Deep Phishing: Comparing Deep Learning Models at Detecting Phishing Emails from Their Plain Text Properties

Ethical Hacking BSc(hons)

School of Design and Informatics
Abertay University

Table of Contents

Table of Figures

Table of Tables

Acknowledgements

I want to say thank you to everyone throughout my journey at university and my supervisor for guidance and help throughout this project. "you will never make it into university'" – Mr Butterworth Milnthorpe school 2011

Abbreviations, Symbols and Notation

1D-CNN	*One Dimensional Convolutional Neural Network*
LSTM	*Long-Term Short-Term Memory Network*
BLSTM	*Bidirectional recurrent neural networks (BILSTM)*
Tf-idf	term frequency-inverse document frequency
GloVe	Global Vectors for Word Representation

Abstract

Context

Phishing emails have become more complicated and harder to detect in recent years. Hackers are using more complex methods of attack to foil their victims, as humans are at the front line of defence in some cases against phishing emails, they are the key point of contact for an attacker to attack. Phishing emails can be composed of both technical components such as a payload and social engineering, which foil the user to either click a link or download a file which compromises the network/system. Web filters are commonly machine learning models which have worked, but deep learning is a more accurate version of this which can help learn from the emails and become more accurate and better at detection, which can be done by learning similarities in the phishing text.

Aim

The project aims to research, and test different deep learning models used to detect phishing emails based on their sentiment. Based from previous studies referenced in the literature review, the selected models will consist of an LSTM, BLSTM,1D-CNN and a hybrid network which is half BLSTM and half 1D-CNN. The project will also use a custom dataset used to form real phishing emails gathered online.

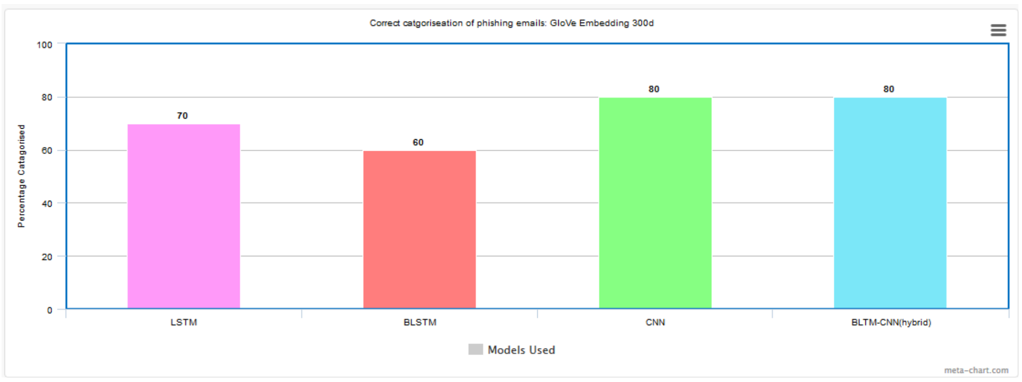
Method

This project will involve building neural networks that can categorize the plain text of phishing emails. The first stage of the project was to research what models to categorize text and sentiments such as amazon reviews and movie reviews. This was applied to phishing emails to see if they can be categorized in the same manner, as sentiment analysis is used to categorize good and bad movie reviews and news articles from public copra online. The next stage was to build a corpus file for each of the models to learn from; the corpus file consisted of phishing emails gathered online and legitimate emails which were gathered online. Once the data was gathered, each model was tested from a large embedding file size which was lowered to a smaller embedding file to determine

if a model can detect phishing emails with less contextual meaning from sentiments.

Results

All the models proved to detect phishing emails with varies in accuracy and correct classification. There was no significant drop in accuracy; each model was above 50% accurate. The increase of file embedding proved to have a higher classification rate to the model's accuracy, which was a key finding when conducting tests for this project. The average categorization with a large embedding file was 80% correct categorization.



1 Results categorisation of models against phishing emails (Glove embedding 300)

Conclusion

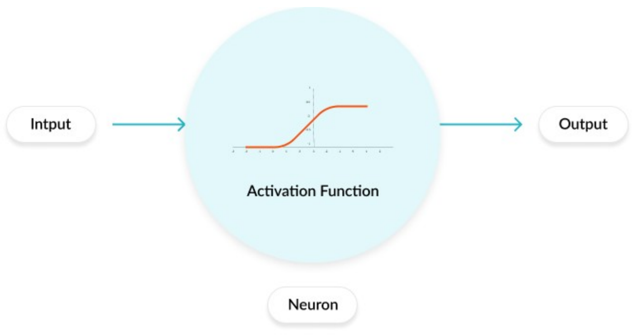
Deep learning can be used to detect phishing emails based on their plaintext, each model varies with accuracy, however there is a crucial link between the larger file embedding and the percentage of phishing emails being detected. Each model could make an accurate prediction overall to categorize phishing emails based on syntactic properties and common words used in each of the given emails.

Chapter 1.0 Introduction

A phishing email is an email that impersonates a company or a person, with the intention of fraudulently gaining information or confidential and personal information of the victim for malicious purposes. The first reported case of phishing happened in the 1990s by a hacker collective called ‘The Warez Community’ who targeted people using AOL mail. The group would target users and would then spread credit card fraud with crafted emails for the victims to upload their credit information to their web servers. The term ‘phishing’ was coined and used to describe this type of attack.

Phishing emails have always been a threat in cybersecurity as they comprise both social engineering and technical knowledge. With these two factors combined, they are the most effective way to target a company. This can be done by having emails scraped from websites such as LinkedIn. In recent years, phishing attacks have adapted and become better at foiling the victims. There has also been a wider increase in these attacks carried out by nation-states. Targeted attacks can come from the use of websites such as LinkedIn and Facebook, as you publish more information about yourself online within the public domain. Social engineering can take place to help gather information. Reported in (ISTR,2017) that spear-phishing emails are the most used method of attack and employed by 71% of their known hacking groups, other finding forms the report show that 91% of the phishing attacks are used for information gathering of the target organization. During the March 2016 election the DNC was targeted by phishing emails which allowed a backdoor to be created causing the leak of Hillary's Clinton's encrypted email server. It is reported that the specific type of phishing used for this attack was an urgency phishing email - ‘ *Someone just used your password to try to sign in to your Google account*’ - which allowed the backdoor to be created and the data leak to take place. With the use of natural language processing, this can be applied to create a deep learning network which can detect phishing emails, by sentiment analysis. This study's aim is to determine how effective different deep learning techniques are at detecting phishing emails

1.2 Deep learning



A neural network is a complex programmatic structure designed to function much like the brain. Real brains are constructed of a vast number of neurons that work in conjunction with one another to identify and make decisions and make sense of their surroundings. Neural networks constructed from simple objects which are named neurons, in reference to the human brain and how the connections work. There are many different types of neural networks with different functions and methods of how they process data and train/learn. A basic example of how a neuron works in the neural network/deep learning model is shown in figure 1.

1.2.1 Activation function

An activation function is a type of mathematical function which includes taking an input and applying a calculation to produce one or more outputs, each output can be graphed on a cartesian plane to show how it performs. An example of this would be SoftMax, which takes values and squashes them between one and zero (true or false) depending where they lie on the curve is the outcome for the output function of the model.

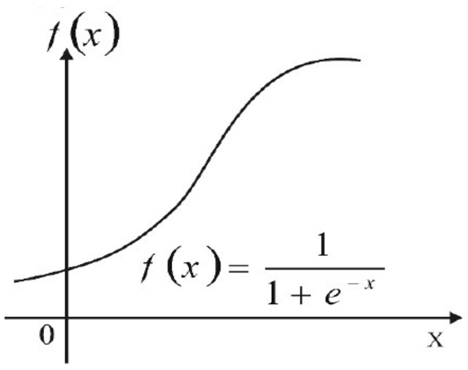


Figure 3 Sigmoid function (Snehasish Dutta Gupta (2014)

1.2.2 Convolutional Neural Networks (CNN)

A convolutional neural network is commonly used for image detection but can be used for natural language processing as in theory, it can work in a similar way.

A conversational neural network comprises three main elements. Convolutions are represented by the * symbol in Mathematical equations, X will represent the corpus text (phishing email corpus), the filters will be represented as F.

$$Z = X * F$$

1. Input layer holds the values of the corpus file or image file used.

2. Convolutional layer determines the output of the neurons via the calculation of the weight between them, which is done by Reflected Linear unit (reLU)

3. Pooling layer reduces the parameters within the activation.

4. The final layer of the model is the fully-connected-layers of the CNN which breaks up the data and learns from the samples, with the use of reLU and other activations which can drastically improve the models learning rate.

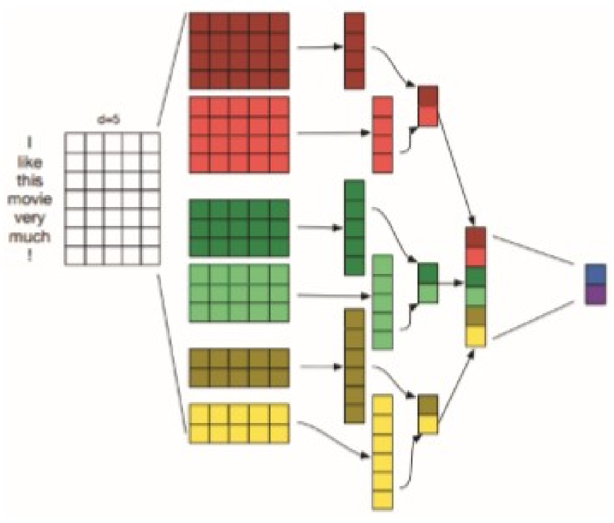


Figure 4 CNN Architecture Joshua Kim. (2017)

1.2.3 Long Short-Term Memory Networks

LSTM is a deep learning method that is a linear learning method in which data is passed through ,and after each time the sequence is mapped and learned. There are different types of LSTM models that can be used for different purposes; this is due to the layout of the model. The one used for this project will be a many to one architecture where there are different architectures which are fit for different purposes. Many to one help refine the corpus text read in to produce one fixed sentiment to a tag.

The LSTM learns the frequency depending on what came after it; this is then fed to the next layer with an added weight proving the given value should go there, which helps the model learn and make a prediction based on the weighted word/ value[1]. After each cell has been passed, the cell state will either read or write from one another, thus learning.

The first component in an LSTM gate is the forge layer; this outputs a number between 1 and 0 if the forget gate outputs 0 it will forget the pattern if it outputs one it will keep it and move to the next stage. The next layer is the input gate which decides what value will be updated.

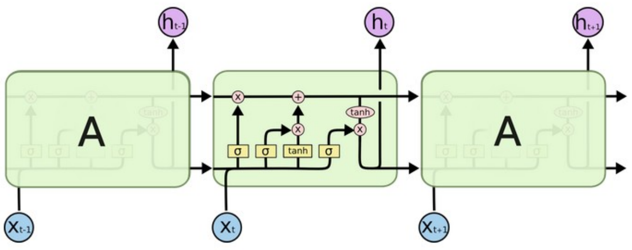


Figure 5 LSTM architecture Colah (2015)

1.2.4 Bidirectional - Long Short-Term Memory Network

The bidirectional Neural Network was created in 1997 (M. Schuster and K. K. Paliwal,1997) that the paper which discussed the use of LSTM layers both going forward and backwards after one computation. As the LSTM data is being passed each time, it makes a prediction based upon what came before and after it. As the gate is bidirectional, it is commonly used within natural language processing because of the added functionality.

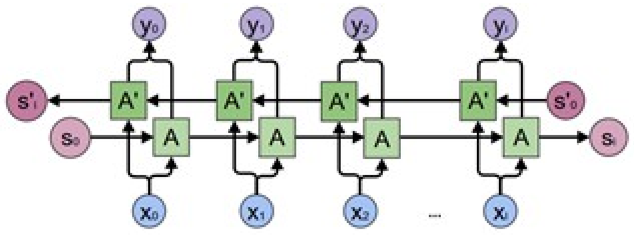


Figure 6 Bidirectional LSTM Colah. (2015)

1.2.5 Hybrid Networks

A hybrid network is constructed to have two different networks within one network; this is done at the end of each layer allowing the data to be processed one way and then filtered and processed by another way, this is a new type of deep learning method which can combine different types of networks together in a sequential stack.

1.3 Word Embeddings

Word embeddings are a way to represent words and language for machines to understand where the words come contextually for the sentence. Word embeddings are based on a distributed hypothesis that gives words meaning in each context. Embedding files help machine learning and deep learning models learn where words fit in a sentence. There are different ways which word embeddings are used and different types, such as Word2vec and Global vocabulary word representation (GloVe).

1.4 Word representation

Word repetition is a method used to represent words in corpus text and allocate words of importance, which can be common words of the document; an example would be the word ‘king’ in a history book, how many times it appears and what appears next to it. Frequency-inverse document frequency (TF-IDF) is the founding base of word representation and embeddings of word files, TF-IDF works by searching a corpus or a document and weighing each word contextually with a weighted value.

1.5 Natural Language Processing

Natural language processing is a technique used to learn a language with machines and deep learning methods. Natural language processing works by reading in corpus text of pre categorized files such as a CSV file with a label once the data has been learned the model can then make a prediction based on what text the model has been given. When the model is being trained, it will gather word's context and understand what should and should not come before words. An example would be the Russian language to the English language commonly in Russian 'a' isn't used in sentences an example would be ' I need doctor' which if a language model had Russian sentences and English language sentences it would learn that 'a' is not commonly used in the language and make a prediction saying that 'i need doctor' is a Russian sentence.

1.6 Good word attacks

A good word attack is a method of attack where an hacker can edit the amount of words in the emails which will go unnoticed through web filters, this is because the standard filters will be able to recognize regular emails which fit a specific character size, anything above the given size will be flagged and, anything below the size will also be flagged. A good word attack is an attacker finding the perfect amount of words to allow the email to bypass the filter.

1.7 Project focus

The proposed outcome of the project is to have one deep learning model, which is the most efficient at detecting phishing emails. By researching techniques already being used, then applying deep learning to a corpus file of legitimate and illegitimate phishing emails. The selected models will be based on the literature review, models will then be used to classify phishing emails to test their efficiency overall.

1.7.1 Research question

Within the given scope of the project, the result will be able to answer the following question:

"How effective are different machine learning algorithms for phishing email detection?

A study into how effective different deep learning techniques are at detecting phishing emails."

A large corpus file must be created, then different deep learning models created to train the form which will then be given emails to categorise to test their ability to categorise phishing emails to see if they can detect them based on their syntactic properties. Each model will be analysed on its performance and ability to determine which is a ham and which is a spam email.

Chapter 2.0 Literature Review

This chapter will discuss methods used by attackers to bypass web filters, phishing emails and how users are susceptible to them. Deep learning techniques used to categorise text and different ways in which deep learning can be applied for cybersecurity. Embedding and categorizing language.

2.1 Web Filters

Filters are a method to remove junk mail and phishing emails, these types of filters can be deployed quickly and look for specific patterns within the email to see if it is legitimate or not. A standard method used is the Naive Bayes algorithm which looks for an outlier which is an anomaly from the average email that is passed through the filter, as discussed in (Daniel Lowd Christopher Meek 2005). Methods to bypass this type of filter by having a 'good word attack' which will use many common words found within phishing emails and spam emails which are referred to as Buzzwords for the filter to pass as detectable. With appending words or decreasing sets of words, this can allow the filter to pass the malicious email as legitimate.

One way to combat good word attacks is to have a model automatically update the spam list regularly with detected spam and phishing emails, this allows the training corpus to continually grow and the accuracy of the model to improve while detecting phishing emails. The algorithms used commonly in web filters tend to be linear regression models, that prove to be accurate when testing against phishing emails, but the use of the updated corpus file can give a misclassification rate as shown in paper (TA Almeida,2010).This may have been down to the algorithm used to detect the phishing emails as it was a linear regression model which is simplistic for language processing.

Machine learning can be used to detect phishing emails, through the use of an RNN, which can detect longer types of phishing emails as it will not take string length into account and will limit the chance of a good word attack allowing the phishing email to bypass a web filter. Other limitations from linear regression-based models were present in the tests conducted such as string length and file headers used, which an RNN can handle when learning from the corpus. Limitations of some previous studies are that the models used may be outdated ,and a newer study may need to be conducted; with more recent types of phishing email the study's finding that the corpus automatically updating decreased the models' accuracy of detecting phishing emails, which suggests having a large corpus with a mix of many types labels as phish to all the detection to be accurate. The limitations of most of the web filter related studies are that only one model has been predominantly used; results would be more comprehensive if the study conducted used many types of model and explained the source of the sample data. Other issues in previous studies that there was no prediction against any of the emails to show the model was indeed accurate-only the model's percentage accuracy.

2.2 Phishing emails

Phishing emails are a common way to attack, setting out a way for someone to give away personal data; it was reported in 2019 by the Phishing Labs, fraudulent emails have increased by 28.9% during 2019. Humans are susceptible to these kinds of attacks because of trust ,and at face value, phishing emails can seem legitimate with the language used and the layout of the email.

People can be bad at detecting phishing emails, because they have not received relevant training on email detection and do not know what to look out for; specifically, keywords can be mapped out of phishing emails and classified. Emotive language may be used to cause the feeling of urgency, e.g. the victim has lost money or is in trouble and needs to act now. Other types of attack can be used with a phishing email being used to deploy malware or ransomware, such as Petya and Orris was used to spread ransomware through the financial sector (Acroins,2017). These were spread through spear-phishing attacks where the language is targeted at a user or looks like it has come from a user, these attacks are more sophisticated and require a social engineering approach to allow the victim to fall for it. As reported in the NCSC 2019 (NCSC 2019) report 80% of businesses and charities data breaches came from phishing/spear-phishing based attacks.

Some users are susceptible to phishing emails because the language can cause panic and cause users to open the attachment or click the link to the landing page of the malicious link. Research from the Australian Intelligence Agency on phishing email susceptibility shows that there is a link to participants' cognitive ability and their susceptibility on clicking a link from a phishing email. (Marcus Butavicius, et al ,2015) some aspects of the study were very limited as only a finite number of participants were involved with the study; it would have been improved if there was a larger sample size of participants.

Due to the smaller sample size, this can lead to a more significant margin of error, meaning the results of a larger sample will be more accurate for statistics. One bias of the study is how much of the report has been redacted or classified? This seems to be an issue regarding intelligence agencies publications. Under what level of a security clearance was used to conduct the study and how was it released and made public? Another study on susceptibility was conducted with students with larger sample size. A study with a larger batch size proved to be useful regarding phishing emails. One study conducted by the University of Maryland and Baltimore, (Alejandra Diaz,2018) reinforced the theory that different groups of people are more susceptible to phishing emails; the study had a large batch size of 450 students with a mix of humanities and arts, and others from STEM-based subjects. Gathered from the studies,93% of students opened the emails. After reviewing the studies, the results showed that people are susceptible to phishing emails regardless of their technical knowledge and background. The main emails which the participants fell for were banking emails, which caused panic in the participants as they started to think their banking details had been changed or their account had been accessed.

2.3 Deep learning

There are different ways which text can be classified with machine learning and deep learning both having issues which have been raised in different papers.

One is the complexity of the models used, which can cause the training of the model to stop, due to the model underfitting or overfitting, which can cause it to be slow and not learn correctly. Another is the data being used if their characters are misread and tokenized; this can cause the model to train from incorrect data as shown in a study conducted by the University of Oxford. An RNN (Lukáš Halgaš1, et al. 2019) was used but was proven too unstable, with different accuracy rates throughout the study. Other issues were identified in different studies, such as learning and adding to the corpus after each epoch, dropped the models' accuracy; meaning it could wrongly classify each phishing email yet still add the prediction to the corpus for the model. Another limitation of the study is that there was not an embedding file used for the model to train on correct language from the emails, only the spam and phishing emails which may have affected the accuracy of the university's study on recurrent neural networks. The results considered where the accuracy and the precision of the models were; these were graphed by accuracy by epoch, which was the method used to collect the results of the current study. Other studies compared the effectiveness of deep learning natural language processing by comparing the effectiveness of Deep neural networks, specifically a CNN and an RNN. The study aimed to categorise different sentences with four different corpus files; issues which arose during the study,different sizes of corpus files which could mean that some of the tests had some bias as the corpus size would affect the overall results of the study. Prediction Is an excellent way to measure how accurate the model is. Different sizes of prediction text may alter the model’s prediction. The study merged different sentiments from different datasets that are publicly available online (Nhan Cach Dang, et al. 2020).

The study shows that the accuracy of a network can drastically drop depending on the quality of the dataset it has been given, which was noted throughout the research, showing that the dataset may cause issues with prediction and accuracy. Another Issue within some of the deep learning papers is that some only discuss the model’s accuracy during sentiment analysis. Not if the model can make a prediction based on what sentiment it has been given; which defeats the point of creating an accurate language classification model if it is not used to classify anything and only the accuracy and f-score are measured.

2.4 Semantic role labelling (SRL)

Within natural language processing, semantic role labelling is the process which assigns labels to sentiments within the corpus, the most common example of SRL is *who did what to whom.*

- *who - Agent*
- *Did* - Predicate

- *What to* - Theme
- *Whom* – Location

The primary use for SRL is to determine patterns within the text and append it to the given label. There can be correlations between the use of verbs within phishing email and non-phishing email to categorize them (Matthew R. Gormley et al. 2014). The study did find a correlation between the phishing emails and the usage of given verbs, for example, *'confirmed'* which appeared 43.88% more in phishing emails than legitimate; the study was conducted with machine learning and the classifier being a decision tree. Deep learning for SRL can use both LSTM and a CNN for analysing and categorising the CONLL-2012 dataset (an open-source dataset from the wall street journal).

The issues with the studies are that both are vague in ways in which the data is categorized into different sections; what made the data split into subsections? The use of BiLSTM and RNN (Luheng He,et al.,2017) proves why deep learning is an accurate way in which data can be categorized and tagged with efficiency. The use of the BiLSTM combines the architecture of two different models into a hybrid which may be a more accurate way of detection as BiLSTM first processes the data. It goes both ways for learning the sentiments; the data is learned from right to left then left to right, which shows high accuracy. SRL can have other uses also such as a plagiarism detection-based mode.

2.5 Embedding and Vector Representation

Global Vectors for Word Representation (GloVe) is an excellent model used for natural language processing and helps the prediction of words and language based on statistical prediction. Before the use of GloVe, embedding layers used a distance-based model, such as Described embedding which worked on the distance between keywords, then a prediction was made on what was next and before it. GloVe uses a more accurate version of statistical prediction. An example is how GloVe works *'King is to queen as man is to woman.' (Jeffrey Pennington, et al.2014)* where the text would be encoded then the embedding would find the tokenized values within the corpus and find significant words next to the queen; '*queen = man-woman'*.

The central concept behind GloVe is that the meaning of words can be extracted via the probability of words of interest from a sentence. A sentence is broken down in the main embedding layer, and compared with GloVe for the probability of how common the word is within the layer, a prediction is made on where the word is within the sentiment. The main concepts of word vector representation are 'how statistical word data can help represent words and meaning' which was proven to be useful and accurate within the testing stages showing that it was accurate throughout the tests on large corpora.

The conclusion of the study was that the GloVe model is more accurate with a smaller corpus of values than the Word2vec model used. GloVe in the accuracy tests proved to be the most accurate with the standard deviation of 93.2 while Discrete embedding had 91.0.

The GloVe embedding has become a standard for word embeddings used for deep learning methods as it is efficient and uses statistical prediction for each word and then learns the meaning based on the prediction. The publication provides an understanding of the workings of complicated embedding models and how they can produce accurate test scores. The strengths of the resources considered are that they provide a comprehensive look at GloVe; breaking down how it works into the prediction of words and showing the equations used to make the prediction. One negative is that the study is becoming somewhat outdated as newer NLP models have been made, such as BERT that is based on word2vec. The primary bias from the study is that there is only one comprehensive study on GLove, which is produced by Stanford University, who developed GloVe embeddings, so is likely to be biased. word2vec is an encoding method which uses SRL to match the encoded values to a vector glove - a large corpus file of embeddings. While both prove to be useful, the use of word2vec will be limited to syntactically correct spelling and words which can affect accuracy.

Issues have been found with embedding files such as it may have gender bias and appear to be sexist. The way the embeddings have been created from old books and online articles which sadly may have some sexist connotations in; it has been found that the word *woman* is more closely related to the word *maid* than doctor or teacher(Jieyu Zhao, et al. 2018). However, there have been cases where the use of embedding files with gender-neutral language had no drop in accuracy or categorizing data trained on CNN networks.

2.6 summary

Based on previous work, studies show that humans are more susceptible to falling for phishing emails that pass through the web filter by looking at legitimate emails. Deep learning methods have been shown to learn from large corpora of different sized texts, and the use of SRL can help categorise words to a label. It has also been shown that words have the contextual meaning, which can be used to help deep learning models become more accurate at categorising words.

Chapter 3: Methodology

This chapter details how models were selected for testing, how the corpus data was created. The reasons why the specific types of phishing emails and legitimate emails were selected, and how each model was developed and tested to ensure consistency. With the models being tested to determine the effectiveness of each deep learning model.

The project’s main objective is to review and evaluate the effectiveness of deep learning models used in natural language processing when applied to the detection of phishing emails.

3.1 Methodology overview

- Review the effectiveness of different deep learning methods used for natural language processing.
- Create a dataset of phishing and legitimate emails.

- Implementing each model and running on a GPU for maximized performance.
- Review the test data and test the model's detection accuracy. Give the model a sample email to see if it can detect a phish or a non-phishing email.
- Record and review the overall results during the testing stage of the project.

The first stage of the development was to split the training data into test and training sets, which was then encoded. The embedding file was then loaded into the models to allow the model to understand the corpus data placement in the sentence, giving the words contextual meaning thus becoming more accurate. This is calculated by each word and what comes before and after the word, a weighted value is placed on the word for the times it occurs during the sentence.

3.2 Model Development

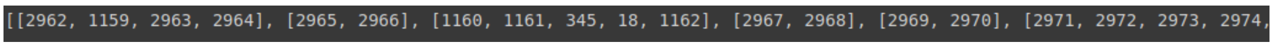
The accuracy and validation loss will determine the model’s effectiveness, this is to check how accurate each model is learning if the validation loss has increased; this shows it has been learning incorrectly after each epoch of training.

The models were developed over several months to ensure enough time for testing of each model and to gather the results. The four models selected were based on the literature review. Most studies on deep learning compared LSTM models with CNN models and bidirectional models, with different architecture; hence the decision for the BLSTM model. Studies tested the effectiveness of an LSTM model and different types of CNN and RNN models, based on their validation and accuracy. The testing methodology was based on a study of deep learning models for sentiment analysis, which fed the model’s data to predict if it was a movie review or a tweet. This study will use phishing emails to see if models can determine they are a phishing email or not by analysing the sentiment form the corpus text. The results gathered will be the f-score which has been referenced through the literature review, for most of the deep learning overall accuracy, which has been used as a measurement in this study.

The first stage of the implementation was to have the text split into test and training data. Which is to ensure the model can learn from some data whilst making predictions based from other data; therefore, a data split is made to allow the model to learn from X and make predictions based from Y.

The data was encoded which converts the plain text into numeric values removing all punctuation with TF-IDF applied to the given string.

By increasing the embedding size for the models to learn from, this may increase the overall accuracy as it will give the sentiments more contextual meaning to the words, and the added context with the supervised learning may create a more accurate model to detect phishing emails.



```
[[2962, 1159, 2963, 2964], [2965, 2966], [1160, 1161, 345, 18, 1162], [2967, 2968], [2969, 2970], [2971, 2972, 2973, 2974,
```

Figure 7 Example of encoded data with vector representation

Then each model would learn from the phishing corpus, which learns from the tag to the sentiment, then what words are used on average to make that definition. All the models were run on Google colab which is a free cloud service that allows developers to run and test code. The project was run on a Tesla k80 GPU, which is a high-performance GPU that has 4992 cores which allows the models to be run quickly.

3.2.1 Models used

3.2.1.1 Convolutional Neural Network (CNN)

- the data is squashed down into small sentiments which means it learns sentiments more specific.
- Three main features; filters, output then pooling. This may increase accuracy due to the detail of which the data is being processed

3.2.1.2 Long short-term memory (LSTM)

- Process any input length.
- Learns quickly and takes in historical information based on the corpus.

3.2.1.3 Bidirectional Long short-term memory (BiLSTM)

- LSTM gate passes right to left then left to right, giving it the full context of word placement.

3.2.1.4 Lstm-cnn (hybrid)

- A basic example of a deep learning model.
- Uses both an LSTM and a CNN layers

3.2.2 Long short-term memory (LSTM)

A long-term short-term memory network was implemented with python and used Keras, much like the other networks mentioned. The LSTM architecture is somewhat different from the 1D-CNN network as shown in figure X. The LSTM has layers which are how the data is learned from when reading it in. The only other difference is the placement of the embedding file; it works much the same just in a different model’s architecture.

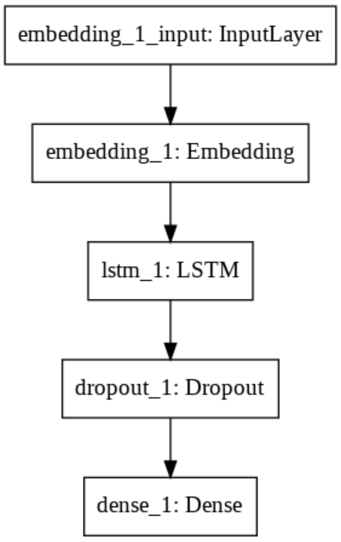
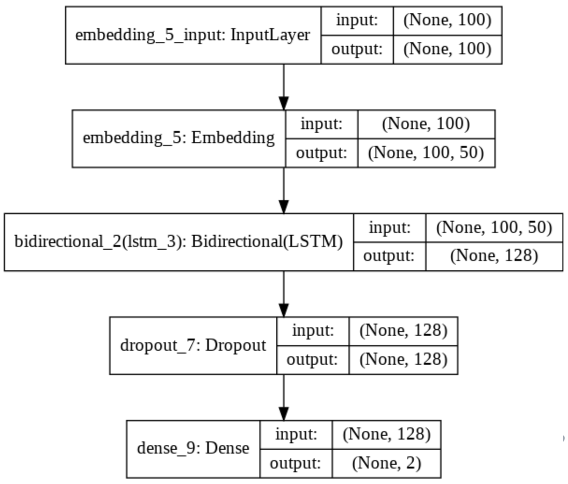


Figure 8 LSTM model architecture

3.2.3 Bidirectional Long short-term memory Network (BRNN)

A BRNN was implemented to see the effects of a bidirectional RNN model compared to CNN and RNN networks. The model works like a recurrent neural network, but the LSTM layers travel both ways right to left then left to right, with the added use of GloVe embedding this would, in theory, create a more accurate model with added context. The Batch size and other factors were kept the same to allow all variables of the testing phase to be the same.



3.2.4 One Dimensional Convolutional Neural Network implementation

The 1D-CNN was implemented with Python and Keras. Layers of the model were kept limited to ensure that the testing of each model was roughly the same, if the model had more layers compared to the LSTM model this would prove to be more biased which may be more accurate thus making the study unfair as each layer has to be kept somewhat the same during the testing process.

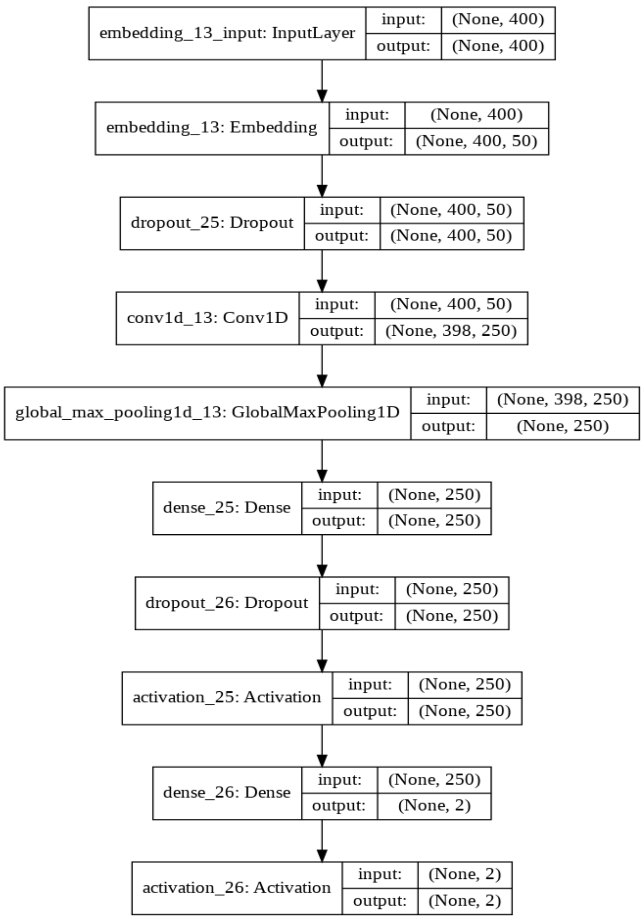

embedding_13_input: InputLayer | input: (None, 400) | output: (None, 400)
embedding_13: Embedding | input: (None, 400) | output: (None, 400, 50)
dropout_25: Dropout | input: (None, 400, 50) | output: (None, 400, 50)
conv1d_13: Conv1D | input: (None, 400, 50) | output: (None, 398, 250)
global_max_pooling1d_13: GlobalMaxPooling1D | input: (None, 398, 250) | output: (None, 250)
dense_25: Dense | input: (None, 250) | output: (None, 250)
dropout_26: Dropout | input: (None, 250) | output: (None, 250)
activation_25: Activation | input: (None, 250) | output: (None, 250)
dense_26: Dense | input: (None, 250) | output: (None, 2)
activation_26: Activation | input: (None, 2) | output: (None, 2)

Figure 10 One Dimensional Convolutional Neural Network Implementation Architecture

3.2.5 Bidirectional Recurrent - One Dimensional Convolutional Neural Network (BI-CNN)

A BI-CNN model is a type of network which is a linear stack of layers and the combined stack of an LSTM layer then a CNN layer; this model was selected as it will see if the combination of the two is more accurate during training, this was developed with Keras, the choice of the model being sequential allows the linear stack to be implemented.

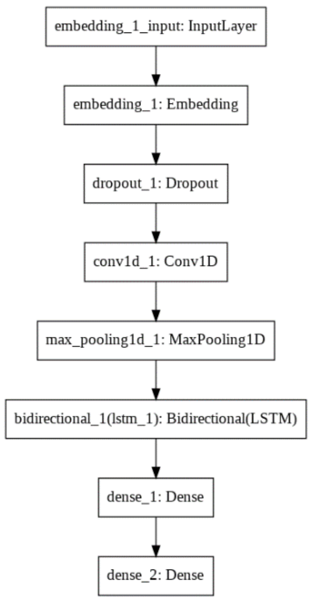

embedding_1_input: InputLayer
embedding_1: Embedding
dropout_1: Dropout
conv1d_1: Conv1D
max_pooling1d_1: MaxPooling1D
bidirectional_1(lstm_1): Bidirectional(LSTM)
dense_1: Dense
dense_2: Dense

Figure 11 1d CNN architecture

3.3 Phishing emails

The next stage of the project was to gather a wide range of emails. This was used to create the corpus to train the models on, samples were collected from app.anyrun which is a cybersecurity website that is a community-based collective; this lets the users upload phishing emails and malware samples from the wild to be analysed by other members of the community.

Examples from the literature review and some public data sets such as ham and spam; comprising spam and non-spam messages. The Eron dataset (William W.

Cohen,2015)., which is a large corpus used for sentiment analysis, that consists of internal company emails over a period of months. Other phishing emails were gathered from the literature review on humans detecting phishing emails, which had a corpus set of HSBC emails and PayPal phishing emails(Millersmiles,2014). The importance of legitimate emails was also needed; the use of legitimate emails was needed so the model can detect grammatical errors and issues with int corpus to learn. The phishing email data was gathered from other studies mentioned in the literature review which tested the user’s susceptibility to phishing emails; the data was stripped of special characters and all non-UTF-8 encoding, which was fed directly into the corpus file. The data was classified with spam and ham; the reasoning for this is that other papers detecting phishing emails used the same naming and techniques to categorize their corpus data.

3.3.1 Creating the dataset

The corpus data set was created by using phishing emails and regular emails. The first stage was to import regular emails; Other datasets were imported such as the Ham and Spam dataset, this is a fundamental corpus used which has regular and spam SMS messages. As the language varies, this is commonly used for research for natural language processing. This method was also used in studies referenced in the literature review such as this study (Lukáš Halgaš, et al. 2019) by the University of Oxford on recurrent neural network detection of spam emails. The spam and ham dataset are based on SpamAssassin's public corpus.

Different types of phishing emails were used; commonly banking emails were used for the corpus as they were the most common type that victims are more susceptible to click. Banking and PayPal phishing emails (Alejandra Diaz, Alan T. Sherman, Anupam Josh,2018) other phishing emails where downloaded and collected this was done from app.antrun ,which is a cybersecurity site where users can upload their investigations and malicious files and hashes for security teams, and recherche to use and keep up to date with recent security threats and trends.

3.3.2 Characteristics of corpus file

As shown in figure 10, this is a word cloud model, is a visual representation of the corpus data which shows the most common words within the dataset. When the deep learning model is being trained on the dataset, it will pick these words listed in figure 11 that are the most important. These words are then appended to the spam tag, which helps give the model an understanding of common words used for a spam email.

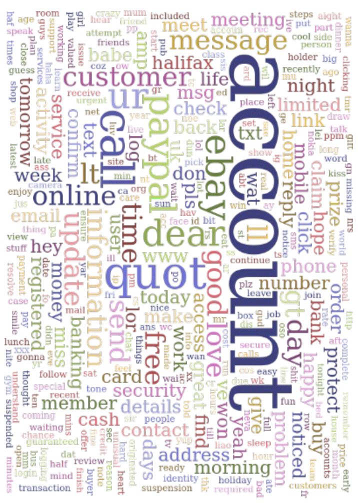


Figure 12 Word cloud representation of corpus

3.4 Testing methodology

The testing methodology was based on previous studies on deep learning methodology referenced within the literature review. Which compared the accuracy of the models and the loss of the models. The main tests of the project were increasing the epochs of the models then seeing how accurate they are after being tested on the corpus file.

The testing of the models with 100 epochs, after each time the models were run and the results were gathered, the embedding size increased from glove.6B.100d to glove.6B.300d. After each run, the model had a sample text to classify whether it is a phishing email or not. The models were given phishing emails to categorise as this was mitigated in (Lukáš Halgaš, et al. 2019) that highlighted the importance of having the models be used to classify legitimate emails. Their study only tested the accuracy of a trained model on the data they had gathered, and not classifying them.

3.4.1 Bassline

This study compares the use of a CNN with an LSTM and a BLSTM and a hybrid model. The use of layers and sizes had to be standardized to allow the study to mitigate any bias when gathering the data and to compare the results. The data was encoded, which was then split into the same sample sizes and batch sizes; so the model can learn from it correctly and mitigate any bias of the model's architecture is different; this is to eliminate any bias for the study and try to keep each model the same.

LSTM and the BRNNs LSTM units were kept the same to allow the standardization and the models to say somewhat the same despite the architecture being different. The LSTM layers are much slower than the BLSTM with the data processing, which the study is not conducting speed tests. The speed is taken out of the results. The study is comparing the accuracy and learning rate over epoch size, and glove embedding size increased. Then having the models then classify sample phishing emails.

3.4.2. Optimizers

Adaptive Moment Estimation (Adam) is an algorithm made explicitly for deep learning. Adam uses *momentum* which learns to form its past steps and gradient to work out what direction to go. Adam incorporates momentum by increasing the use of fractions and the previous gradients to the most recent gradient, thus becoming more accurate after each epoch.

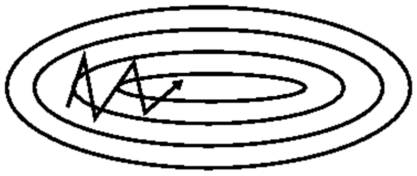


Figure 13 example of calculating distance without momentum (inefficient) Sebastian Ruder. (2016).

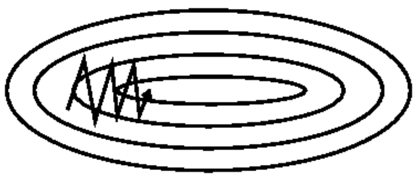


Figure 14 example of calculating distance with momentum (efficient) Sebastian Ruder. (2016)

3.4.3 Activation

SoftMax was used as the activation, which is the way to polish off the model's predictions and learning, which is used in the final layer of the neural networks. The use of SoftMax applies decimals to the probabilities at the end of each feature, by taking in the input of a vector in the (ham or spam), which must add up to 1 the function calculates the probability of the distribution inputs. In the use of the project, it is used to help determine the probability of a spam or a ham email which is then labelled and categorized. An example can be a Neural network using multi-label classification classifying food types by text. As shown in figure 17

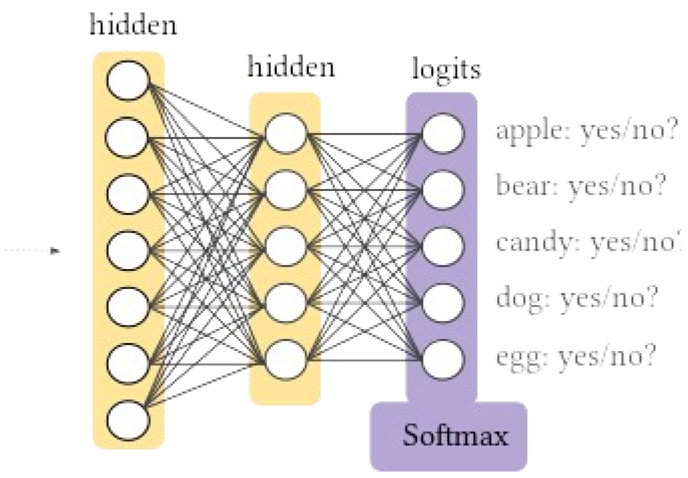


Figure 15 SoftMax function classifying food types (Google developers ,2017)

3.5 Testing baselines

Each model was tested against ten illegitimate emails and three legitimate emails. All a mix of different types of language syntactically, this to determine if the models could tell the difference between different scams and by their language. Two different functions which were used; predict and predict_proba

The primary predict function will return the label it thinks the sample text is assigned to, spam or ham predict_proba returns how accurately the model scored the text. The selection of emails used was kept separate to the corpus data which would allow the models to be tested against independent data.

3.6 Sample emails

Sample emails were gathered to test the models on. The emails gathered were kept separate to the phishing email corpus that would stop any bias within the testing phase of the project.

The range of phishing emails gathered is to see if the models can detect emails with different use of language as this would test the efficiency and accuracy of deep learning models detecting phishing emails. As shown in table 1 are example emails which the models will be used to classify in chapter 4. The emails used were kept separate from the corpus file so they would be hard to detect, the testing emails were gathered from the University of Berkeley open phishing email database. X The website has examples of phishing email split into their categories, this was useful as it allows the deep learning models to be tested against different types of language and the email size which may look like a regular email as this can be an example of a good word attack.

Table 1 Example of phishing emails

Phishing email plain text
I’m aware is your password. You may not know me, and you are most likely wondering why you’re getting this mail, right Overview: I installed a malware on the adult vids (sex sites) site, and there’s more, you visited this site to have fun (you know what I mean). Once you were there on the website, my malware took control of your browser. It started operating as a keylogger and remote desktop protocol which gave me access to your webcam. Immediately after that my software collected your complete contacts (you have a good taste lol…
Are you around? I need to pay a vendor with the blucard
Dear User,Someone else was trying to use your Berkeley ID to sign into iCloud via a web browser.
To:Hello, Please refer to the vital info I've shared with you using Google Drive. Click https://www.google.com/drive/docs/file0116
Dear customer your account has been locked please login into click to see https://dkdkdkdkddkkkkd

Dear HSBC Bank Personal and Premier Banking Online customer! Our Maintenance Subdivision is performing a scheduled Banking On-line Service upgrade By clicking on the link below you will begin the webform of the customer's
Good Morning We received 2 high value CHAPS payments requests into the branch today.Please complete and sign the attached documents and return for processing.We require this information before we can release the payments to your account.
Dear User,Someone is trying to use your password please update now.

3.7 Gathering Results

There are different ways to show how each model has learned and displayed the results. Which can be done by showing the accuracy, f-score, a confusion matrix to show the models predictions on the emails it has been given to classify.

3.7.1 Accuracy and Loss

Accuracy is how the model's performance can be classified. This value will be presented as a percentage for the results of the paper. Loss is harder to calculate as it is not a percentage; it is several errors within the training or validation of the model. This percentage shows that the model has been incorrectly trained, producing poor results linear to the loss accuracy.

Defining the relationship between the two is an excellent way to see the model's effectiveness and the way it learns over the set number of epochs.

3.7.3 F-score

The f-score is used to calculate the models' test accuracy. This equation uses precision and recall doing so. The f-score provides a better measurement of the accuracy by measuring both the recall and the precision

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

TP = True positive

TN = True negative

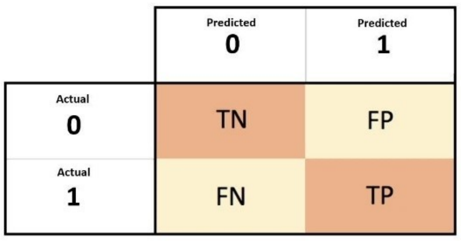
FP = False positive

FN = False negative

Figure 16 calculation for f score Koo Ping Shung. (2018)

3.7.4 Confusion matrix

A confusion matrix is used to show the summary of the model's predictions. This matrix is displayed by having the number of correct and incorrect predictions summarized as numeric values. This method helps indicate issues found with it when it makes predictions by showing what overall result it got correct.



	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

Chapter 4.0 Results

This chapter will display the results conducted based on the informed methodology.

This chapter will detail the overall results of the models when testing. This chapter will cover the effect of different embedding files on accuracy. The results will be presented, showing both the accuracy and the f- score, and the percentage of correct phishing emails categorized will be listed as a percentage for comparing each of the deep learning models. The tests aim to define what is the most efficient deep learning model at detecting phishing emails based on their plain text. All graphs can be found in the Appendices section.

4.2 Parameters

Table 2 Parameters used for each model

Epochs	100
Batch size	200
Sequence length	100
Test Size	0.5
Kernel Size	7
filters	250

4.3 Overall results

Each test was conducted by running 100 epochs with the parameters as listed in table 2 after each test was running the embedding size was lowered from glove 300 to glove 100d, this was to test if the large embedding file could affect the accuracy and prediction of each of the deep learning models against phishing emails.

4.3.1 glove.6B.300D Results

Table 3 Results for each model tested with GloVe embedding 300D

Model	**Accuracy**	**F1**	**Percentage of correctly categorized phishing emails**
LSTM	66.86 %	0.44836	70%
BLSTM	69.71	0.41234	60%
1D-CNN	61.13	0.72353	80%
BLstm-cnn(Hyybrid)	58.60 %	0.42876	80%

4.3.2 glove.6B.200D Results

Table 4 Results for each model tested with GloVe embedding 200D

Model	**Accuracy**	**F1**	**Percentage of correctly categorized phishing emails**
LSTM	64.68 %	0.3324	50%
BLSTM	69.09 %	0.3246	50%

1D-CNN	62.05%	0.4235	90%
BLstm-cnn(Hyybrid)	59.05%	0.5247	30%

4.3.3 glove.6B.100D Results

Table 5 Results for each model tested with GloVe embedding 100D

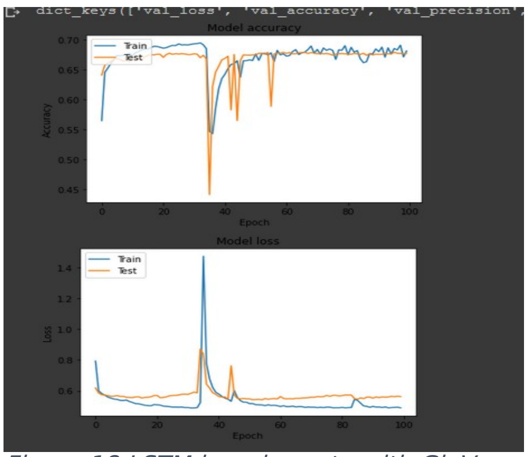
Model	**Accuracy**	**F1**	**Percentage of correctly categorized phishing emails**
LSTM	67.72%	0.1534	20%
BLSTM	69.14%	0.2356	20%
1D-CNN	65.48	0.8457	40%
BLstm-cnn(Hyybrid)	59.13%	0.46235	30%

4.4 Results analysis

4.4.1 LSTM

The LSTM model was accurate during testing with 70% of emails identified as a phishing email, the model when learning had some spikes that do indicate it was unstable during the training process. The model did prove to be accurate overall when tested against phishing emails, it was noted the slowest at learning.

As figure 16 shows the LSTM had a few drops and spikes in the models training, this may be because of the embedding causing the loss of accuracy as this happened throughout the models training when given a large embedding file. The model did not appear to overfit during training.



```
dict_keys(['val_loss', 'val_accuracy', 'val_precision',
```

Figure 18 LSTM learning rate with GloVe 300d

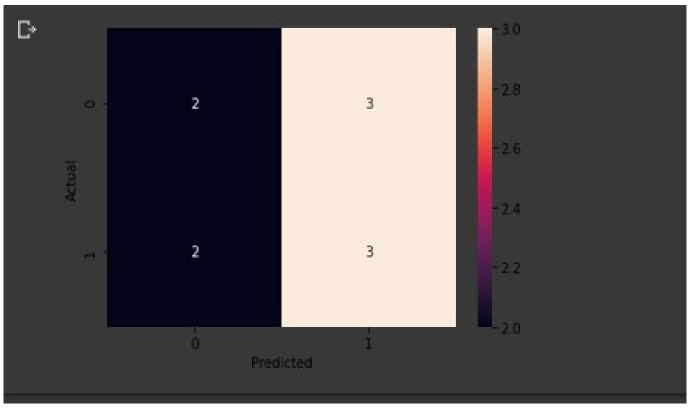
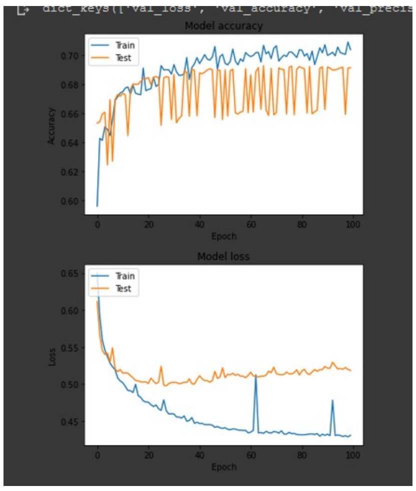


Figure 19 Confusion Matrix for LSTM classification

4.4.2. BSLTM

The BLSTM was one of the more accurate deep learning models when training. This may have been because of the bidirectional function of the model, meaning it would quickly make assumptions of word placement and learn quicker as it reads; left to right then right to left in each of the LSTM gates. The model proved to be accurate during training, but overfitting as shown in figure 17. The test data was very unstable and wavy, this indicates the model proved to be accurate during training, but overfitting, as shown in figure 19. The test data was volatile and wavy. This graph indicates the model was finding issues in categorising the ham and spam categories of the corpus file.



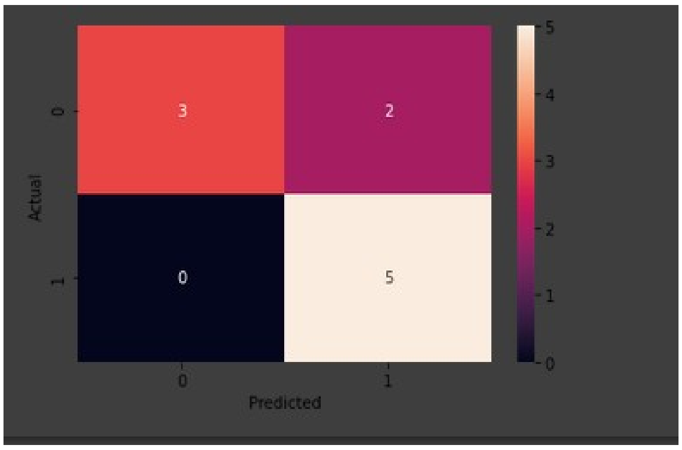


Figure 21 BiLSTM learning rate with GloVe 300d

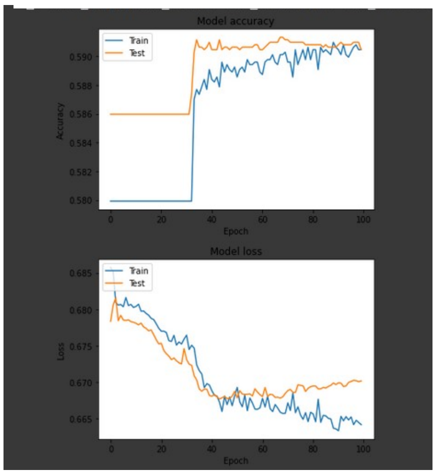
Figure 20 Confusion Matrix for BiLSTM classification

4.4.3 1D-CNN

The model proved to be the most accurate during testing when results showed it had 100% accuracy. This model had to be run twice to see if they have all been a false negative. The model was then given three legitimate emails which proved to be 100% accurate with its detection. The Convolutional neural network proved to be the most accurate model.

The CNN model was the most accurate of any of the models selected when training. It appeared to overfit at the start of the training then the gap narrowed to the point where both trained and test merged, this may have been the reason for 100% detection for when it was given phishing emails to analyse.

GloVe 300d



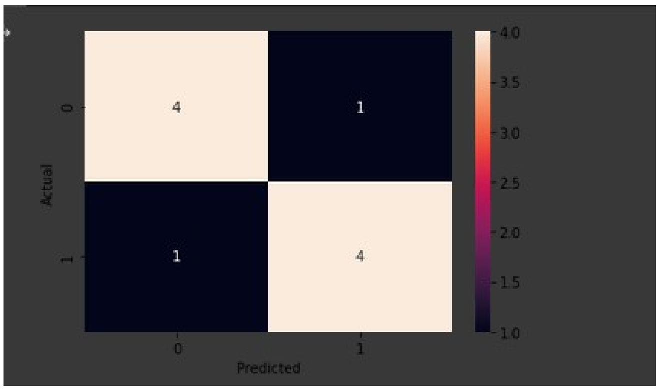
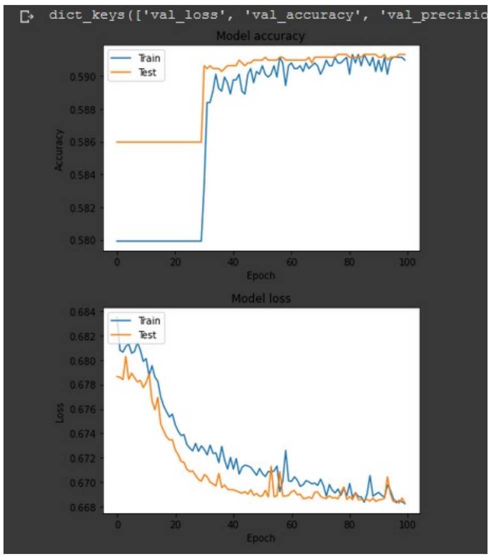


Figure 23 Confusion Matrix for 1D-CNN classification

4.4.5 BNN-CNN (Hybrid)

The hybrid model was accurate when tested against phishing emails. However, the model had the least accuracy of any of the models. It was the most stable with little overfitting during training. Nevertheless, its accuracy was very low, which could be improved. The model had good f-score and performed well during the training process.



```
dict_keys(['val_loss', 'val_accuracy', 'val_precisio
```

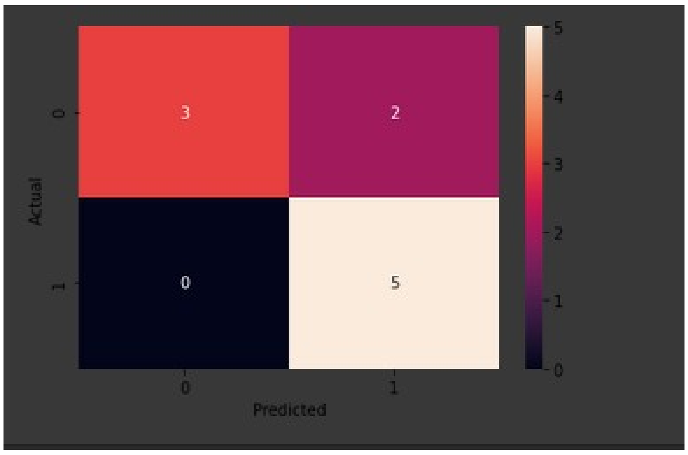


Figure 23 Figure 18 Hybrid Model learning rate with GloVe 300d

Figure 24 Figure 16 Confusion Matrix for Hybrid Model classification

Chapter 5.0 Discussion

This chapter will discuss the overall results of the models and how they performed at detecting phishing emails and training; this will discuss the use of the embedding file and the changes to phishing emails being categorised. And the overall performance of models.

5.2 Network evaluation

Each model had a different accuracy during the testing of the models, this was guaranteed to happen, as each model functions and learns differently from one another, this was also displayed in study (Nhan Cach Dang et al,2020) which concludes RNN models are more accurate than the CNN models used in the study. It also shows different datasets can affect learning rates due to the difference in the data, which may be harder to learn from. The overall accuracy of each model's used varied, this was also concluded in other studies with recurrent neural networks to detect phishing emails. The RNN model had very high accuracy but it was never tested on phishing emails to classify, which seems a huge limitation of the study. The decision to test each of the neural networks on phishing emails was based on the future work of (Lukáš Halgaš et al, (2019) which future work suggested having it classify phishing emails as a result.

When measuring the accuracy of each of the networks, the average accuracy was noted and the f score was calculated, to show the overall accuracy of each of the given networks. the f-score proves to be a more accurate measurement as it takes in the precision-recall of each of the networks to calculate the overall accuracy.

The 1D-CNN performed the best with the f-score with the highest at 0.7, which shows it was most accurate. 1D-CNN showed to have the best performance and percentage of categorising phishing emails. The best network performance was the1D-CNN network as it had the highest f-score and the highest correctly labelled emails. The BLSTM performed the worst from any of the networks as it had the worst f-score of 0.3 and the worst detection rate overall.

Each of the networks had strengths and weaknesses. Overall, the 1D-CNN proved the best at detecting phishing emails based on their plain text. CNN performed the best with the use of a max-pooling layer that allowed the data to be squashed and processed better. That would have helped define features that are common in a phishing email. The accuracy spikes and the overfitting stop during the training,

which hits a perfect fit with the test/train of the network at 100 epochs. The model had the least accuracy during the learning of 60% but had the highest f-score, which shows the network had a better validation and recall showing it was the most accurate at learning overall.

The hybrid network somewhat performed well, with an f-score of 0.6 which performed better score of 0.3 (glove 300 embedding)

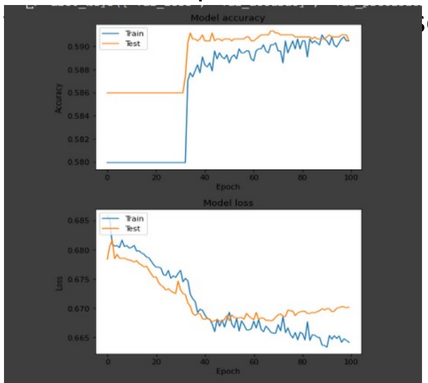


Figure 25 1D-cnn results Glove 200d

5.2.1 File embedding

Each network, when tested, was first used with Glove embedding 300 which was then lowered to the embedding of 100. On average each of the accuracies of the network increased as the embedding file was reduced from 300d to 100d.

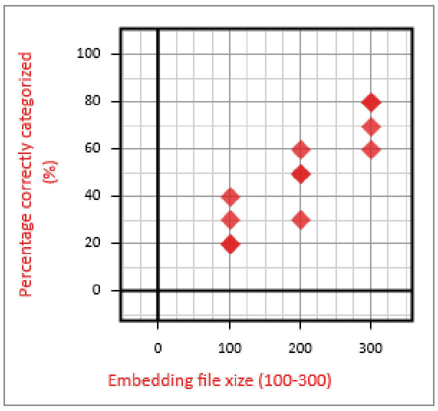


Figure 26 Correct percentage of emails categorised to embedding file size correlation

As shown in figure 23, there is a correlation between the model's prediction and the embedding file being more significant, this may be due to an increased vocabulary available to the networks. That allows the networks to have a clearer understanding and better distinguish 'spam' from 'ham'. The more dimensions of the embedding file indicated that a neural network could make more of an accurate prediction to determine a spam form ham. Which relates back to *tf-idf* which indicates a broader contextual meaning of words in each file for the networks to learn from can help phishing email detection regardless of the overall accuracy of the model.

The larger file has more contextual meaning for each of the given words, and each deep learning network understands the language better, as 300 dimensions is a larger amount of added context.

The use of larger file embeddings allows the models to better learn how words fit into sentences. The wider context of the embedding file increases the accuracy of the deep learning classification of phishing emails. As shown in figure 23 this correlates with each of the model's successful email classification, which increases when the embedding file is increased.

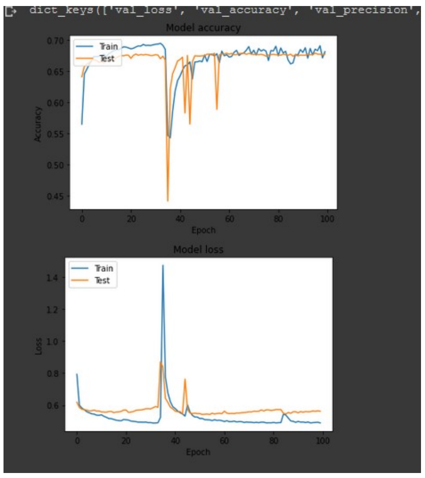
The use of word embedding can make each model more accurate when given emails to categorize and predict. One major part of natural language processing is giving words syntactical meaning, as shown in figure 25 the larger file with more significant, the more accurate it will be at classifying phishing emails.

5.3 Issues found when testing

During the testing of each of the model’s accuracy varied throughout the testing stage when gathering results. Model's accuracy was different from every other model. Some were more successful at categorizing phishing emails as others were not. That was a finding which was mentioned throughout the literature review as every model performs differently. The CNN model proved to be the most accurate during this stage as it did categorize each phishing email correctly.

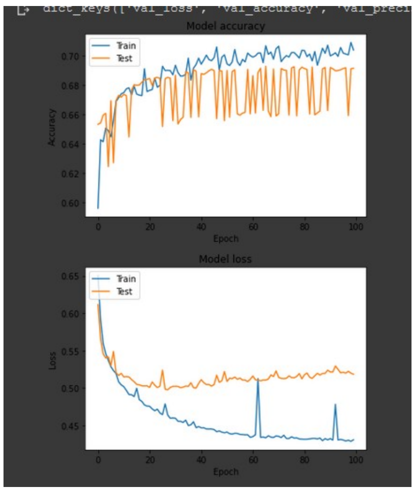
Issues were found when the results were being gathered such as overfitting. This issue can be exemplified with the BLSTM as the training data is incredibly unstable during this stage, which indicates that the model was overfitting during this process. This issue may have been due to the dataset used to train the model struggling to distinguish ‘spam’ and ‘ham’ emails. An attempt to early stop was made where the test and the train were as close as possible as shown in figure 24. That would be around 15 epochs to eliminate this issue. Once tested with phishing emails, the model predicted incorrect phishing emails. Overfitting is a common issue faced when testing machine learning.

Other issues were found, such as Google collab stopping the run time that is likely to be the reason for considerable spikes in the models' learning rates as shown in all figures of the training process. That can be shown with figure 27 as the timeout happened the accuracy dropped massively.



```
dict_keys(['val_loss', 'val_accuracy', 'val_precision',
```

Figure 28 connection time out effect on learning



```
dict_keys(['val_loss', 'val_accuracy', 'val_preci
```

Figure 27 over fitting on dataset (BSLTM 100)

5.4

The purpose of the study was to evaluate and see the effectiveness of different deep learning models used to detect phishing emails based on their plain text properties. Such as commonly used language and grammar. In theory, there are patterns in written texts that are the basis of X hypothesis. Applied with natural language processing creates a reasonable means to detect phishing emails. The literature review shows that phishing emails can bypass web filters through an attack method called a good word attack which the attacker can add or decrease given word count of the spam emails; The attacker can manipulate the word count of a given spam email to trick an email filtering system which it will classify as legitimate.

The review also covered user susceptibility to phishing emails, as different studies showed that most of the users fall for banking emails, telling them to update their banking information. Other emails such as urgency types: “your account has been logged into by X” which the victim will click the link and fall for the phishing email. Other studies on deep learning show that networks can detect sentiment in language and distinguish a positive sentence from a negative sentence. Issues were found when reviewing the Amazon open dataset, as the negative reviews vary in size compared to more abundant positive reviews, this caused smaller positive reviews to be tagged as unfavourable. Other studies show that deep learning can learn from corpus files but make false predictions and miss categorise the data.

The results from comparing the deep learning models show that each varies inaccuracy to the classification of the phishing emails. Which was demonstrated by lowering the embedding file which gave each network a higher overall accuracy but a lower score for predicting phishing emails. Each model had strengths and weaknesses. Some took longer to train than others; some had to be changed in parts to allow the data to merge and learn. Overall a 1-D convolutional network proved to be the most accurate and stable during the learning and best at categorising phishing emails, it had one of the lowest accuracies but overall the model didn't underfit or overfit, it learned at a steady pace and classified all the phishing emails with glove.6B.200d.

Chapter 6.0 Conclusion

The result shows that deep learning can detect phishing email and proves that natural language processing can be applied to different types of datasets such as phishing emails. Other studies compared movie reviews being good or bad and news articles. The language used varies in a phishing email but will have specific linguistic traits that define it as a phishing email.

The convolutional neural network proved to be accurate during the training phase, which can be used for sentiment analysis. CNN's is mainly used for image recognition. The correct classification of phishing emails might be because of the max-pooling layer and the use of the flatten function, which brings the array size into smaller sections that it can learn.

One key point noted is that the embedding size does correlate to the model categorizing phishing emails shows that the larger the word embeddings. The more accurate the deep learning model is going to be at detecting phishing or classifying any text which demonstrate that each model can learn human language and successfully categorizing it with training. The main impact the study shows is that all language and typed text can be categorized and fitted to a label to be trained from. Studies show that humans are poor at detecting phishing emails. The findings show that deep learning models prove to be significantly more effective at detecting phishing emails than a human. Even sophisticated phishing emails which often trick a human were detected.

One key point noted is that the embedding size does correlate to the model categorizing phishing emails shows that the larger the word embeddings the more accurate the deep learning model is going to be at classifying phishing or any text, which demonstrates that each model is capable of learning human language and successfully categorizing it with training. The main impact the study shows is that all language and typed text can be categorized and fitted to a label to be trained. Studies show that humans are poor at detecting phishing emails. The findings show that deep learning models prove to be significantly more effective at detecting phishing emails.

6.1 Future work

The study had limitations. An example would be the dataset. This could be improved with more phishing emails and being made into a multi-label classification, so the supervised learning model would learn what type of phishing email it is, such a 419 email or a fraudulent email. With the appended tag of what the phishing email is, this would be a vast improvement showing deep learning can categorize phishing emails and define what subcategory of email they are, a multi-classification method would be useful as it would show the client what types of attack they are being faced with and this could help them improve their web filter security, and look at methods of the email type of how it bypasses their web filters. The use of Bidirectional Encoder Representations from Transformers (BERT) would vastly improve the accuracy of the model also if this were used as a detection-based system, this could be paired with an API feature which could then be a real-world tool used to detect phishing emails. Also testing other embedding techniques such as Word2vec and different types of Glove embedding such as the twitter corpora, and a gender-neutral embedding file to eliminate it having any sexist connotations or gender bias with labelling language.

References

Alejandra Diaz, Alan T. Sherman, Anupam Josh (2018) Phishing in an academic community: A study ofuser susceptibility and behavior

Cach Dang María N. Moreno García Fernando De La Prieta (2020) Sentiment Analysis Based on Deep Learning: A Comparative Study

colah. (2015). Understanding LSTM Networks. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Last accessed 06/03/2020.

Daniel L ,Christopher M (2005)Good Word Attacks on Statistical Spam Filters

Daniel Lowd. Christopher Meek (2005) Adversarial Learning

HoaT.Le, Christoph ,Cerisara, Alexandre Denis (2019) Do Convolutional Networks need to be Deep for TextClassification ?

http://www.tfidf.com/

Jeffrey Pennington, Richard Socher, Christopher D. Manning(2014) Glove: Global Vectors for Word Representation

Jie Zhou and Wei Xu,(2015) End-to-end learning of semantic role labeling using recurrent neural networks

Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, Kai-Wei Chang(2018) Learning Gender-Neutral Word Embeddings

Jose-marcio Martins Da Cruz , Gordon V. Cormack,(2009) On the relative age of spam and ham training samples for email filtering

Joshua Kim. (2017). Understanding how Convolutional Neural Network (CNN) perform text classification with word embeddings. Available: https://towardsdatascience.com/understanding-how-convolutional-neural-network-cnn-perform-text-classification-with-word-d2ee64b9dd0b. Last accessed 05/03/2020.

Koo Ping Shung. (2018). Accuracy, Precision, Recall or F1?. Available: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9. Last accessed 017/03/2020

Jeffrey Pennington, Richard Socher, Christopher D. Manning . (na). *GloVe: Global Vectors for Word Representation.* Available: https://nlp.stanford.edu/projects/glove/. Last accessed 20/04/2020.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation

Luheng He, Kenton Lee†, Mike Lewis, and Luke Zettlemoyer(2017) Deep Semantic Role Labeling: What Works and What’s Next

Lukáš Halgaš1, Ioannis Agrafiotis1, and Jason R.C. Nurse (2019) Catching the Phish: Detecting Phishing Attacks Using Recurrent Neural Networks

M. Schuster and K. K. Paliwal(1997) Bidirectional recurrent neural networks

Marcus Butavicius,Kathryn Parsons,Malcolm Pattinson,Agata McCormac (2015) Breaching the Human Firewall: Social engineering in Phishing and Spear-Phishing Emails

Matthew R. Gormley .Margaret Mitchell .Benjamin Van. Durme Mark Dredze (2014) Low-Resource Semantic Role Labeling

MillerSmiles.co.uk 419 scams. http://419.millersmiles.co.uk

missinglink. (2017). 7 Types of Neural Network Activation Functions: How to Choose?. Available: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/. Last accessed 28/04/2020.

NA. (2020). Recurrent Neural Networks (RNN) with Keras. Available: https://www.tensorflow.org/guide/keras/rnn. Last accessed 14/03/2020.

na. (2020). Text classification with TensorFlow Hub: Movie reviews. Available: https://www.tensorflow.org/tutorials/keras/text_classification_with_hub. Last accessed 06/03/2020.

NA. (2020). tf.keras.layers.Bidirectional. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Bidirectional. Last accessed 05/03/2020.

Nhan Cach Dang , María N. Moreno-García and Fernando De la Prieta(2020)Sentiment Analysis Based on Deep Learning: A Comparative Study

Nhan Cach Dang , María N. Moreno-García and Fernando De la Prieta (2020) Sentiment Analysis Based on Deep Learning: A Comparative Study

Sebastian Ruder. (2016). An overview of gradient descent optimization algorithms. Available: https://ruder.io/optimizing-gradient-descent/. Last accessed 20/04/2020.

Snehasish Dutta Gupta (2014) Applications And Potentials Of Artificial Neural Networks In Plant Tissue Culture

TA Almeida (2010) Content-Based Spam Filtering

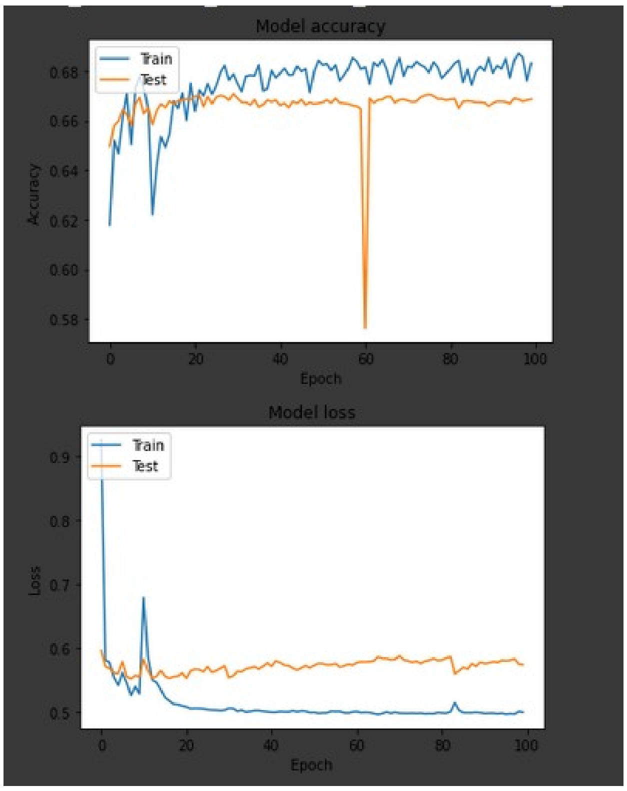
TA Almeida(2010)Probabilistic anti-spam filtering with dimensionality reduction

Tarek Atwan. (2017). How To Plot A Confusion Matrix In Python. Available: https://tatwan.github.io/How-To-Plot-A-Confusion-Matrix-In-Python/. Last accessed 05/03/2020.

Appendices

Appendix A

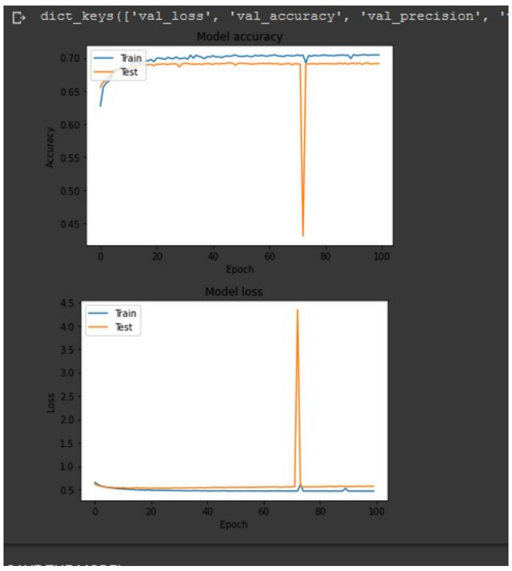
LSTM glove 300d



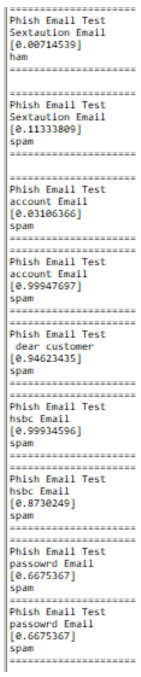
LSTM PREDICTION 300 embedding

```
LSTM_300D
=====================
Phish Email Test
Sextaution Email
[[0.09928547 0.00071454]]
ham
=====================

=====================
Phish Email Test
Are you busy Email
[[0.01866619 09462343]]
spam
=====================

=====================
Phish Email Test
account Email
[[0.09689364 0.00310637]]
ham
=====================
=====================
Phish Email Test
account Email
[[5.230326e-05 9.994769e-02
spam
=====================
=====================
Phish Email Test
 dear customer
[[0.00537656 0.09462343]]
spam
=====================
=====================
Phish Email Test
hsbc Email
[[6.54073e-05 9.99346e-02]]
spam
=====================
=====================
Phish Email Test
hsbc Email
[[0.01269751 0.08730248]]
spam
=====================
=====================
Phish Email Test
passowrd Email
[[0.03324633 0.06675367]]
spam
=====================
[[0.09070623 0.00929377]]
ham
=====================
[[0.00537656 0.09462343]]
spam
=====================
```

BSTM 300



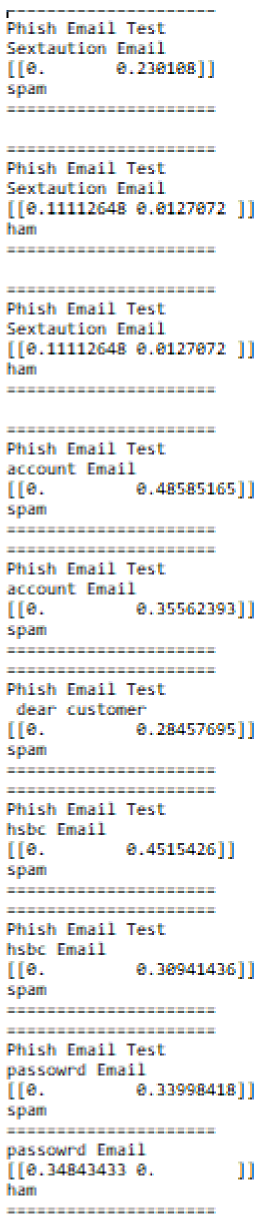
```
dict_keys(['val_loss', 'val_accuracy', 'val_precision', '
```



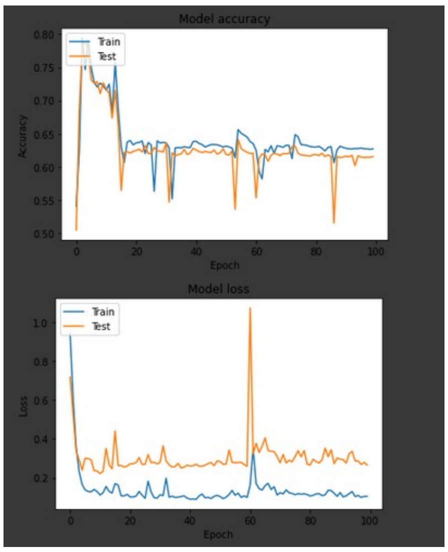
```
======================
Phish Email Test
Sextaution Email
[0.00714539]
ham
======================

======================
Phish Email Test
Sextaution Email
[0.11333809]
spam
======================

======================
Phish Email Test
account Email
[0.03106366]
spam
======================
======================
Phish Email Test
account Email
[0.99947697]
spam
======================
======================
Phish Email Test
 dear customer
[0.94623435]
spam
======================
======================
Phish Email Test
hsbc Email
[0.99934596]
spam
======================
======================
Phish Email Test
hsbc Email
[0.8730249]
spam
======================
======================
Phish Email Test
passowrd Email
[0.6675367]
spam
======================
Phish Email Test
passowrd Email
[0.6675367]
spam
======================
```

BLSTM PREDICT

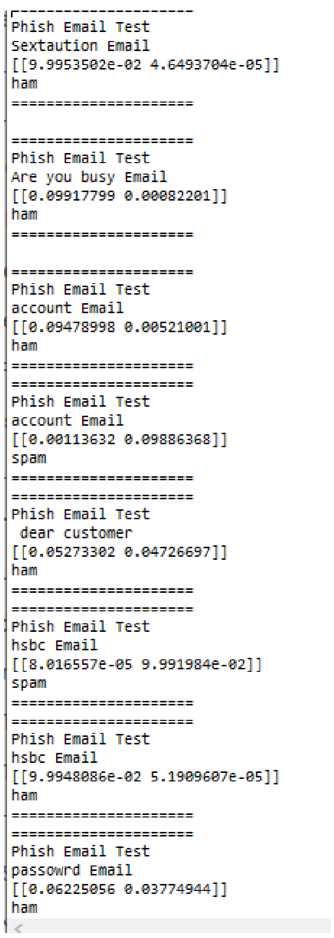
1D-CNN



```
=====================
Phish Email Test
Sextaution Email
[[0.        0.230108]]
spam
=====================

=====================
Phish Email Test
Sextaution Email
[[0.11112648 0.0127072 ]]
ham
=====================

=====================
Phish Email Test
Sextaution Email
[[0.11112648 0.0127072 ]]
ham
=====================

=====================
Phish Email Test
account Email
[[0.         0.48585165]]
spam
=====================
=====================
Phish Email Test
account Email
[[0.         0.35562393]]
spam
=====================
=====================
Phish Email Test
 dear customer
[[0.         0.28457695]]
spam
=====================
=====================
Phish Email Test
hsbc Email
[[0.        0.4515426]]
spam
=====================
=====================
Phish Email Test
hsbc Email
[[0.         0.30941436]]
spam
=====================
=====================
Phish Email Test
passowrd Email
[[0.         0.33998418]]
spam
=====================
passowrd Email
[[0.34843433 0.         ]]
ham
=====================
```



Hybrid model

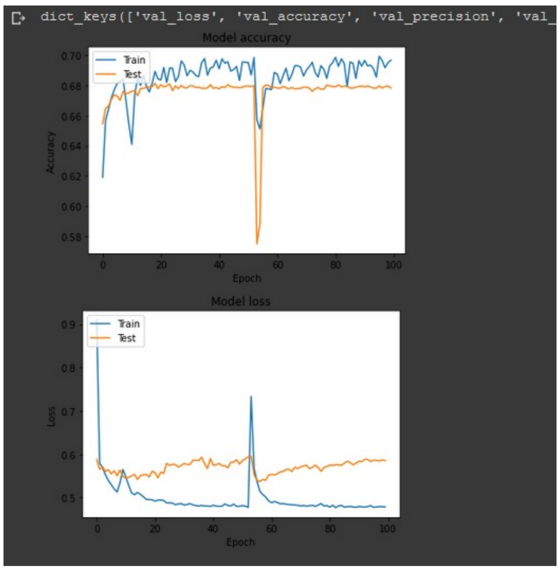


```
=====================
Phish Email Test
Sextaution Email
[[9.9953502e-02 4.6493704e-05]]
ham
=====================

=====================
Phish Email Test
Are you busy Email
[[0.09917799 0.00082201]]
ham
=====================

=====================
Phish Email Test
account Email
[[0.09478998 0.00521001]]
ham
=====================
=====================
Phish Email Test
account Email
[[0.00113632 0.09886368]]
spam
=====================
=====================
Phish Email Test
 dear customer
[[0.05273302 0.04726697]]
ham
=====================
=====================
Phish Email Test
hsbc Email
[[8.016557e-05 9.991984e-02]]
spam
=====================
=====================
Phish Email Test
hsbc Email
[[9.9948086e-02 5.1909607e-05]]
ham
=====================
=====================
Phish Email Test
passowrd Email
[[0.06225056 0.03774944]]
ham
```

Appendix B

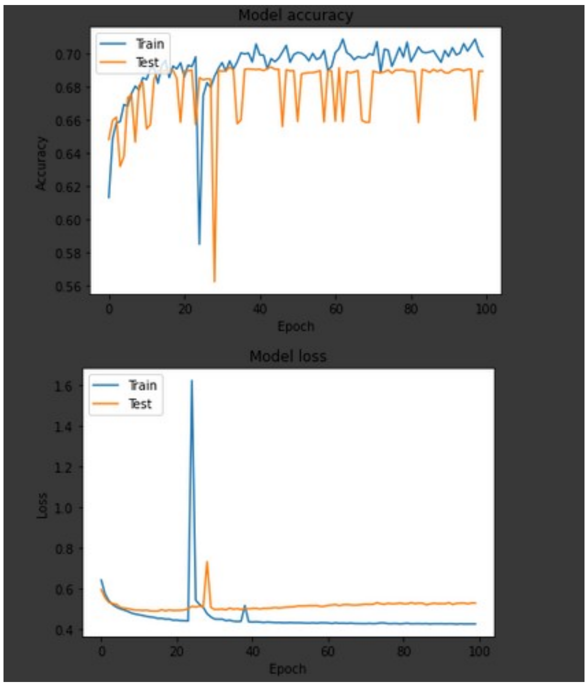
Glove 200 results

LSTM

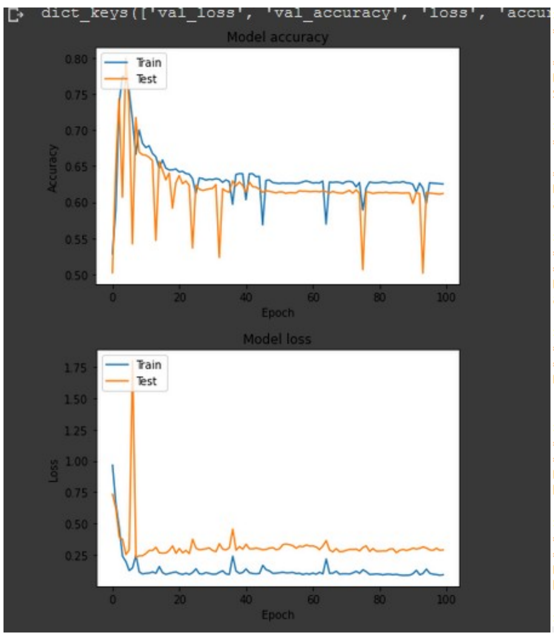


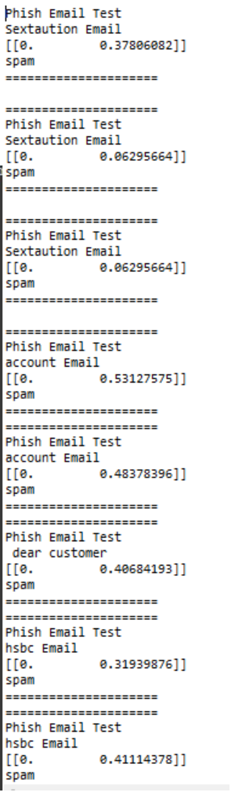
```
dict_keys(['val_loss', 'val_accuracy', 'val_precision', 'val_
=====================
Phish Email Test
Sextaution Email
[0.00046494]
ham
=====================

=====================
Phish Email Test
Sextaution Email
[0.00822012]
ham
=====================

=====================
Phish Email Test
account Email
[0.0521001]
ham
=====================
=====================
Phish Email Test
account Email
[0.9886368]
spam
=====================
=====================
Phish Email Test
 dear customer
[0.47266972]
ham
=====================
=====================
Phish Email Test
hsbc Email
[0.9991984]
spam
=====================
=====================
Phish Email Test
hsbc Email
[0.0005191]
ham
=====================
=====================
Phish Email Test
passowrd Email
[0.37749442]
ham
```

BLSTM



CNN

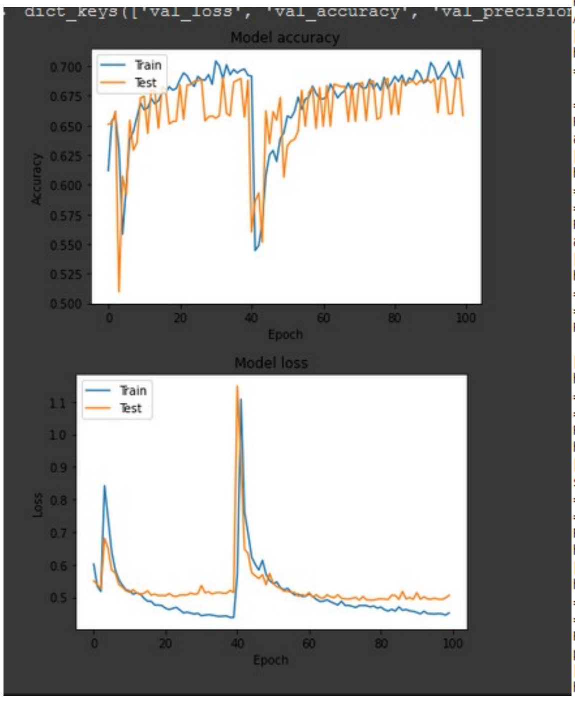

```
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accu
```


```
Phish Email Test
Sextaution Email
[[0.         0.37806082]]
spam
=====================

=====================
Phish Email Test
Sextaution Email
[[0.         0.06295664]]
spam
=====================

=====================
Phish Email Test
Sextaution Email
[[0.         0.06295664]]
spam
=====================

=====================
Phish Email Test
account Email
[[0.         0.53127575]]
spam
=====================
=====================
Phish Email Test
account Email
[[0.         0.48378396]]
spam
=====================
=====================
Phish Email Test
 dear customer
[[0.         0.40684193]]
spam
=====================
=====================
Phish Email Test
hsbc Email
[[0.         0.31939876]]
spam
=====================
=====================
Phish Email Test
hsbc Email
[[0.         0.41114378]]
spam
```

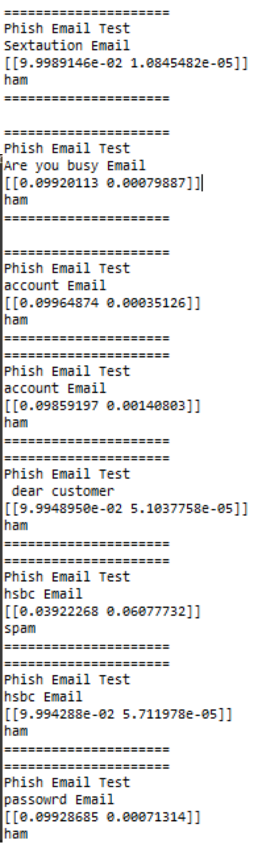
Appendix C

Glove 100 results

LSTM

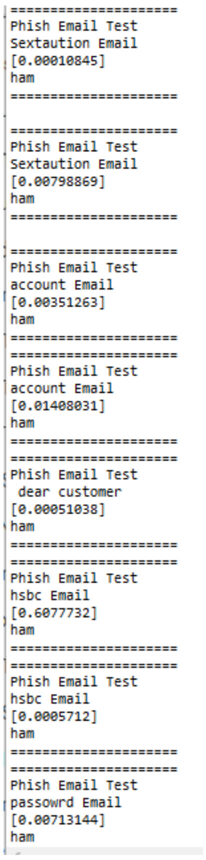


```
dict_keys(['val_loss', 'val_accuracy', 'val_precisio
```

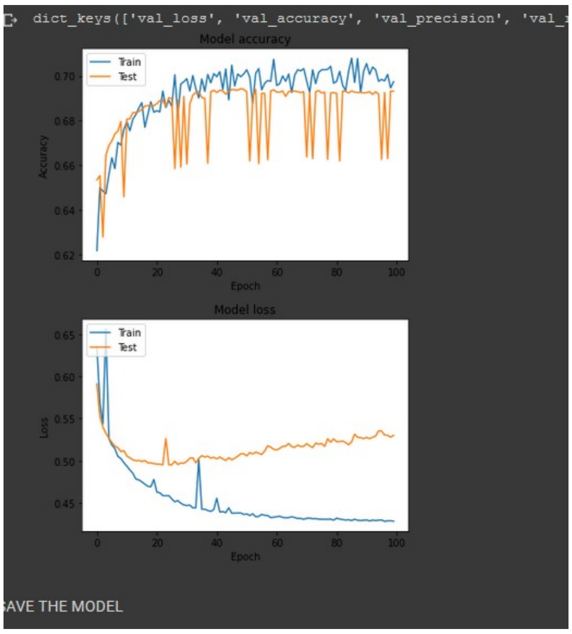


```
====================
Phish Email Test
Sextaution Email
[[9.9989146e-02 1.0845482e-05]]
ham
====================

====================
Phish Email Test
Are you busy Email
[[0.09920113 0.00079887]]
ham
====================

====================
Phish Email Test
account Email
[[0.09964874 0.00035126]]
ham
====================
====================
Phish Email Test
account Email
[[0.09859197 0.00140803]]
ham
====================
====================
Phish Email Test
 dear customer
[[9.9948950e-02 5.1037758e-05]]
ham
====================
====================
Phish Email Test
hsbc Email
[[0.03922268 0.06077732]]
spam
====================
====================
Phish Email Test
hsbc Email
[[9.994288e-02 5.711978e-05]]
ham
====================
====================
Phish Email Test
passowrd Email
[[0.09928685 0.00071314]]
ham
```

BLSTM

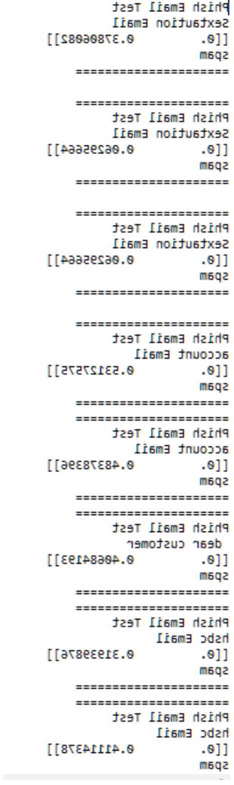


```
=====================
Phish Email Test
Sextaution Email
[0.00010845]
ham
=====================

=====================
Phish Email Test
Sextaution Email
[0.00798869]
ham
=====================

=====================
Phish Email Test
account Email
[0.00351263]
ham
=====================
=====================
Phish Email Test
account Email
[0.01408031]
ham
=====================
=====================
Phish Email Test
 dear customer
[0.00051038]
ham
=====================
=====================
Phish Email Test
hsbc Email
[0.6077732]
ham
=====================
=====================
Phish Email Test
hsbc Email
[0.0005712]
ham
=====================
=====================
Phish Email Test
passowrd Email
[0.00713144]
ham
```



```
dict_keys(['val_loss', 'val_accuracy', 'val_precision', 'val_
```

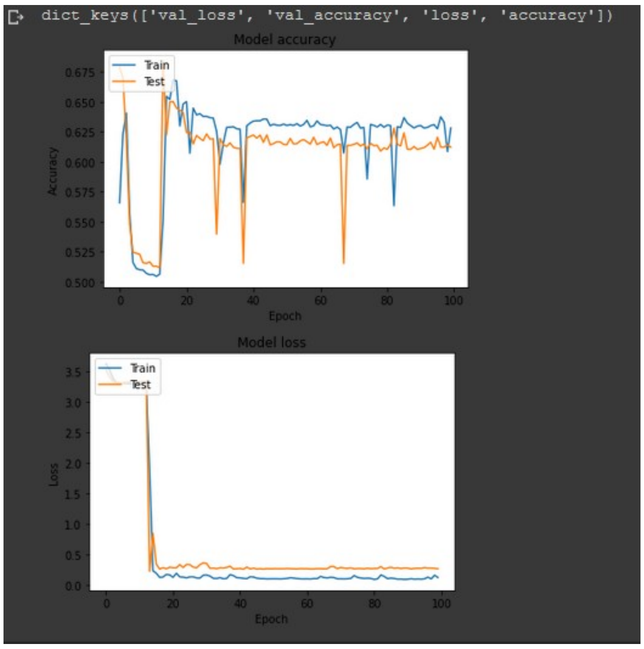
AVE THE MODEL



```
Phish Email Test
Sextaution Email
[[0.          0.37806082]]
spam
=====================

=====================
Phish Email Test
Sextaution Email
[[0.          0.06295664]]
spam
=====================

=====================
Phish Email Test
Sextaution Email
[[0.          0.06295664]]
spam
=====================

=====================
Phish Email Test
account Email
[[0.          0.53127575]]
spam
=====================
=====================
Phish Email Test
account Email
[[0.          0.48378396]]
spam
=====================
=====================
Phish Email Test
dear customer
[[0.          0.40684193]]
spam
=====================
=====================
Phish Email Test
hsbc Email
[[0.          0.31938876]]
spam
=====================
=====================
Phish Email Test
hsbc Email
[[0.          0.41114378]]
spam
```

Hybrid

1D- cnn



```
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```