed for the use of a model with rigid

stages, each with specific deliverables associated. As this model was easily manageable, it allowed the project to prioritise the implementation and testing of the system.

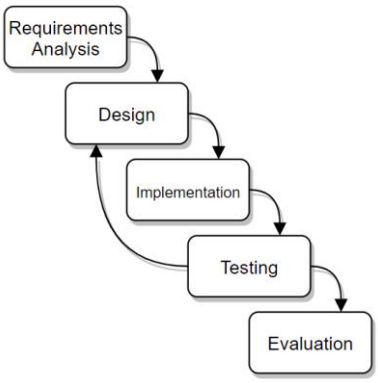


Figure 4 - An Iterative Waterfall Model

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 LLC, 2020a). This model was designed to classify emails as legitimate or as one of four categories of phishing.

This section covers the design, implementation and unit testing of the ML model.

3.1.1. Design

3.1.1.1. Algorithm

The design of the ML model began with an analysis of which type of artificial neural network should be implemented to allow the model to make the most accurate predictions. Artificial neural networks (ANNs) are computational algorithms based on the model of biological neurons in the human brain, in which information is processed using numerous connected units or 'neurons' to produce a result. 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 in which 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 a text overall, 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 more aptly than traditional RNNs; as the distance between

previous information and present input data grows, traditional RNNs become less effective at connecting this information in order to apply context and produce a more accurate result. 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 be used to influence the output value at each stage. Due to this technique, LSTMs are shown to outperform standard RNNs at learning both context-free and context-sensitive language (Gers & Schmidhuber, 2001) (Sak, et al., 2014).

Bidirectional Long Short-Term Memory networks (BLSTMs) further improve on the ability to learn long-term dependences by operating on the input sequence from both directions, allowing the network to incorporate context from both before and after the present item in a sequence. This method has proven to be very 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.

3.1.1.2. Classification

Other factors considered during the design of the model were the classes that input data could be categorised as, and the manner in which output data would be presented.

As discussed in Section 2.2.1, binary-class models allow data to be classified as one of two categories, typically 'positive' or 'negative'. While this approach could be applied to this project, the issue of the many differences in phishing email patterns and vectors had to be considered.

Avanan's 2019 Global Phish Report classified the phishing emails reviewed into four vectors: spearphishing, extortion, credential harvesting and malware phishing.

'Spearphishing' attacks – namely phishing attacks directly targeted at a specific individual, such as a high-level employee in an organisation – were commonly found to impersonate senior employees such as CEOs and to use social engineering to urge their victim into completing a task, such as granting the attacker access to company information or finances. This form of attack is known as business email compromise (BEC), or more commonly as 'CEO fraud'. While an attacker requires little technical expertise to carry out a BEC attack, they can be very expensive for organisations that fall victim to them (Mansfield-Devine, 2016).

Extortion attacks use threats to pressure their victim, such as by threatening to share compromising information about them if they do not pay a ransom, typically requested in

cryptocurrency to hide their identity. These emails often use email spoofing techniques and passwords uncovered from data leaks to add credibility to their commonly false claims.

Credential harvesting attacks aim to steal sensitive information from their victim, 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 attacks attempt to install harmful software on a victim's device, and can exhibit characteristics similar to the aforementioned techniques (Avanan, 2019).

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 (Postolache & Postolache, 2010). 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 Commision, 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 such large differences in the data. To avoid this issue, 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, ensuring the ML model's accuracy, as well as 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 was designed to use 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 useful 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.

3.1.2. Implementation

3.1.2.1. Datasets

As discussed in Section 2.2.1, ML models require training data in the form of a dataset. Due to the lack of email datasets with classifications as specific as this project required, data was compiled for a novel bespoke dataset using web scraping techniques.

This section covers the dataset used to train the ML model, comprised of data from existing corpora and data collected from web data extraction.

3.1.2.1.1. Premade Datasets

The Fraud Email Dataset published by Verma (2018) on the data science website Kaggle was included in the final dataset used to train the ML model. Verna's dataset contains fraud emails described as "Nigerian fraud" (a form of 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).

This dataset was chosen as it did not require much 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 (Verma, 2018).

3.1.2.1.2. PRAW Script

The Python Reddit API Wrapper (PRAW) allows users to extract comments from Reddit submissions using a Python script (Boe, 2020). PRAW was used to collect extortion phishing emails posted on a series of Reddit threads titled "The Blackmail Email Scam" (EugeneBYMCMB, 2019). The extracted comments were saved to a CSV file and then manually reviewed for any unsuitable entries. This review was achieved by sorting the entries by length, as the email texts shared tended to be far longer than the other comments made. Once reviewed, the entries were labelled as EXT and added to the dataset.

3.1.2.1.3. Tesseract OCR Script

During the development of the dataset, it was found that online records of phishing emails are commonly 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. This allowed for greater automation of image acquisition, such as 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 LLC, 2020b). The Python wrapper tool PyTesseract was used to allow the use of Tesseract as part of a Python script (Lee, 2020). Using PyTesseract, the script analysed each folder at a time, reading the text embedded in each image, writing the text to a CSV file and labelling each entry as appropriate.

The final results were compiled to a large CSV file, which was used as the dataset to train the ML model.

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**

Table 2 - Number of Emails in Dataset by Category

3.1.2.2. Training

The 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).

Prior to text data being used to train the ML model, high-frequency words that may take up processing time but that have no impact on a text’s sentiment were filtered out 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.

The ML model cannot be trained directly with text data, so words in the dataset were converted to integers using a tokenizer – a tool that converts text into meaningful data or 'tokens'. The tokenizer used was included in the Keras Python library and created using the 10,000 most reoccurring words in the dataset vocabulary. The tokenizer was then 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 LLC, 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) of the sequence. 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 for the emails was 500 words. The mean (average) word count of the emails used in training was 201 and the standard deviation from this value was roughly 266. The sequence length was calculated as the mean plus one standard deviation rounded to the nearest hundred, arriving at 500. On inspecting the distribution of word counts of the emails, it was confirmed that this length was suitable, as the majority of emails were within this range.

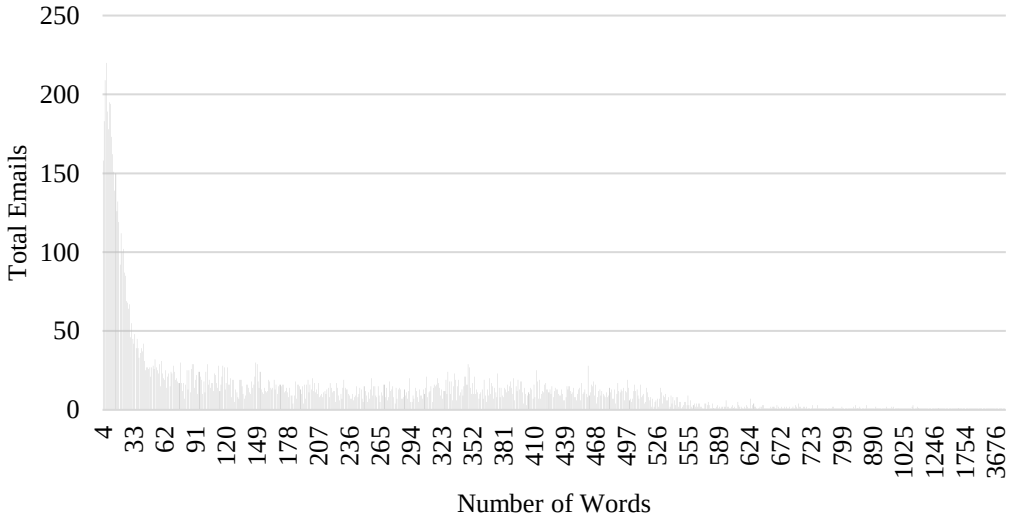


Figure 5 – Distribution of Training Emails Word Counts

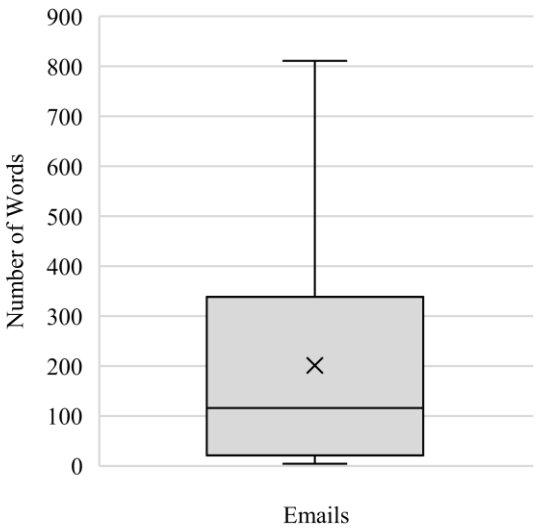


Figure 6 - Distribution of Training Emails Word Counts (cont.)

The mode (most common) word count was 7 words, while the median (middlemost value) was 116.

Emails that had a word count longer than the standard length of 500 had to be truncated, meaning that words would be removed. Similar to padding, 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.

It was decided that post-truncation would be used, so not to remove any key words or phrases commonly located near the beginning of phishing emails, such as “Dear Customer” or “I hacked your device”.

Due to the unequal split of types of email in the dataset as shown in Table 2, there was initial concern that the model’s predictions may be inaccurate. For this reason, a version of the dataset was created with more evenly split categories by removing emails. However, it was found from initial metrics during training that when the model was trained with the uneven set, it still provided good overall accuracy, so the larger and uneven dataset was used to avoid omitting emails from the training data.

3.1.3. Unit Testing

3.1.3.1. Exporting and Importing Model

It was vitally important to ensure that once the ML model had been created it could be exported for use in other systems, such as the browser extension. Once the model was trained, it was exported using the Keras ‘save’ function. The model was exported as a Hierarchical Data

Format (HDF) file, as this format allowed for the model to be converted for use with TensorFlow.js and therefore usable by the browser extension.

As discussed, a tokenizer is used to convert words from an email into integers that can be processed by the ML model. To ensure that the ML model produces predictable results, the tokenizer used in training was exported for future use as a JSON file, mapping each word with its associated key. This JSON file can be imported into Python to load the tokenizer for use with the ML model.

To test that the model and tokenizer could be exported and imported successfully, a Python script was developed to load the model, convert text to numeric sequences and make classification predictions based on the text.

3.1.3.2. Model Evaluation

In order to evaluate the model's accuracy, the FP and FN rates were calculated. Due to the multi-class system of the model, it was not possible to use existing tools to calculate these rates, and therefore they were calculated using a script. The model and tokenizer were loaded into a Python script and then used to classify a series of emails. Legitimate emails were treated as negative, while phishing emails were treated as positive. The model's predictions were compared with the correct labels to produce the number of TPs, TNs, FPs and FNs, which were then used to calculate the FP and FN rates.

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.

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

Table 3 - Categories of ML Model FP and FN Rates

If either rate was found to be below 'Good', further training would take place to improve the model's accuracy.

A confusion matrix is a table used to display the number of correct and incorrect predictions generated by an algorithm. Confusion matrices can be used to visualise an algorithm's accuracy and may include performance metrics. Using the FP and FN rates calculated, a confusion

matrix was developed to present and allow comparisons of the model's accuracy. The metrics used included those discussed in Section 2.2.2, as well as Specificity and Negative Predictive Value (Sirsat, 2019).

	Actual Positive	Actual Negative	
Predicted Positive	True Positive *(TP)*	False Positive *(FP)*	Precision $\frac{TP}{(TP+FP)}$
Predicted Negative	False Negative *(FN)*	True Negative *(TN)*	Negative Predictive Value $\frac{TN}{(TN+FN)}$
	Recall $\frac{TP}{(TP+FN)}$	Specificity $\frac{TN}{(TN+FP)}$	Accuracy $\frac{TP+TN}{(TP+TN+FP+FN)}$	F_1 Score $2\frac{precision \times recall}{precision + recall}$

Figure 7 - Confusion Matrix

3.2. Web Browser Extension

The browser extension was designed to allow the user to classify an email in their webmail client as either legitimate or a phishing attempt, using the ML model.

As shown in Figure 1, Google Chrome has the highest market share of all desktop web browsers (StatCounter, 2020), and so the browser extension was developed for Chrome to be accessible for the highest number of users. The extension was developed following official documentation from Google Developers (Google LLC, 2014a), with consideration for how it could be successfully integrated with the ML model.

This section covers the design, implementation and testing of the web browser extension.

3.2.1. Design

The browser extension uses a complementary colour palette of turquoise and gold, ensuring that text and buttons are highly contrasted and easy to visually distinguish. 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. In universal colour codes, such as those followed by traffic lights, 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.

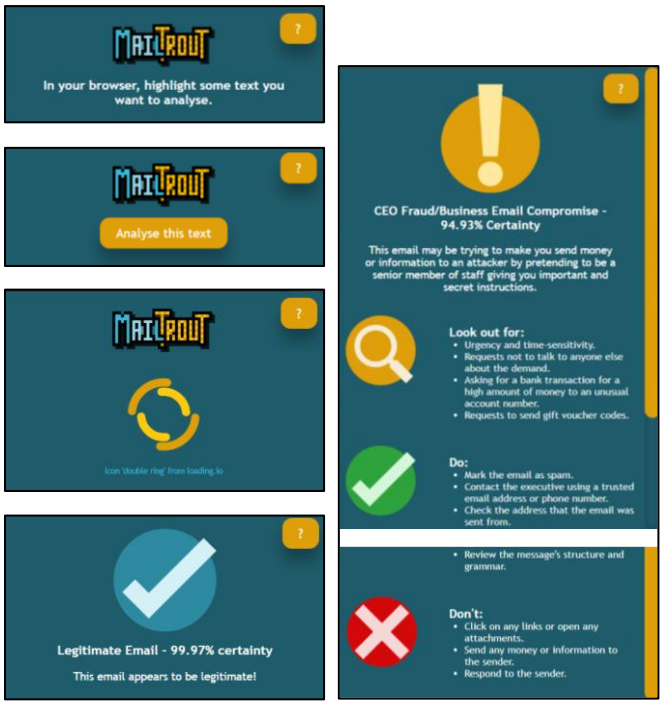


Figure 8 – Example Screenshots of Web Browser Extension

Once a prototype of the extension was developed using this colour scheme, screenshots of the prototype were tested using 'Coblis', an online tool that allows the user to view an uploaded image as it would be seen by a person with a colour vision deficiency (CVD), or 'colour-blindness' (Wickline, 2001). This tool was used to ensure that the browser extension could be read easily by a person with various types of CVD.

The layout of the extension was designed with simplicity in mind, ensuring that text, images and buttons were large, clearly visible and sufficiently spaced, so that each item of the extension could be seen and easily clicked. The buttons in the extension change colour to a lighter shade when hovered over in order to draw the user's attention to them. They also cause the user's cursor to change to a pointer, demonstrating that the user can click on the button in order to perform an action.

Due to the amount of time that could be taken to process and make a prediction from the information selected by the user, a loading icon was used to demonstrate to the user that the browser extension was working to produce its results, so that it was clear that it was working and had not simply stopped. This icon was displayed once the user's input had been registered and hidden once the ML model's prediction and other useful information were displayed.

3.2.2. 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 different web email clients as it did not require email contents to be automatically detected, ensuring that only the text a user wanted to evaluate was processed by their extension. Other options considered for selecting and processing text included highlighting text then using the context menu – shown when the user right-clicks – to send the selected text to the ML model. However, the script used by the machine learning model – TensorFlow.js – could not be accessed from the context menu, as external scripts could not be loaded from the extension's background script.

The extension popup displays on the right side of the page; this is common practice for web browser extensions and so would be expected by the user. This also prevents disruption to the user's browser; 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 on the webpage.

The layout and design of the web browser extension popup was developed using HTML and CSS, while JavaScript was used to make changes to the popup content and to use the ML model to generate a prediction.

3.2.3. Unit Testing

Once a prototype of the browser extension had been developed, testing was conducted to ensure that text could be gathered and processed. Unit testing was carried out to ensure that the extension met its intended purpose and also to ensure that the information displayed, such as instructions on how the extension was to be used and how users should respond to receiving a phishing email, could be easily understood.

3.2.3.1. Functionality

In order to test the browser extension's ability to receive, process and display a result from selected text, a prototype extension was developed to display selected text in the popup. This allowed the functionality of the browser extension to be evaluated prior to integration with the ML model. This method worked successfully and demonstrated that the method used to receive the text of an email for processing was working as expected.

3.2.3.2. Readability

To ensure that the instructions and information given by the extension could be understood by its users, 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. For this series of testing, 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 (Begeny & Greene, 2014).

This formula was used to ensure that text was readable for people around 13 years of age, as this is the minimum age required for user access by most webmail providers, such as Microsoft Outlook (Microsoft Corporation, 2020) and Google’s Gmail (Google LLC, 2013). This is in response to the Children's Online Privacy Protection Act (1998), a US law that regulates how websites can target or collect information on children under the age of 13 (Children's Online Privacy Protection Act, 1998). Many websites choose to avoid the requirements of this legislation by ensuring that all their users are above 13 years old (Boyd, et al., 2011).

3.3. PROJECT INTEGRATION

The ML model was converted from an HDF file 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 (Google LLC, 2018).

This section covers the design of the browser extension to use the ML model to process data, the conversion of the ML model for successful integration with the extension, and the tests conducted on the system to ensure that the units were integrated successfully.

3.3.1. Design

The browser extension was designed to use the ML model to make classifications on the client side, as opposed to on a server. This approach was suitable as it ensured that the analysis of emails would be a faster process than it would if the extension had to load and use the model by sending requests to a server; as a server may be running multiple requests for many users,

it will share its computational power between all these processes, meaning they may take a long time to complete. However, requests made on the client side may take considerably less time, as the computational resources are being used by only one user

The browser extension was designed to recognise text that had been selected by a user in their web browser, process this text and use the ML model to generate a prediction, and then produce an output on the extension popup based on this prediction.

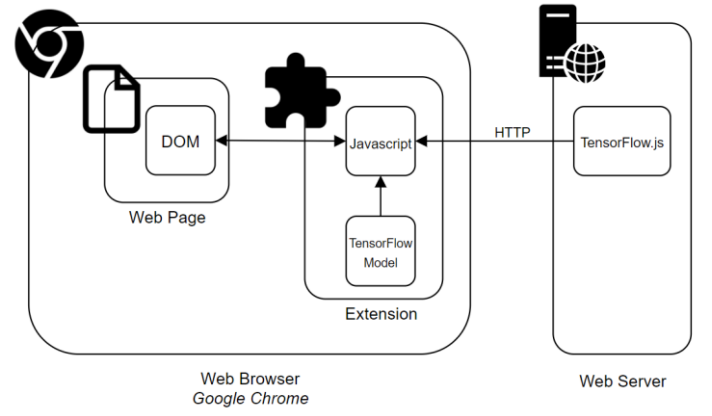


Figure 9 - Architecture of Chrome Extension

3.3.2. Implementation

As discussed in Section 3.1.2.2, ML models can only process numerical data, meaning that text must be converted to integer sequences before being used for training or making predictions. During the training of the ML model, a Keras tokenizer from the TensorFlow Python library was used to convert text into sequences, however TensorFlow.js for JavaScript does not have any inbuilt tokenizers or text-to-sequence functions. Therefore, text had to be manually converted to tokenized integer sequences by the browser extension before it could be processed by the ML model.

The first stage of pre-processing was to convert the selected text to lowercase characters, and then remove all punctuation characters and stop words. Then, the text was converted into an array of words, with each item being split by the spaces between each word.

As discussed in Section 3.1.2.2, the tokenizer used during training was exported from the Python script as a JSON file. This file contained a word index, mapping each word to its numerical token, which was manually copied from the file for use by the browser extension. The extension created a tokenized sequence by searching for each word in the text in the word index; if the word was found, its index was added to a new list, and if not the value zero was added.

Once the tokenized sequence was created, it was pre-padded with zeros using the ‘unshift’ function or post-truncated using the ‘pop’ function until it was the standard length of 500 items. The sequence was then converted to a TensorFlow.js stack, so it could be processed by the ML model. The model was loaded from the JSON file stored in the extension directory and then used to generate a prediction based on the sequence.

The prediction consisted of an array of probabilities that the sequence was one of the potential categories of email. The browser extension then displayed a result based on the classification with the highest probability.

3.3.3. Integration Testing

After the integration of the ML model with the browser extension, the system was tested using a selection of example emails displayed in webpage on Google Chrome. The extension was used to select the emails, process the data, and output a prediction result.

Initially, the browser extension occasionally failed to recognise when text was highlighted on a webpage, resulting in the button for analysing the text not appearing, and requiring the user to refresh the extension to proceed. This issue was caused by the relevant function not being included in the DOMContentLoaded listener, which executed code once the extension’s HTML popup was displayed (Mozilla Foundation, 2019). This listener was in use to listen for a button press to generate a prediction, but not to initially recognise if any text was selected. Once the script to analyse the user’s text selection was added to this listener, the issue was resolved.

The manual tokenization process created scope for human error in the processing of the email data. For example, initially the ML model was unable to process the sequence in order to generate a prediction. This was due to an error in which zeros were added for padding to the email using speech marks, resulting in them being recognised as string characters and therefore not processable by the model. Once speech marks were removed, the model worked as expected.

3.4. User Acceptance Testing

Following integration testing, user acceptance testing was carried out to evaluate the extension’s usability by understanding if potential users could use the extension without guidance and if they would like to use the extension again, among other factors of usability. Participants were able to take part in testing either by attending an in-person session or by carrying out the experiment remotely from their own computer.

The survey used to record participants' feedback on the extension's usability was developed with Google Forms (Google LLC, 2014b), and the webpage used to display sample emails for participants to test the extension on was developed with HTML, PHP and JavaScript.

The materials required for the study included the browser extension being tested, computers running Google Chrome suitable for supporting the extension, the Google Forms survey, and access to the webpage for testing.

This section covers the design and implementation of the survey and experiment used during user acceptance testing.

3.4.1. Design

The testing survey was designed to gather demographic information about participants, namely their age, gender, highest attained level of education, nationality, and field of study or work, as well as information about their prior knowledge of phishing, such as how they would rate their ability to spot a phishing email. This information was collected to allow trends to be identified in the test results, such as if variances in any of these characteristics tended to affect how usable a participant found the extension or how likely they would be to use it again.

Participants were initially asked about their familiarity with the terms used to describe the four categories of phishing email. 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. This information was collected to determine the effect that using the extension had on a participant's ability to spot phishing emails and determine if they found some categories of phishing attacks harder to identify than others.

To ensure that results were accurate and that the participants' actions were not influenced by prior knowledge about the purpose of the study, participants were given a scenario to reduce their bias. This scenario was that the participant was working for an organisation and had been asked to review their boss' email inbox using the extension to identify phishing emails that had been received.

A set of 10 example emails which the participants were asked to use the extension to evaluate were then displayed in the web browser. Once they completed this evaluation, they were asked to complete the survey with a review of how usable they found the extension, using the System Usability Scale (SUS) – a series of 10 questions used to calculate a system's usability score as a number out of 100, with a higher number representing a greater level of usability (Brooke, 1996). They were also asked how likely they would be to identify phishing emails of each

category, 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 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.

3.4.2. Implementation

The emails used to test the extension were displayed on a PHP webpage which allowed the user to mark each email as either safe or phishing. This webpage was designed to randomly select 10 emails out of a possible 30 and display these to the participant one at a time, moving to the next email once the participant marked each as either safe or phishing. This webpage was designed to look as similar to common webmail clients as possible to increase the believability of the given scenario.

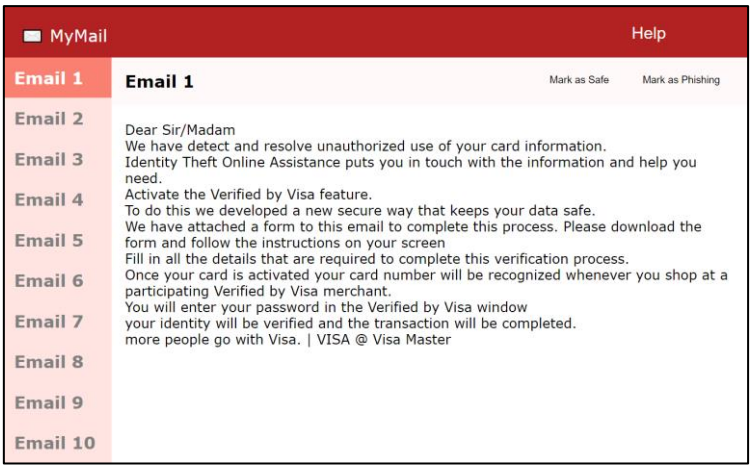


Figure 10 - A screenshot of the PHP webpage used during user acceptance testing

Other approaches considered for displaying emails to the participants included using a fake SMTP inbox such as Mailtrap (Railsware Solutions FZ-LLC, 2012) to display emails as they would appear in a typical webmail client, and using the browser extension to hide elements of the page not needed for the test, such as links to account information. However, doing so caused unpredictable results, such as links on the webpage not functioning or causing the webpage to appear empty.

The development of a simple webpage was chosen over this approach, as it was found to be easier to develop a site accommodated to the needs of the research than to attempt to edit an existing page. Also, access to a tool such as Mailtrap would require authentication if

participants were to test remotely, however the PHP webpage could be accessed by simply following a link.

The bespoke PHP webpage also allowed for the results of how emails were marked by participants to be recorded in a database using SQL queries. These results were then used to assess how accurate the classifications made by the participants were and evaluate if any categories of phishing email were generally harder to identify than others.

As discussed, the System Usability Scale (SUS) was used to evaluate how usable participants found the extension. The original 10 SUS statements alternate between positive and negative wording to minimise respondents' bias to give the answer that they feel the researcher desires. However, Sauro and Lewis (2011) compared the original statements to an all positively worded version and found that there was no evidence that the original statements impacted respondents' bias. Furthermore, they found that by using an all positive version of the SUS, researchers reduce the risk of respondents making mistakes by forgetting to reverse their score, as well as researchers reporting incorrect data as a result of not reversing scales when calculating usability (Sauro & Lewis, 2011). For these reasons, an all positive version of the SUS was used during user acceptance testing. A comparison of the original SUS questionnaire and an all positively worded variant can be viewed in Appendix A.

The SUS provides a usability score out of 100, however this score does not clearly translate to a clear judgement of the system's usability. Bangor et al. (2009) proposed adding an adjective rating scale to the SUS to assist interpretation of SUS scores. By adding a seven-point adjective scale to the standard SUS questionnaire, the researchers found a correlation between adjective ratings and SUS scores (Bangor, et al., 2009). Using the results of this research, Amrehn et al. (2019) mapped the probabilities of each adjective rating against SUS scores, allowing for the absolute interpretation of a single SUS score.

Adjective	SUS Score
‘Best Imaginable’	> 93.1
‘Excellent’	> 77.9, ≤ 93.1
‘Good’	> 60.7, ≤ 77.9
‘OK’	> 44.0, ≤ 60.7
‘Poor’	> 28.6, ≤ 44.0
‘Awful’	> 14.0, ≤ 28.6
‘Worst Imaginable’	≤ 14.0

Table 4 – Adjective Ratings of SUS Scores (Amrehn, et al., 2019)

The feedback gathered from the survey was used to identify minor issues that could be resolved on discovery, such as issues with text displaying incorrectly, as well as areas that could be reviewed and improved in future work on the project.

3.5. SUMMARY

Overall, the project was implemented and tested in its component parts, which were then integrated and tested further. The ML model was implemented using a BLSTM network found to be the most suitable for sentiment analysis, and was trained using an email corpus collated from existing datasets, online forums and image search results. The model was tested to ensure its accuracy and its ability to be exported for use in the browser extension.

The browser extension was built for use with Google Chrome and was designed to be a usable and accessible security tool. Once implemented, it was evaluated for its functionality when interacting with a webpage in the browser, and for the readability of the information it provided.

Once both components were implemented, they were integrated into the final tool. It was ensured that the ML model was hosted locally on the user’s web browser, as this ensured that the system would run faster. The extension was developed to use the ML model to classify emails. The integrated solution then underwent acceptance testing, in which participants were asked to use the extension to classify a series of emails, then evaluate the extension’s usability.

The testing results for the extension’s readability, the model’s accuracy, and the usability of the integrated solution are documented in Section 4.

4. RESULTS

This section covers the readability results of the browser extension text, the accuracy of the machine learning model and the usability results of the user acceptance test. This section is followed by an evaluation of these results in Section 5. These results show that overall, the ML model classified emails accurately and test participants were happy with the usability of the extension, finding it simple to use and educational on the techniques commonly used in phishing emails.

4.1. TEXT READABILITY

As discussed in Section 3.2.3.2, the aim of the extension text was to be readable for people around 13 years of age according to the Dale-Chall formula. However, it was not always possible to ensure that the text met this aim while ensuring that the desired message was conveyed. For this reason, text deemed readable for those up to 18 years old was considered acceptable, although not ideal. The age of 18 was decided on as this was the minimum age required for participants to take part in user acceptance testing, as well as the age of majority – the legally recognised threshold of adulthood – in most countries (UN General Assembly, 1989).

The readability test found that the descriptions of impersonation, extortion and BEC phishing emails were too complex, so new versions of these descriptions were made using simpler language. A comparison of the initial and simplified versions of the description for BEC phishing emails along with their readability ratings can be viewed in Table 5.

	BEC Summary V1	**BEC Summary V2**
Text	*CEO Fraud/Business Email Compromise - This email may be attempting to trick you into sending money or information to an attacker by impersonating an organisation executive giving urgent and confidential instructions.*	*CEO Fraud/Business Email Compromise - This email may be trying to make you send money or information to an attacker by pretending to be a senior member of staff giving you important and secret instructions.*
Dale-Chall Readability Score	10.17	7.73
US Grade Level	16+	9 - 10
Equivalent Age Level	21+	14 - 16

Table 5 – Comparison of initial and simplified versions of BEC phishing description

4.2. MODEL ACCURACY

As discussed in Section 3.1.2.2, the model was trained with a sequence size of 500 words, with pre-padding and post-truncation used to reach this standard size. The size of 500 words was chosen as the majority of emails in the dataset were found to fit in this range. However, the model was initially trained using sequences with a length of 200 items and with pre-truncation.

The initial model version was trained using a dataset of 11341 records. However, the dataset was later reviewed and records that were found to be blank or irrelevant were cut from the dataset. For this reason, the second version of the model was trained with a smaller dataset of 11227 records – 114 less than before.

	Version 1	**Version 2**
Sequence Size (words)	200	500
Padding Method	Pre	Pre
Truncation Method	Pre	Post
Dataset Size (records)	11341	11227

Table 6 - Comparison of Model Training Data

The results of these two methods were compared and it was found that version 2 of the model – which used a larger sequence length and post-truncation - produced more accurate results. Notably, version 2 produced no false positives.

Metric	**Version 1**	**Version 2**
TPs	5882	5930
TNs	5342	5287
FPs	7	0
FNs	110	10

Table 7 – Model Results

To understand the model's accuracy when classifying each category of phishing email, the true categories of emails were recorded when counting the false negatives (FNs).

	Version 1		**Version 2**	
Category	**Total**	**No. of FNs**	**Total**	**No. of FNs**
Unexpected Money/Winnings (UNX)	3601	14	3581	0
Extortion (EXT)	1446	42	1427	0
Impersonation (IMP)	552	36	541	7
Business Email Compromise (BEC)	393	18	391	3

Table 8 - Model FNs by Email Category

These results demonstrate that overall, version 2 of the model produced more accurate results than version 1. The accuracy of these models is compared further in Section 5.2.

4.3. User Acceptance

As discussed in Section 3.4, user acceptance testing was carried out to evaluate the extension's usability. 44 individuals with varying demographic characteristics tested the extension and provided feedback on their experience.

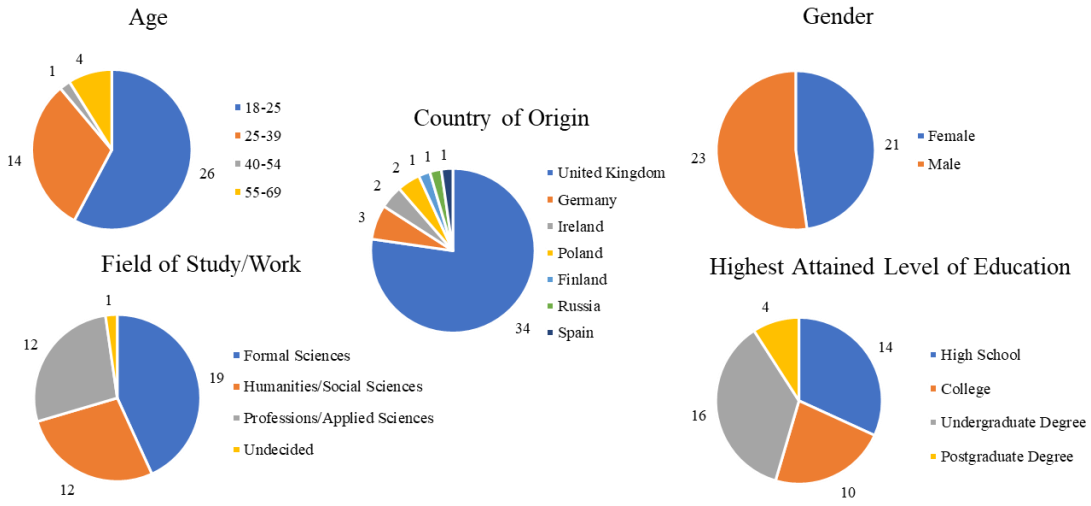


Figure 11 – Demographic Characteristics of Participants

As discussed, participants' responses to the System Usability Scale (SUS) questionnaire are used to calculate a usability score as a number out of 100, with a higher number representing a greater level of usability. Participants' answers to the SUS questionnaire gave the extension a SUS score of 87.5 out of 100.

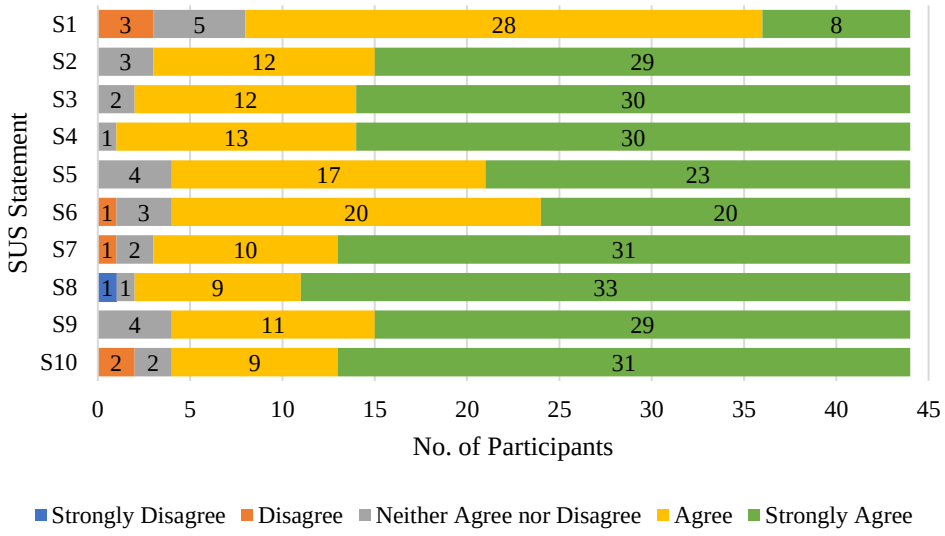


Figure 12 – Participant Agreement Scores by SUS Statement

Notably, younger participants were shown to give the extension a higher usability rating than older participants. As shown in Figure 11, only one participant reported to be in the age range

of 40-54. To aid in the presentation and interpretation of these results, the two highest age ranges (40-54 and 55-69) were combined into one range of 40-69.

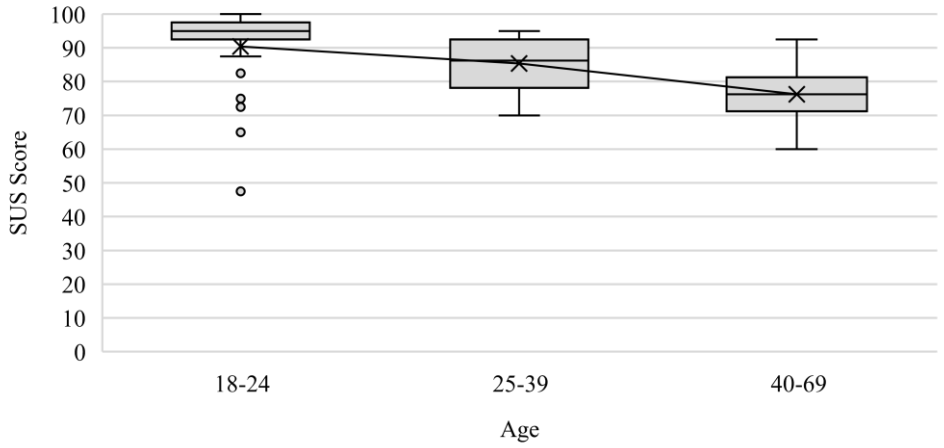


Figure 13 – SUS Score by Age

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, as discussed in further detail in Section 5.3.

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, and a common view among participants was that the extension was well designed and laid-out, making text easy to read and understand.

Overall, participants found the instructions provided for using the extension to be very helpful, scoring them an average of 4.66 out of 5.

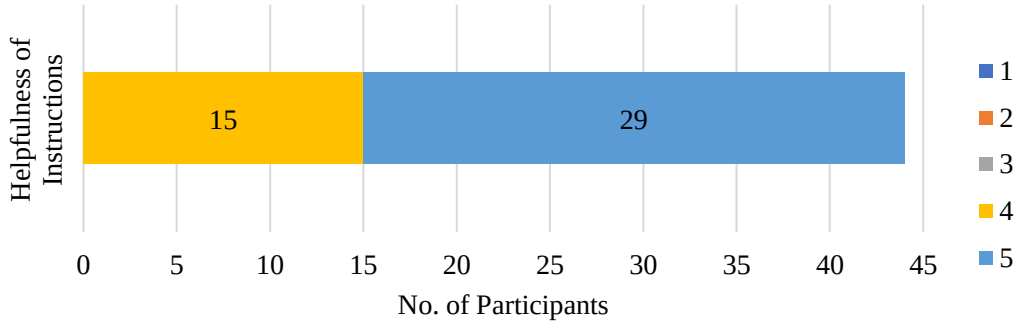


Figure 14 - Participant ratings of helpfulness of the instructions provided for using the extension, where 1 is 'not very helpful' and 5 is 'very helpful'

Several participants also expressed views that the extension would be useful to those who are less confident online and more vulnerable to phishing emails, such as the elderly. A common

theme among responses was that participants said that they would recommend the extension to people they knew who commonly receive and view phishing emails. Overall, participants said that 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.

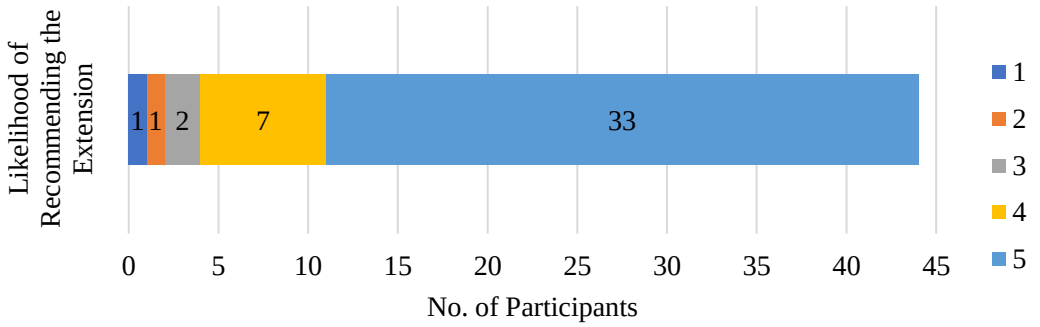


Figure 15 - Participant ratings of their likelihood to recommend the extension, where 1 is 'not very likely' and 5 is 'very likely'

Prior to testing, 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 of this again.

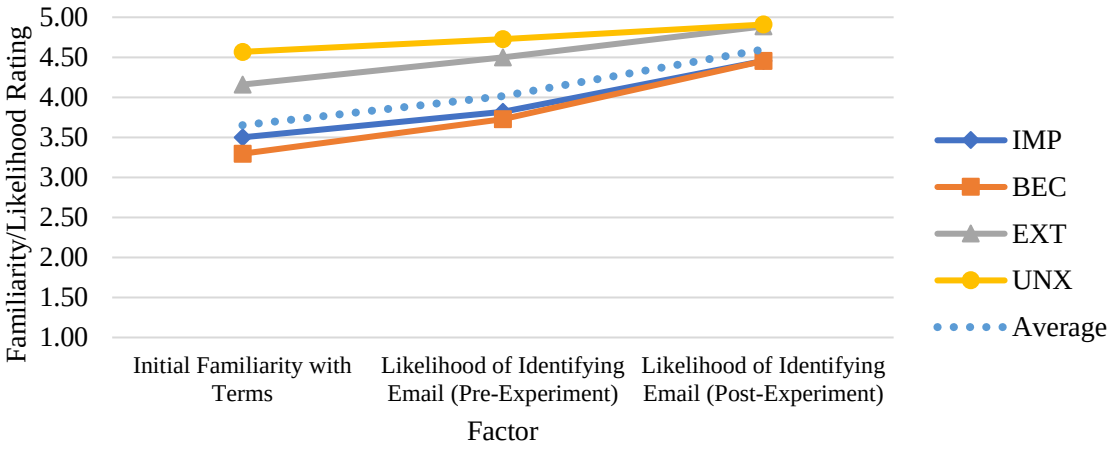


Figure 16 – Familiarity with Categories of Phishing Emails, where 1 is 'not very familiar/likely' and 5 is 'very familiar/likely'

Some participants raised issues with the extension's functionality, particularly relating to the highlighting of text. Participants reported that the result produced by the extension was more accurate if more text was selected for analysis, and therefore users could easily receive less accurate results if they miss out some words when highlighting an email's text. Participants also expressed issues with the level of interaction and clicking required to use the extension –

notably, the need to highlight text, click the extension icon, then click the button in the extension popup to analyse an email.

Some participants argued that the extension often flagged emails as phishing where there were no common characteristics of phishing attacks present in the text, such as requests for information or money, or when the email appeared to have been sent from an individual trusted by the recipient.

As discussed in Section 3.4.2, the results of the user acceptance testing experiment were recorded to evaluate the accuracy of participants' classification of emails either as safe or phishing. These results are displayed as a confusion matrix – a table of the number of correct and incorrect predictions – in Figure 17.

	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 17 - Confusion Matrix of Participants' Email Classifications

The specific categories of phishing emails were also recorded and used to determine the number of phishing emails erroneously marked safe (false negatives) by each category, as shown in Table 9. These results showed that BEC emails had the largest number of FNs, suggesting they may have been the category detected with the least accuracy.

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

Table 9 – Participant FNs by Email Category

The results presented in this section are discussed and evaluated in Section 5.

5. DISCUSSION

This section discusses the readability of the text displayed in the browser extension, the accuracy and success of the ML model used by the extension and the usability and functionality of the integrated solution as a security tool.

5.1. TEXT READABILITY

As discussed in Section 4.1, the Dale-Chall readability test found that the descriptions of impersonation, extortion and BEC phishing emails had a readability difficulty higher than the acceptable range.

It was taken into consideration that the readability scores calculated by the Dale-Chall formula were limited by the use of complex words throughout the extension, such as information regarding machine learning or mentions of cryptocurrency wallet addresses in the advice on what to look out for in potential extortion scam emails. Where possible, text was simplified, which was achieved by replacing long and uncommon words with shorter and more commonplace equivalents, such as “confidential” with “secret”. Similarly, complex words were replaced with simple descriptions, such as “executive” with “senior member of staff”.

As exemplified in Table 5, the simplified versions of the summaries produced lower readability difficulty scores according to the Dale-Chall formula, resulting in lower equivalent age levels.

5.2. MODEL ACCURACY

An initial objective of the project was to develop a ML model that could detect phishing emails with a high level of accuracy. The accuracy of the model was primarily based on its FP and FN rates, for which adjective descriptors were assigned based on existing work, as shown in Table 3.

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

Table 3 – Categories of ML Model FP and FN Rates

Using the formulas shown in Figure 2 and the results shown in Table 7, the FP and FN rates of the models were calculated as the values presented in Table 10.

Model Version	False Positive (FP) Rate	False Negative (FN) Rate
1	0.00131	0.0183
2	0.0	0.00168

Table 10 – Model FP and FN Rates

These results clearly demonstrate that the second version of the model was the most accurate. Based on the categories of FP and FN rates discussed in Table 3, version 1 of the model had a 'good' FP rate and an 'excellent' FN rate, while version 2 had both 'excellent' FP and FN rates. While both sets of these results were very successful, version 2 – which was used in the final product – produced stronger results.

Confusion matrices were created for both model versions to compare their accuracy further, using the metrics shown in Figure 7. As expected, the second version of the model outperformed the initial version on every metric, and notably had a precision of 100%, due to its FP rate of 0. These confusion matrices can be viewed in Appendix B and Appendix C respectively.

In comparison to existing research, this model outperformed other ML-based phishing detection methods. As shown in Table 1, the email classifier PILFER combined with a feature using the spam filter SpamAssassin developed by Fette et al. (2007) had a FP rate of 0.0013 and a FN rate of 0.036, while the trained SpamAssassin filter alone had a FP rate of 0.0012 and a FN rate of 0.130 (Fette, et al., 2007).

The most accurate RNN phishing classifier developed by Halgaš et al. (2019) had FP and FN rates of 0.0126 and 0.0147 respectively (Halgaš, et al., 2019). The second version of the model demonstrated more accurate FP and FN rates than all three of the aforementioned phishing email classifiers, demonstrating its very impressive level of accuracy.

The data shown in Table 8 was used to calculate a FN rate for each email category, as shown in Table 11.

	Version 1	**Version 2**
Category	**FN Rate**	**FN Rate**
Unexpected Money/Winnings (UNX)	0.0039	0.0
Extortion (EXT)	0.029	0.0
Impersonation (IMP)	0.0652	0.0129
Business Email Compromise (BEC)	0.0458	0.0077

Table 11 - Model FN Rates by Email Category

Version 1 produced ‘excellent’ FN rates for UNX and EXT emails, but ‘good’ FN rates for IMP and BEC emails. Version 2, however, produced ‘excellent’ FN rates for all four phishing email categories. While the lowest FN rates were still in an acceptable range, they were considerably worse than the other rates.

These results show that, while the second model had an overall lower FN rate, both models had their highest FN rate when attempting to classify impersonation phishing (IMP) emails. A possible explanation for these results may be that due to the language used in impersonation phishing being similar by design to the language of emails from legitimate organisations, they may be more difficult for a ML model to identify.

Both models’ second highest FN rate was for BEC emails. Much like emails that impersonate organisations, emails that impersonate individuals may be more difficult for a ML model to identify due to their similarity to legitimate emails. Another possible explanation may be the generally short length of BEC emails, especially those which simply set out to understand if a victim is available for further emails. The smaller length of these emails may result in a lack of data for the ML model to process, rendering it less likely to produce an accurate result with a high certainty.

In line with expectations, the second version of the model displayed superior accuracy to the initial version. This may be explained by the differences between the methods used when training the models, such as the size and quality of the dataset used; as discussed, version 2 was trained using a dataset from which some irrelevant data had been removed, reducing its size but improving its quality. These findings on the importance of dataset quality are consistent with those of Halgaš et al. (2019) as mentioned in Section 2.2.1.

Another factor in version 2’s superior accuracy may have been its greater sequence size. By allowing over double the number of words to be processed by the ML model in order to classify an email, the results generated were more accurate. As discussed in Section 3.1.1.1, the BLSTM network used by the model to generate predictions uses the output values of each item in an input sequence to provide contextual information at each stage of processing. Therefore, this suggests that a larger input sequence would allow the network to gather more contextual information from the text, enabling it to produce more accurate results.

These results may also be attributed to the use of post-truncation in version 2, as this prevented the loss of potentially important data present at the beginning of an input sequence. However, the scope to test the impact of post-truncation in this study was limited, given the

large sequence size in use; few emails required truncation as they did not exceed 500 words in length. No information was found in the literature on the question of the impact of truncation methods on the accuracy of a ML model, however it is possible that these impacts may vary depending on the type of data being processed and the purpose of the ML model.

However, these findings are somewhat limited by issues with the dataset. Firstly, due to the lack of data available, the same dataset was used for training and testing. This is generally not considered best practice, as the model's familiarity with the training data 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 are unable to make more general predictive rules that can be applied to new data, in an issue known as 'overfitting' (Dietterich, 1995). Due to the use of the same data 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.

Secondly, the quality of the dataset may have been negatively impacted by the methods used to collect data or by issues with the existing corpora used. The Fraud Email Dataset (Verma, 2018) used as part of the training dataset – discussed in Section 3.1.2.1.1 – contained some email metadata such as the date and time that emails were sent, and encoded text for use with older email servers. This was not useful for training and may have caused the ML model to become overfitted or to not identify words and phrases correctly. This 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 (PRAW) as part of a Python script to extract comments from a Reddit thread of extortion emails (EugeneBYMCMB, 2019) as discussed in Section 3.1.2.1.2, 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, or due to the presence of text added to images to highlight common signs of phishing attacks.

5.3. USER ACCEPTANCE

Using the adjective ratings shown in Table 4, the SUS score of 87.5 given to the extension can be described as 'excellent'. This is a very positive finding and proves that the extension met its aim of being a usable security tool. As discussed in Section 3.4.2, Bangor et al. (2009) proposed an adjective rating for SUS scores by adding an adjective scale to the SUS questionnaire. Of the 959 participant responses to SUS questionnaires with added adjective scales collected by Bangor et al., 289 gave systems the adjective rating of 'excellent'. 654 participants gave systems a rating of 'good' or below, meaning that responses with a rating of 'excellent' rated systems as more usable than 68.2% of other responses (Bangor, et al., 2009). This demonstrates that, with an 'excellent' SUS score, the extension exhibits very high usability in comparison to existing tools.

As discussed, many participants reported that they found the extension easy to use and understand. One participant suggested that, due to the extension's accessibility and embedded nature, users would be more likely to keep using it.

- *"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 on how impressed they were with the speed and ability 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"*
- *"I was amazed by how quickly [the extension] could analyse whether [an email] was a phishing email or not"*

Commenting on the design and layout of the extension, one participant with strong colour vision deficiency (CVD) reported that they found the colour scheme well contrasted and therefore had no issues, while other participants with specific learning difficulties (SpLDs) found the extension accessibly designed with simple layouts, colour-coding, and succinct and useful 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 how to identify 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 extension, as well as being useful for finding out the likelihood of something being a phishing email, also taught me about forms of phishing emails that I had not previously known to look out for"*

The responses to each statement in the SUS survey were very positive overall, generally averaging between 'Agree' (4) and 'Strongly Agree' (5). Surprisingly 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".

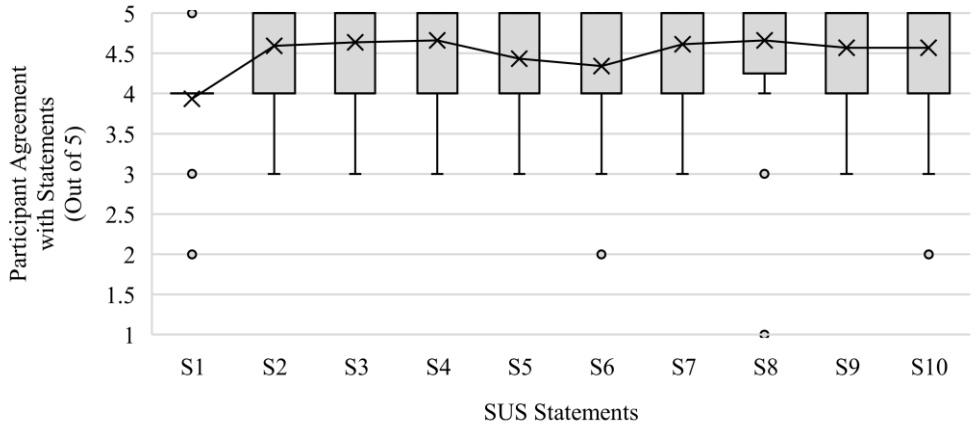


Figure 18 – Distribution of Participant Agreement Scores by SUS Statement

A possible explanation for this might be that, while participants had positive feedback to give on the extension, they did not feel that they themselves had a need for it, 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 that they were able to identify phishing emails themselves with high levels of accuracy, they may have felt that the relative advantage of using the extension was low and therefore felt less likely to adopt the practice of using the extension.

To understand this finding further, 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.

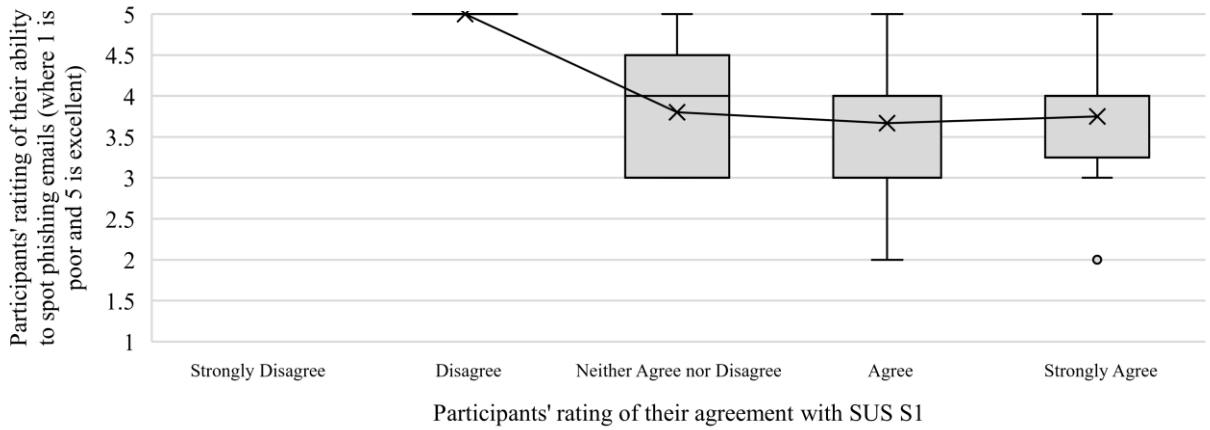


Figure 19 – Agreement with SUS Statement 1 by Ability to Identify Phishing Emails

As hypothesised, participants who answered that they would not like to use the extension frequently reported that they had a strong ability to spot phishing emails, suggesting that they would find using the extension unnecessary.

Another of the characteristics of an innovation is its 'observability', meaning the degree to which the effects of the innovation are visible. If a user perceives the visible results of an 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 the developed browser extension 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.

Further characteristics of an innovation include its 'compatibility' with existing tools and its 'trialability', meaning the degree to which it can be experimented with by the user (Rogers, 2003).

Users may have found the extension to be incompatible with their current technology; for example, they may not have been Google Chrome users and therefore would not have been able to use the extension with their web browser of choice. Users may also have found the extension to have low trialability as they were not able to use it in a genuine setting, such as on

their own emails, resulting in them feeling less confident that adopting the tool would help them.

Overall, participants were shown to have an increased understanding of types of phishing email after using the extension. As shown in Figure 16, average familiarity ratings for each phishing email category increased as participants used the extension to learn about what to look for in phishing emails. These findings demonstrate the extension's ability to not only detect phishing emails in the short term, but also to educate users about how to identify phishing emails in the long term.

As discussed in Section 2.1, existing phishing education tools such as PhishGuru may make use of 'embedded training', a method in which instructional materials are integrated into a user's everyday tasks (Kumaraguru, et al., 2010). PhishGuru utilises users following links in simulated phishing emails as opportunities to educate users on how to protect themselves against phishing attacks. However, the proposed browser extension educates users on phishing emails whenever it is used to analyse an email, and therefore may be more useful for an individual wishing to protect themselves from phishing emails with immediate results, rather than with education over an extended period as is offered by PhishGuru.

Participants who rated their ability to identify phishing emails as a 2, where 1 was poor and 5 was excellent, reported the largest increase in familiarity with phishing email categories as a result of using the extension. To aid interpretation of results, a rating of 2 was considered 'below average', 3 was considered 'average' and 4 considered 'good'. No participants rated themselves as a 1 on the scale.

The higher participants initially rated their ability to identify phishing emails, the greater their familiarity with the categories of phishing email discussed in the experiment. However, participants of all abilities to identify phishing emails demonstrated an increase in familiarity with categories of phishing emails from using the extension to evaluate emails as shown in Figure 20.

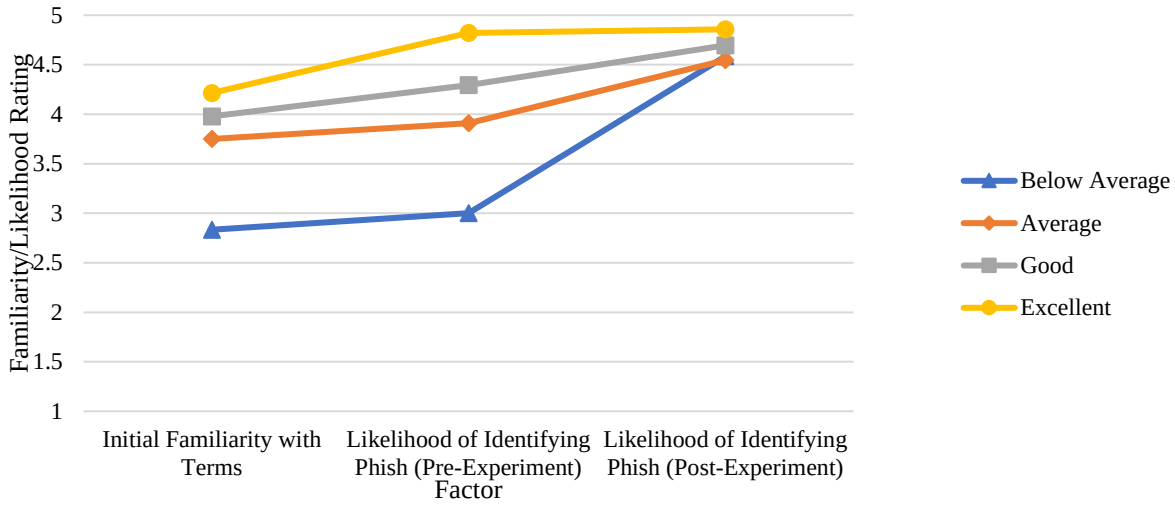


Figure 20 – Familiarity with Phishing Email Categories by Ability to Identify Phishing emails, where 1 is 'not very familiar/likely' and 5 is 'very familiar/likely'

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.

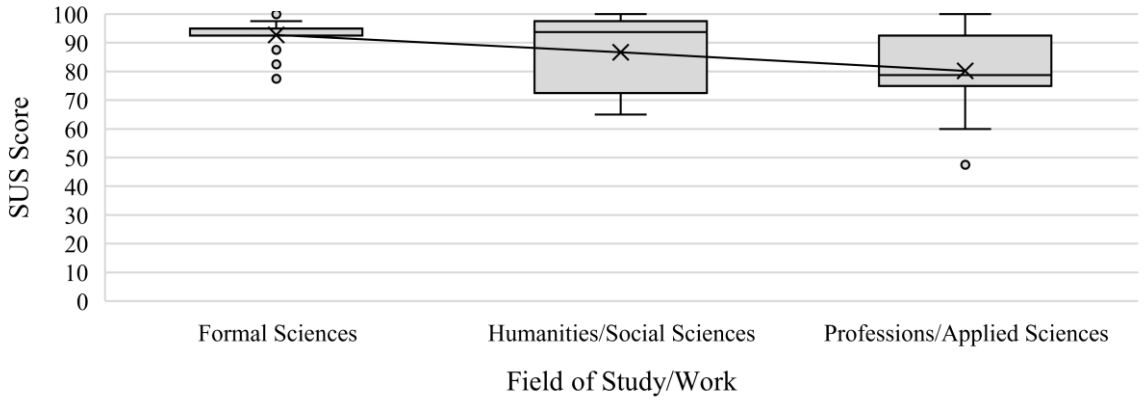


Figure 21 – SUS Score by Field of Study/Work

A potential explanation for this result may be that those in formal science fields are more frequent users of computer software and therefore more comfortable using computers and better at learning how to use new tools. Participants in formal sciences may have had more experience specifically with web browser extensions and therefore 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, specifically 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.

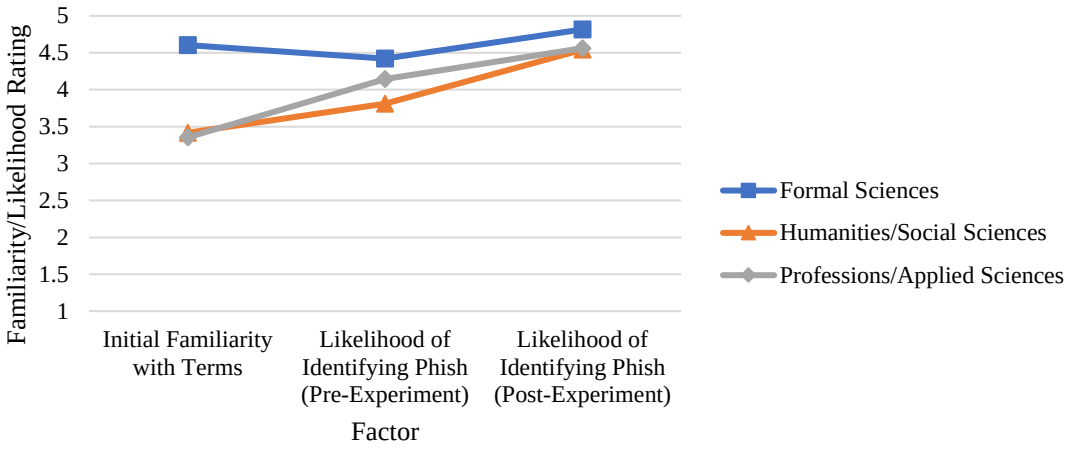


Figure 22 – Familiarity with Phishing Email Categories by Field of Study/Work

However, it is important to note that the average SUS scores of each subject field were all in the 'excellent' category, so the differences between subject fields is only a minor concern.

Secondly, as shown in Figure 13, younger participants were shown to find 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. However, users born before the 1980s – commonly referred to as 'digital immigrants' – grew up before the widespread use of digital technology and so 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 of using browser extensions and of dealing with phishing emails; younger participants demonstrated an overall higher familiarity with categories of phishing emails throughout the experiment than older participants.

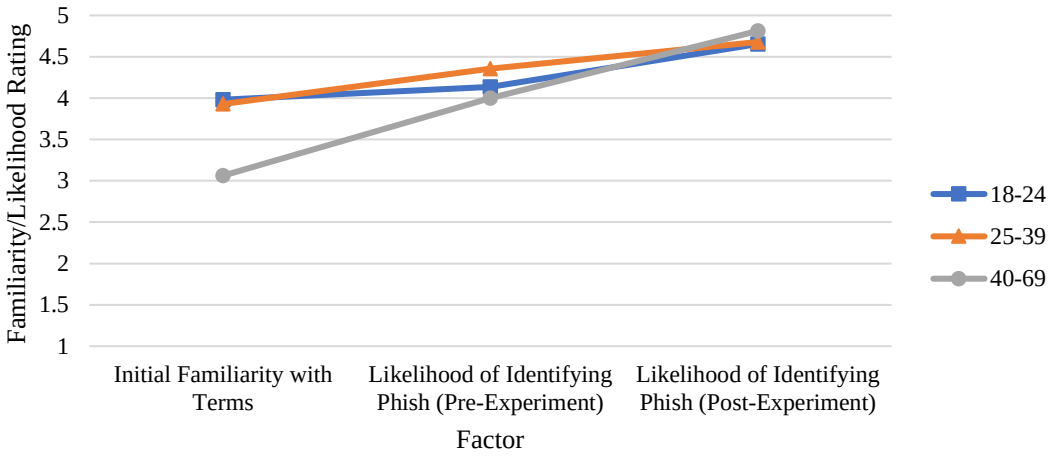


Figure 23 – Familiarity with Phishing Email Categories by Age

As discussed, some participants raised issues with the extension's functionality relating to the highlighting of text, reporting that users could receive inaccurate results if they do not select all of an email's contents.

- *"If upon first highlighting I missed a couple of words of text and checked it, then fully highlighted the text and checked it, I received two very different answers on what kind of phishing email it was likely to be"*
- *"If I mistakenly highlight the first line "Good morning" then it says 99.98% sure this is a legit message."*

One participant expressed an issue that they often forgot to click the button in the popup, resulting in the extension not running when they thought it was.

- *"You forget to click analyse text and so are sitting there wondering what's taking it so long."*

Another participant commented that users' actions after receiving a suspicious email could not be based on the extension alone, and ultimately relied on the user's own intuition and expertise to make the final judgement.

- *"[The extension] would be best suited for people who would maybe struggle to spot a more elaborate and thought out [phishing] email but who still had the common sense to know if something's legitimate"*

Overall, a majority of the issues raised by participants regarded the lack of automation present in the tool. Due to the reliance on user involvement to process emails, the functionality of the tool was impacted by human error. Considerations of how to address these issues in further work on this project are discussed in Section 6.1.

The results of the user acceptance testing experiment shown in Figure 17 were used to calculate the following FN and FP rates to evaluate 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
	FN Rate 0.0701	**FP Rate** 0.168

Figure 24 – FN and FP Rates of Participants' Email Classifications

Using the ratings from Table 3, the FN rate was found to be 'good', which was acceptable. However, the FP rate was found to be extremely poor. This may have been due to participants being overly cautious when dealing with the emails shown.

During testing, four participants also reported an error in which the webpage stopped responding. If, during an occurrence of this error, participants tried pressing the 'Mark as Phishing' button repeatedly then this may have given an inaccurate depiction of the number of FPs recorded.

Using the findings shown in Table 9, the FN rates of each category of phishing email were calculated as presented in Table 12.

Email Category	FN Rate
Unexpected Money/Winnings (UNX)	0.0081
Extortion (EXT)	0.0103
Impersonation (IMP)	0.0350
Business Email Compromise (BEC)	0.1742

Table 12 – Participant FN Rates by Email Category

While these rates were found to be 'excellent' for UNX, EXT and IMP phishing emails, the rate produced for BEC emails was 'poor'. A possible explanation for this may be due to the nature of BEC emails and how they did not appear suspicious in the testing scenario. As one participant commented, the experiment scenario did not give details such as the nature of work of the organisation involved. This may have made it more likely that participants believed that within this scenario these emails were legitimate and therefore should be marked as safe.

5.4. SUMMARY

Overall, the results of testing were very satisfactory.

The readability levels calculated for the text displayed by the extension were all within a range considered acceptable. Due to the complex nature of language used when discussing phishing techniques, the readability scores of some pieces of text equated to age ranges older than desired, however this text was made less complex wherever possible.

The ML model produced an excellent level of accuracy in comparison to other ML-based phishing detection tools. From a comparison of two versions of the model, it was determined that the use of a longer sequence length when processing text enabled greater model accuracy, as this provided the model with more data to process and base its prediction on. However, the findings that can be drawn regarding the ML model's accuracy are limited due to potential overfitting as a result of reusing the same dataset for training and testing, rather than using separate datasets for each stage.

The extension was given an excellent usability rating by user acceptance testing participants. Test participants also reported that they would be very likely to recommend the extension to someone looking to protect themselves from phishing emails, and highlighted that the extension may be especially helpful to those less confident with technology and vulnerable to phishing emails. Overall, older participants gave the extension a lower usability score than younger participants, suggesting that they found the extension less easy to use. This was attributed to younger users having more experience of digital technology from a younger age, thus making it easier for them to learn how to use new technologies.

6. CONCLUSION

The aim of this project was to investigate how a machine learning model could be integrated with a web browser extension to use natural language processing to identify phishing emails based on their content.

This project found that, due to both 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.

This project also found that the web browser extension provided a suitable way to create an embedded learning tool, as users can access the extension while going about their standard business on their computer, such as checking their emails.

Many participants reported that they considered the tool to be potentially useful for those less confident when using the internet and thus more at risk of falling victim to phishing campaigns, such as the elderly, due to its simple nature and helpful instructions. Participants widely agreed they would recommend the extension to someone looking to protect themselves from phishing emails, and many reported that it would be a valuable resource in preventing people from falling victim to phishing attacks.

The project was designed to be an educational tool, teaching users how to detect phishing emails themselves so that eventually they may no longer need the tool in order to keep themselves protected. This aim was met successfully, as users were shown to have learned more about different types of phishing emails as a result of using the extension and considered themselves more likely to be able to identify these emails in the future.

These results suggest that by training a ML model with a dataset of phishing and legitimate emails, an algorithm can be developed that can detect phishing emails with a high level of accuracy. This algorithm can then be integrated into other systems to create a tool that can identify and classify phishing emails.

These findings also indicate that browser extensions can act as accessible security tools, as they require little technical knowledge to use and can easily be included into a person's standard online activities. Due to their simplicity and embedded nature, browser extensions may be particularly useful for those with less experience of using the internet, and as security tools may effectively protect those most vulnerable.

These results highlight the importance of including meaningful descriptions of results in security tools, as these can help educate users on how to remain protected by identifying potential security risks themselves. Users should be provided with descriptions of what they

should and should not do in the event of identifying a security risk so that they can develop an understanding of how to respond appropriately.

The findings of this project contribute to the existing knowledge of ML-based phishing detection by demonstrating the impact of dividing phishing emails into smaller categories to enable more accurate predictions, as well as how sentiment analysis can be utilised to successfully categorise emails based on their text content. This project has demonstrated how factors in the processing of data – such as the use of larger sequence sizes – also help to develop a more accurate ML model.

These findings will also be relevant to those interested in usable security, as this project has demonstrated how a web browser extension containing readable instructions and information effectively protects users from security risks.

6.1. Future Work

A limitation of this project was the relatively small size of the dataset used to train and test the ML model, which may have resulted in the model not being able to accurately classify new data. Further research could be conducted to evaluate the impact on the model's accuracy when trained with a larger dataset. Such data could be compiled by setting up an inactive email address designed to receive spam email – known as a 'honeypot' or 'spamtrap'. Emails could also be collected by asking for email users to forward any phishing emails they have received to this unused address. The messages sent to this address could then be harvested, divided into the categories of phishing, and included in the dataset.

Further development of the web browser extension could be carried out to improve the tool's functionality and usability, as per the responses of testing participants.

The number of keystrokes required from users to evaluate emails could be reduced by evaluating highlighted text once the extension icon is clicked, rather than requiring the user to click another button in the extension pop-up window. Also, the option to evaluate highlighted text using the extension could be added to the context menu, shown when the user right-clicks.

To address issues around the impact of users not selecting all the email text correctly on the tool's prediction, the extension could be developed to automatically detect the body of an email present on a webpage, evaluate the email and warn the user if it detects anything suspicious. This may provide further convenience for users as well as increased security, as it would require less interaction with a potentially dangerous email.

Additional features could be implemented into the extension to educate users about phishing techniques, such as the highlighting of words and phrases that are particularly indicative of a phishing email, and the expansion of shortened URLs, which may be used to obfuscate malicious links. Similarly, the method of processing text could be developed to account for attempted filter evasion through the use of 'homoglyph' attacks, in which characters are replaced with similar-looking characters, such as those from the Greek or Cyrillic alphabets, to bypass standard spam detection methods. The use of OCR technology could also be used to identify text shown as an image in an email, and to include this text when evaluating the email's text content.

This project could be repeated using other approaches, such as developing a version of the extension for other web browsers such as Mozilla Firefox, to make the tool accessible for a wider range of users. Similarly, the tool could be developed as an add-on for a webmail client, such as Microsoft Outlook or Google's Gmail, to expand its abilities to detect and act on suspicious emails by deleting or quarantining them. The tool could also be developed as software designed to run on an email server to detect and filter phishing emails before they reach their intended recipient.

To further evaluate the educational benefits of the tool, a longitudinal study could be conducted to observe the impact of using the tool over a period of time on the ability of end users to identify phishing emails.

In summary, this project demonstrated how a machine learning model can be integrated with a web browser extension to use natural language processing to identify phishing emails based on their content. The resulting integrated tool demonstrated high levels of accuracy when detecting phishing emails, as well as high levels of readability and usability for end users.

LIST OF REFERENCES

Aggarwal, A., Rajadesingan, A. & Kumaraguru, P., (2012). *PhishAri: Automatic realtime phishing detection on twitter.* Las Croabas: eCrime Researchers Summit, IEEE.

Amrehn, M. et al., (2019). A Semi-Automated Usability Evaluation Framework for Interactive Image Segmentation Systems. *International Journal of Biomedical Imaging,* Volume 2019.

Australian Competition & Consumer Commision, (2015). *Types of scams.* [Online]
Available at: https://www.scamwatch.gov.au/types-of-scams
[Accessed 13 March 2019].

Avanan, (2019). *Global Phish Report,* New York: Avanan.

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.* [Online]
Available at: https://github.com/shivam5992/textstat
[Accessed 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.

Boe, B., (2020). *PRAW: The Python Reddit API Wrapper.* [Online]
Available at: https://praw.readthedocs.io
[Accessed 21 January 2020].

Boyd, D., Hargittai, E., Schultz, J. & Palgrey, J., (2011). Why parents help their children lie to Facebook about age: Unintended consequences of the 'Children's Online Privacy Protection Act'. *First Monday,* 16(11).

Brooke, J., (1996). SUS - a quick and dirty usability scale. In: P. W. Jordan, B. Thomas, B. A. Weerdmeester & I. L. McClelland, eds. *Usability Evaluation in Industry.* London: Taylor and Francis, pp. 189-194.

Chen, Y. Y. et al., (2019). Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in Demand-Side Management for Smart Homes. *Sensors (Basel),* 19(9).

Children's Online Privacy Protection Act, (1998). *15 U.S. Code c.91*. [Online]
Available at: https://www.law.cornell.edu/uscode/text/15/chapter-91
[Accessed 9 March 2020].

Chollet, F., (2015). *Keras Documentation.* [Online]
Available at: https://keras.io/
[Accessed 14 March 2020].

Cloudmark Security Blog, (2016). *Survey Reveals Spear Phishing as a Top Security Concern to Enterprises.* [Online]
Available at: https://blog.cloudmark.com/2016/01/13/survey-spear-phishing-a-top-security-concern-to-enterprises/
[Accessed 30 September 2019].

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

Dunlop, M., Groat, S. & Shelly, D., (2010). *GoldPhish: Using Images for Content-Based Phishing Analysis.* Barcelona: Fifth International Conference on Internet Monitoring and Protection, IEEE.

Dwarampudi, M. & Reddy, N. V. S., (2019). *Effects of padding on LSTMs and CNNs.* [Online]
Available at: https://arxiv.org/pdf/1903.07288.pdf
[Accessed 5 February 2020].

EugeneBYMCMB, (2019). *The Blackmail Email Scam (part 3) : Scams.* [Online]
Available at:
https://www.reddit.com/r/Scams/comments/biv65o/the_blackmail_email_scam_part_3/
[Accessed 21 January 2020].

Fette, I., Sadeh, N. & Tomasic, A., (2007). *Learning to detect phishing emails.* Banff: 16th International World Wide Web Conference, 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. 1333-1340.

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 LLC, (2013). *Age requirements on Google Accounts.* [Online]
Available at: https://support.google.com/accounts/answer/1350409
[Accessed 2020 March 26].

Google LLC, (2014a). *Getting Started Tutorial - Google Chrome.* [Online]
Available at: https://developer.chrome.com/extensions/getstarted
[Accessed 6 October 2019].

Google LLC, (2014b). *Google Forms: Free Online Surveys for Personal Use.* [Online]
Available at: https://www.google.co.uk/forms/about/
[Accessed 1 April 2020].

Google LLC, (2018). *Importing a Keras model into TensorFlow.js.* [Online]
Available at: https://www.tensorflow.org/js/tutorials/conversion/import_keras
[Accessed 26 March 2020].

Google LLC, (2020a). *Tensorflow.* [Online]
Available at: https://www.tensorflow.org/
[Accessed 15 January 2020].

Google LLC, (2020b). *tesseract-ocr / tesseract: Tesseract Open Source OCR Engine (main repository).* [Online]
Available at: https://github.com/tesseract-ocr/tesseract
[Accessed 6 February 2020].

Google LLC, (2020c). *tf.keras.preprocessing.text.Tokenizer.* [Online]
Available at:
https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer
[Accessed 23 March 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.

Kaggle, (2019). *Hillary Clinton's Emails.* [Online]
Available at: https://www.kaggle.com/kaggle/hillary-clinton-emails/
[Accessed 15 March 2020].

Kumaraguru, P. et al., (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.

Lee, M., (2020). *madmaze/pytesseract: A Python wrapper for Google Tesseract.* [Online]
Available at: https://github.com/madmaze/pytesseract
[Accessed 6 February 2020].

Mansfield-Devine, S., (2016). The imitation game: how business email compromise scams are robbing organisations. *Computer Fraud & Security,* 2016(11), pp. 5-10.

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

Microsoft Corporation, (2020). *How to change a birth date on a Microsoft account.* [Online]
Available at: https://support.microsoft.com/en-gb/help/12411/microsoft-account-change-birthdate
[Accessed 26 March 2020].

Microsoft Support, (2019). *Install MODI for use with Microsoft Office 2010.* [Online]
Available at: https://support.microsoft.com/en-gb/help/982760/install-modi-for-use-with-microsoft-office-2010
[Accessed 26 February 2020].

Mozilla Foundation, (2019). *Window: DOMContentLoaded event.* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Web/API/Window/DOMContentLoaded_event
[Accessed 31 March 2020].

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.* [Online]
Available at: https://aclweb.org/aclwiki/CLAIR_collection_of_fraud_email_(Repository)
[Accessed 15 March 2020].

Railsware Solutions FZ-LLC, (2012). *Mailtrap.io — Fake smtp testing server. Dummy smtp email testing.* [Online]
Available at: https://mailtrap.io/
[Accessed 31 March 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.

Sirsat, M., (2019). *Data Science and Machine Learning: Confusion Matrix.* [Online] Available at: https://manisha-sirsat.blogspot.com/2019/04/confusion-matrix.html [Accessed 3 February 2020].

StatCounter, (2020). *Desktop Browser Market Share Worldwide.* [Online] Available at: https://gs.statcounter.com/browser-market-share/desktop/worldwide/ [Accessed 9 April 2020].

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

Trivedi, P. & Sharma, A., (2013). *A comparative study between iterative waterfall and incremental software development life cycle model for optimizing the resources using computer simulation.* Chandigarh, IEEE.

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.* [Online] Available at: https://github.com/hardikvasa/google-images-download [Accessed 15 March 2020].

Verma, A., (2018). *Fraud Email Dataset | Kaggle.* [Online] Available at: https://www.kaggle.com/llabhishekll/fraud-email-dataset [Accessed 28 January 2020].

Wang, P. et al., (2015). *Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network. ArXiv [Preprint].* [Online] Available at: https://arxiv.org/pdf/1510.06168.pdf [Accessed 9 March 2020].

Wickline, M., (2001). *Coblis — Color Blindness Simulator.* [Online] Available at: https://www.color-blindness.com/coblis-color-blindness-simulator/ [Accessed 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.

APPENDICES

APPENDIX A – COMPARISON OF ORIGINAL AND ALL POSITIVELY WORDED SUS QUESTIONNAIRE

Original SUS	All Positive SUS
I think that I would like to use this system frequently.	*I think that I would like to use this system frequently.*
I found the system unnecessarily complex.	*I found the system to be simple.*
I thought the system was easy to use.	*I thought the system was easy to use.*
I think that I would need the support of a technical person to be able to use this system.	*I think that I could use the system without the support of a technical person.*
I found the various functions in this system were well integrated.	*I found the various functions in this system to be well integrated.*
I thought there was too much inconsistency in this system.	*I thought there was a lot of consistency in the system.*
I would imagine that most people would learn to use this system very quickly.	*I would imagine that most people would learn to use this system very quickly.*
I found the system very cumbersome to use.	*I found the system very straightforward to use.*
I felt very confident using the system.	*I felt very confident using the system.*
I needed to learn a lot of things before I could get going with this system.	*I could use the system without having to learn anything new.*

APPENDIX B – CONFUSION MATRIX OF MODEL VERSION 1

	Actual Positive	**Actual Negative**		
Predicted Positive	**True Positive** *(TP)* 5882	**False Positive** *(FP)* 7	**Precision** 99.881%	
Predicted Negative	**False Negative** *(FN)* 110	**True Negative** *(TN)* 5342	**Negative Predictive Value** 97.982%	
	Recall 98.164%	**Specificity** 99.869%	**Accuracy** 98.968%	**F_1 Score** 99.015%

APPENDIX C – CONFUSION MATRIX OF MODEL VERSION 2

	Actual Positive	**Actual Negative**		
Predicted Positive	**True Positive** *(TP)* 5930	**False Positive** *(FP)* 0	**Precision** 100%	
Predicted Negative	**False Negative** *(FN)* 10	**True Negative** *(TN)* 5287	**Negative Predictive Value** 99.811%	
	Recall 99.832%	**Specificity** 99.869%	**Accuracy** 99.911%	**F_1 Score** 99.916%

APPENDIX D - RESEARCH DATA MANAGEMENT FORM

Abertay University

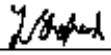
GDPR Research Data Management
Data Sign Off Form

For undergraduate or postgraduate student projects supervised by an Abertay staff member.

This form MUST be included in the student's thesis/dissertation. Note that failure to do this will mean that the student's project cannot be assessed/examined.

Part 1: Supervisors to Complete

By signing this form, you are confirming that you have checked and verified your student's data according to the criteria stated below (e.g., raw data, completed questionnaires, superlab/Eprime output, transcriptions etc.)

Student Name:	Paul Boyle		
Student Number:	1600301		
Lead Supervisor Name:	Dr Lynsay Shepherd		
Lead Supervisor Signature			
Project title:	Implementing a Browser Extension to Detect Phishing Emails Using Natural Language Processing		
Study route:	PhD ☐	MbR ☐	MPhil ☐
	Undergraduate ☑	PhD by Publication ☐	

Part 2: Student to Complete

	Initial here to confirm 'Yes'
I confirm that I have handed over all manual records from my research project (e.g., consent forms, transcripts) to my supervisor for archiving/storage	**PB**
I confirm that I have handed over all digital records from my research project (e.g., recordings, data files) to my supervisor for archiving/storage	**PB**
I confirm that I no longer hold any digital records from my research project on any device other than the university network and the only data that I may retain is a copy of an anonymised data file(s) from my research	**PB**
I understand that, for undergraduate projects, my supervisor may delete manual/digital records of data if there is no foreseeable use for that data (with the exception of consent forms, which should be retained for 10 years)	**PB**

Student signature : 

Date: 27.04.2020