Swinburne University of Technology

Faculty of Science, Engineering and Technology

ASSIGNMENT AND PROJECT COVER SHEET

Unit Code: **COS30015** Unit Title: **IT Security**

Assignment number and title: **2 - Practical Project** Due date: **30 Oct 2022**

Lab group:_____ Tutor:_____ Lecturer: **Andrew Plapp**

Family name: **Anggawijaya** Identity no: **103507067**

Other names:_____

To be completed if this is an INDIVIDUAL ASSIGNMENT

I declare that this assignment is my individual work. I have not worked collaboratively, nor have I copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part been written for me by another person.

Signature: **David**

To be completed if this is a GROUP ASSIGNMENT

We declare that this is a group assignment and that no part of this submission has been copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part been written for us by another person.

ID Number	Name	Signature
_____	_____	_____
_____	_____	_____

Marker's comments:

Total Mark:_____

Extension certification:

This assignment has been given an extension and is now due on _____

Signature of Convener:_____ Date: / 2022

COS30015 IT Security

Assignment 2 - Practical Project
Attacker or Malware Analysis - Source Analysis and Detection model for Email Spams

Author: David Anggawijaya (103507067)
Submitted on: 20 October 2022
Due on: 30 October 2022

Planning and Justification

According to statistic, in 2022 there are 4.3 billions global users of emails and expected to grow to 4.6 billions users in 2026 [4]. Because of high users, email becomes a very good place for performing marketing strategy by sending advertisement to large number of customers that already sign-up for regular news letters. However, spammer take this opportunity to send unwanted or irrelevant contents to large number of random email users who didn't sign up for it repetitively to achieve certain goal [1].

From spam purpose, email spam can be categorised into two type; commercial purpose, and fraud purpose. Email spams for commercial purpose are still used by business corporation due to its low cost for marketing and can be produce in bulk to increase customer reaches [1]. However, email spams can also be used for fraud and stealing personal information. Originally the email message is not harmful, but the contents can be misleading and contains malicious links or attachments which can resulted in further IT security issues. Attacker may use fake website link in the email to perform phishing attack to steal username and password from user [2]. Attacker can also use Email to send virus or malware to a certain target, they do this by constructing email with malicious link that will force victims to automatically download malware inside their computer [3]. Malware can also be put inside email as file attachment, once a victim download and open the attachment the malware will take over their computer.

In 2021, it was estimated that 45% of emails sent are categorised as spams. Among which around 2.5% of it are used for phishing, scams, and fraud [5]. This shows how important is to trace and analyst email before consuming its contents. There are many tools or procedure that IT people used to analyst the email to gain informations such as sender IP address, sender's location, and main purpose of the email. However, doing this manually for each individual email will consume a lot of time and not effective. That is why now days major SMTP server comes with spams filtering feature. This feature work by using machine learning to analyst each individual email and have the machine categorised it into either spam or legitimate email. Despite having accurate spam filters, some spam emails may still be able to pass the SMTP detection. Due to that reasons, most SMTP protocols like Gmail allow their user to manually mark their email as a spam, the labeled email will then be feed to the machine learning to prevent similar email to appear in user's inbox.

In this report, I am going to break down tools and procedures that can be used to track down the purpose and source of an email. I will also analyst a spam detection model from both attacker and defender perspective. To do this, I will create a spam detection model using help from several Python libraries such as pandas for csv file reading and table visualisation, nltk to clean each email text, keras to transform large text into smaller meaningful data, and tensorflow for machine learning tool. I am going to use database from kaggle.com to train my model and analyst its performance by trying to predict my own spam emails that I take from Gmail spam box.

Machine learning classification algorithm such as Naive Bayes has becoming a famous approach in handling spam detection due to its fast implementation and high accuracy result. Machine leaning algorithm is limited to a single train session for the same dataset [6]. While, Deep learning allow the same dataset to be train multiple times to ensure high accuracy and effectiveness of the model. Because of that most of smart AI nowadays have been switching from using machine learning into deep learning. Thus I decided to focused more on using deep learning algorithm for making spam detection model in this report.

Deep learning algorithm is an algorithm that works by mimicking the structure and function of human brain but with computer processor power [7]. Structured of deep learning is separate into three kind of layers: input layer is where dataset is feed into, hidden layer containing weights parameters to process the input data, and output layer to present the output prediction [8]. There are three common types of deep leaning; Artificial Neural Network (ANN), Convolution Neural Network (CNN), and Recurrent Neural Network (RNN). ANN is the most common neural network algorithm that uses feed forward concept where data is passed through hidden layers once per train session, ANN is suitable for tabular data [9]. CNN is neural network that is used to processed and learn image dataset [9]. Lastly, RNN is algorithm that considered the dependency of each input data against one another, RNN is suitable for predicting text data because text is made of multiple words that dependant on one another [9]. Because in this report I am going to deal with email dataset which made of text data, RNN becoming the most suitable deep learning algorithm.

Application and Documentation

Part A - Email's Source and Purpose Analysis

For this report, Gmail service will be considered. Gmail service comes with spam filter model that automatically classify any incoming email as spam or legitimate. However, a very small percentage of emails may be classified wrongly. Some spam email may be able to pass the spam filter or valid email being thrown to the spam junk. Due to that reason analysing the source of the email can help to check the purpose of the email and deduce whether Gmail fail to classify this email. Reporting miss classify email will helping Gmail algorithm to better perform in the future run.

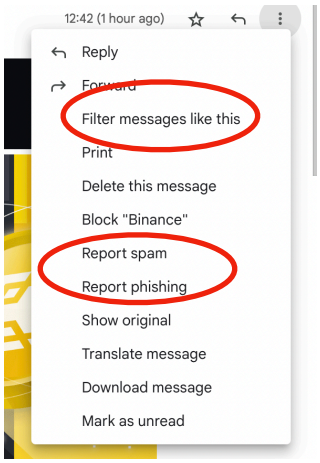

Figure 1. Ways of helping Gmail Algorithm

Any email that marked as spam by Gmail spam filter will be located into spam box. Unlike inbox, spam box usually hidden in Gmail so user must firstly find the spam box location in order to see which emails are mark as spam.

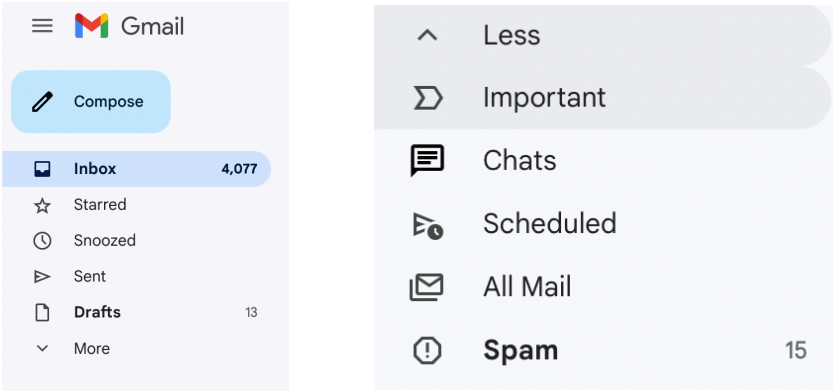

Figure 2. Looking for Spam box

For this report, I am going to take example of two spam emails from my Gmail's spam box.

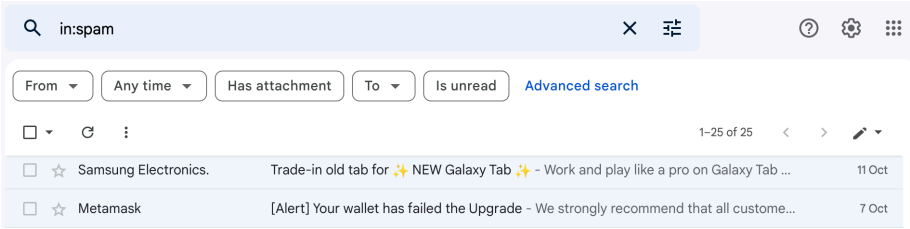

Figure 3. Example Spam Emails

Before I break down procedures that can be follow to trace email source and purpose. It is better to firstly understand how Gmail algorithm classify spam. Gmail filter looks at variety of information such as email's Ip address, domains, bulk sender identity, and user feedback [10]. Looking at how Gmail encourage user intervention in their algorithm shows the importance of knowing how to analyst email source.

First, to analyst the source of an email it is better to check the sender's identity. Senders identity includes IP address, location, and Domain. Once IP address of the sender found, looking for their location and the domain is pretty straight forward. IP address of an email can be found on email original version. Figure 4 shows how to see the original version of the email. Original version of email consist of 'Received' header that shows the sender IP address. Finally, the IP address can be used together with geolocation service to track the sender location.

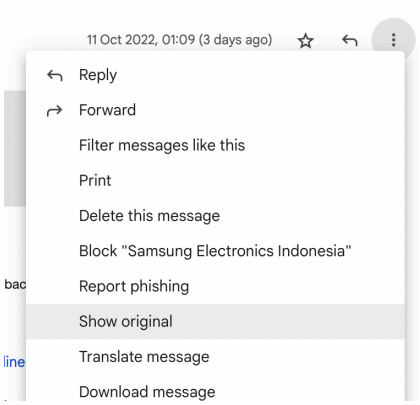

Figure 4. Show Original Email

These whole process of getting IP address of an email is very complicated and consume time. I decided to make it autonomous using Python script and Gmail API. First is to connect Python with Gmail API and request for the spam emails. Once collected, the format will be JSON using indexes we can get the 'Received' header. Then the header is reformat using regular expression to only return the IP address. Finally the Python geolocation library is used to automatically search the location of each IP address. Figure 6 shows the table result from my Python script.

```
path = 'emails' # Email files location
listing = os.listdir(path)
dict = {
"Email Subject" : [],
    "Sender IP Address" : [],
    "Location" : []
}

def get_location(ip_address):
    ip = geocoder.ip(ip_address)
    return "{}, {}".format(ip.city, ip.country)

for fdl in listing:
    p = path + "/" + fdl
    with open(p, 'rb') as fhdl:
        raw_email = fhdl.read() # open the file
    ep = eml_parser.EmlParser()
    parsed_eml = ep.decode_email_bytes(raw_email) # read the original email version
    dict["Email Subject"].append(parsed_eml["header"]["subject"])
    x = parsed_eml["header"]["received"][1]["src"] # get the IP address from header
    ip = re.findall(r'(?<=\[)\d{1,3}(?:\.\d{1,3}){3}(?=\])', x)
    dict["Sender IP Address"].append(ip) # save the IP address
    dict["Location"].append(get_location(ip[0]))
```

Figure 5. Python Script IP tracker

	Email Subject	Sender IP Address	Location
0	Trade-in old tab for ✨ NEW Galaxy Tab ✨	[69.169.235.237]	Seattle, US
1	[Alert] Your wallet has failed the Upgrade	[161.38.200.30]	Los Angeles, US

Figure 6. IP address results

I already get the IP address for the two email spam examples used in this report. These IP addresses can be reverse lookup into their domain names using MXToolBox service on internet.

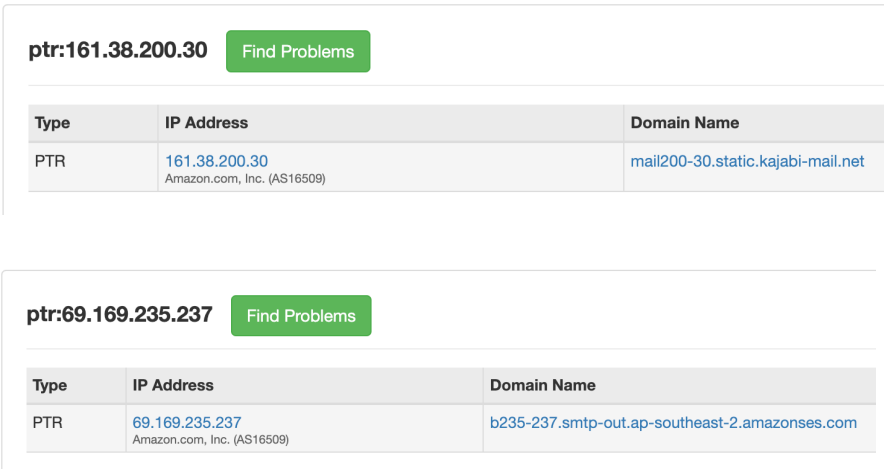

Figure 7. Domain names results

After getting the domain names, we can use WhoIs web service to look up more information about the domain names. The first IP address lead to domain name called “amazonses.com”, when I put this domain name on WhoIs service the response shows that this domain name is under Amazon technology company. Then by looking more information from official Amazon web page and I found that “amazonses.com" is a third party email service that provide bulk sending email for company.

The second IP address lead to domain name “kajabi-mail.net”, using WhoIs service I get information shown in figure 8. Figure 8 shows that the site is no long active, this may be due to a report in previous time. This suggest that the email likely to be a spam.

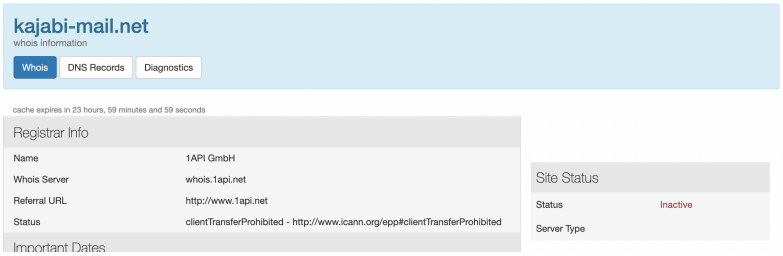

Figure 8. Domain names Details

To further check the legitimate of an email, the contents must be analyst to see the purpose of the emails. There are several steps that can be performed to analyst the email’s content. First, if the purpose of the email is for business purpose such as updates or advertisements, the email should have option for clients to unsubscribed. Second any redirect links in the email should be coming form trusted domains and using secure HTTPS protocol to prevent any phishing attempts.

Figure 9 shows the content of first spam email, the content suggest that the email is sent by Samsung company regarding product advertisement. Because the email is about product advertisement, the email contain unsubscribe button. When I click the button, I got redirected to their website for unsubscribe process. To check if the unsubscribe process is valid, I inspect the network request when clicking unsubscribe button. Assuming the unsubscribe process is valid the network should be able to track a request packet. Figure 10 shows that the unsubscribe option is really valid.

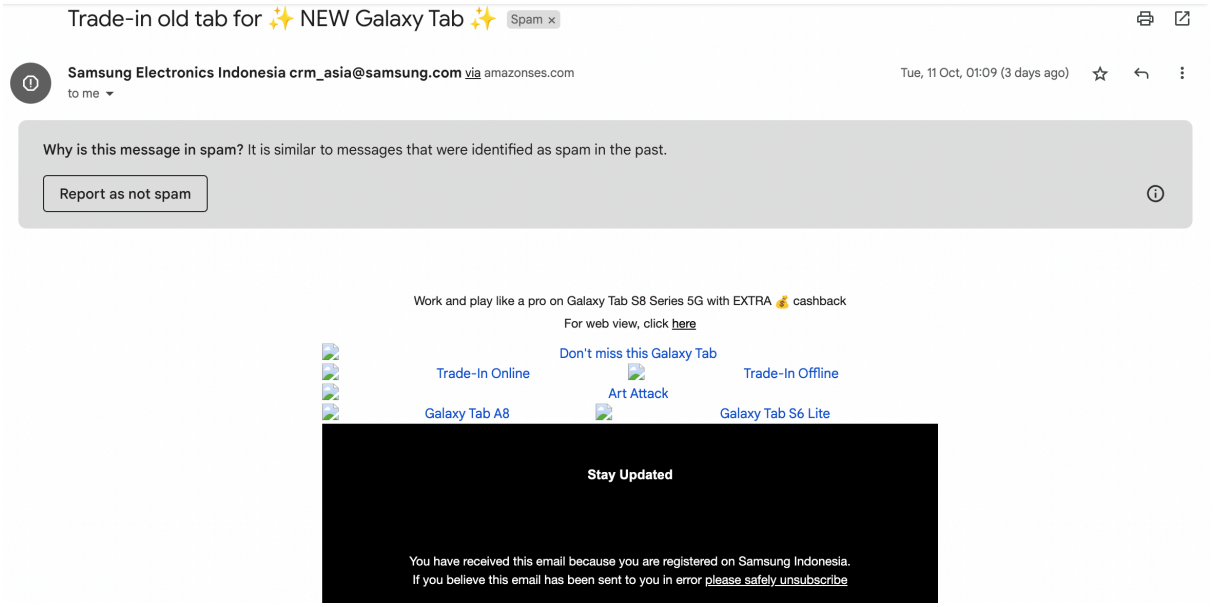

Figure 9. Email Example 1 contents

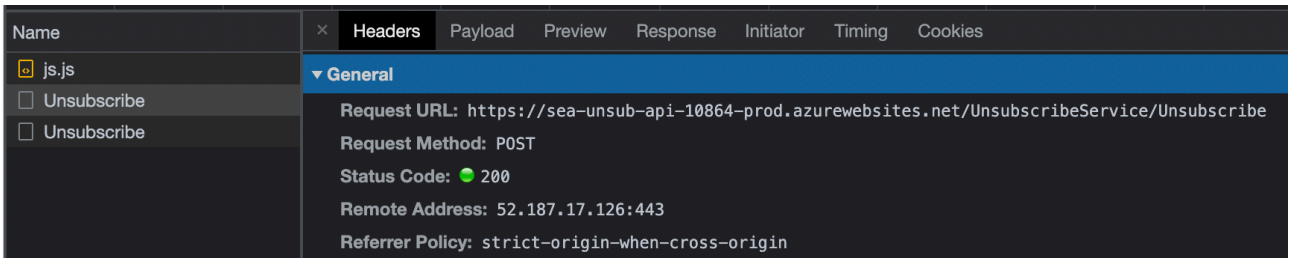

Figure 10. Network Inspect

Despite being sent from legitimate source with safe contents the email is still mark as spam by Gmail. According to Gmail official page, Gmail has a very strict guidelines for sending bulk emails using third-party email service providers. This is why Gmail can't guarantee messages sent by third-party email providers will pass their spam filters [11]. The first email example is marked as spam likely because of using third party email services ‘Amazonses.com’.

Figure 11 shows that the second spam email is pretending to be Metamask asking for me to update my Metamask settings. The previous source analysis shows that the IP address is not coming from official Metamask domain. Thus, the email is likely to be a fraud. First, the email has unsubscribe option but doesn’t redirect user to anywhere. Second, the update wallet button lead user to unsecured website link which shown in figure 12. Especially knowing that the purpose of this email is to update user’s private information, the HTTP protocol provides vulnerability for attacker to steal wallet password. Both source and contents of this email shows that the purpose of this email is for phishing.

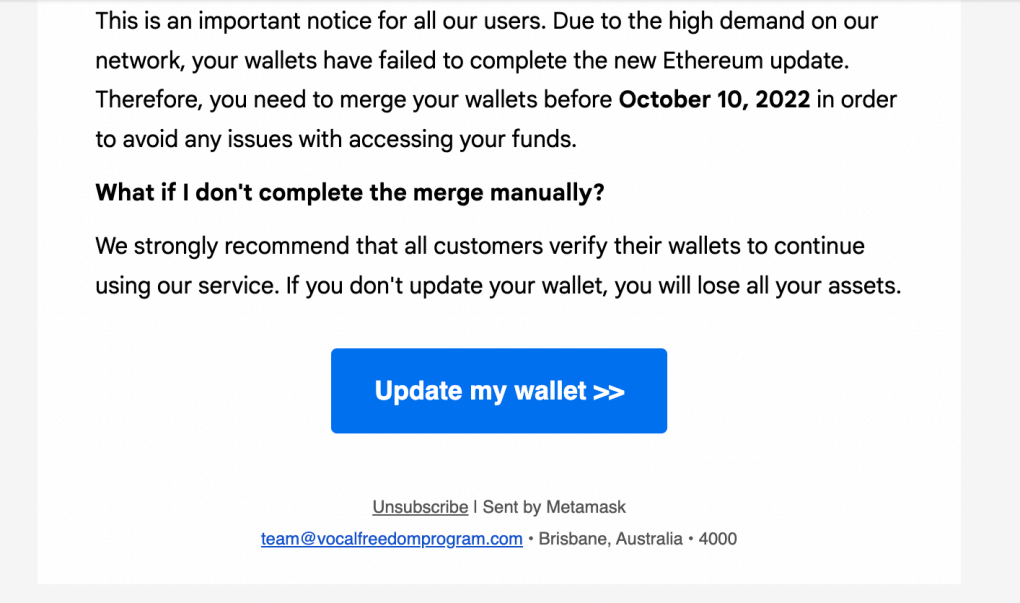

Figure 11. Email Example 2 Contents

Figure 12. Link Inspection

Part B - Spam Detection Model with Python

For the purpose of this research, I am going to use Jupiter lab a web based interactive coding platform. Before I code the detection model, I firstly need to find sustainable database. I use email database from kaggle.com, the database consist of two columns; text and spam classification. Each email in the database have already been classified as either spam or ham. Spam means that the email is marked as spam email, while ham classified email as non spam. Because the deep learning model can only output result between 0 to 1, I used MS Excel to transform spam to 1 and ham to 0.

Before jumping straight to coding, I need to install all Python libraries needed to start the task. First is Pandas library, this library is used to read CSV file and perform table visualisation creation. Next in the line is Numpy, this library allow complex array processing. Since email contents are made of multiple non ASCII characters that are affecting the model accuracy, the text must be clean to only leave meaningful words. To process the email text, I used additional four libraries: nltk, ssl, string, and re. These libraries will work together to remove punctuations, special charset characters, preposition, un-meaningful words, and some emojis.

I started by importing the CSV database into the Python environment using Pandas library. Figure 13, shows the first 5 rows of the dataset that was successfully inserted into Python. Before processing each email, I first clean the dataset by dropping any duplicates rows from the dataset.

```
# email spam dataset
df = pd.read_csv("spam_ham_dataset.csv")
df.drop(['Unnamed: 0', 'label'], axis=1, inplace=True)
df = df.rename(columns = {"label_num":"spam"})
df.head()
```

	text	spam
0	Subject: enron methanol ; meter # : 988291\r\n...	0
1	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	Subject: photoshop , windows , office . cheap ...	1
4	Subject: re : indian springs\r\nthis deal is t...	0

Figure 13. Read CSV dataset

```
# remove duplicate
df.drop_duplicates(inplace = True)
```

Figure 14. Remove Duplicates

```
# clean each raw data lower case, no punctuation, no stop words
def clean_text(text):
    emoj = re.compile("["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
        u"\U00002500-\U00002BEF"  # chinese char
        u"\U00002702-\U000027B0"
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u"\U00010000-\U0010ffff"
        u"\u2640-\u2642"
        u"\u2600-\u2B55"
        u"\u200d"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\ufe0f"  # dingbats
        u"\u3030"
                      "]+", re.UNICODE)
    text = re.sub(emoj, '', text)
    nopunc = [char for char in text if char not in string.punctuation]
    nopunc = ''.join(nopunc)
    clean_words = [word for word in nopunc.split() if word.lower() not in stopwords.words("english")]
    return " ".join(clean_words)

df["text"] = df["text"].map(clean_text)
```

Figure 15. Text Processing Function

Once the dataset cleaned, I proceed to processing each email text so that our deep learning machine can understand the dataset. Figure 15, shows the process of transforming each email text.
The function will take each email text and loop through each character in the text. Any character that resembles emoji's charset, special characters, symbols, or punctuations will be deleted from the text. After that I use nltk.stopwords collection to compare each word in the text, any word that exist in the stopwords collection will be deleted from the text. Stopwords collection consist of 40 useless words that do not describe the main ideas of a text such as 'in', 'or', 'at', and etc [12].

	text	spam
0	Subject: enron methanol ; meter # : 988291\r\n...	0
1	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	Subject: photoshop , windows , office . cheap ...	1
4	Subject: re : indian springs\r\nthis deal is t...	0

	text	spam
0	Subject enron methanol meter 988291 follow not...	0
1	Subject hpl nom january 9 2001 see attached fi...	0
2	Subject neon retreat ho ho ho around wonderful...	0
3	Subject photoshop windows office cheap main tr...	1
4	Subject indian springs deal book teco pvr reve...	0

Figure 16. Before and After Text processing

Since deep learning model works by training on dataset and validating each training accuracy against test dataset, I use sklearn library to split the original dataset into 80% training and 20% testing dataset.

```
from sklearn.model_selection import train_test_split
emails_train, emails_test, target_train, target_test = train_test_split(df["text"],df["spam"],test_size = 0.2, random_state= 0)
```

Figure 17. Splitting dataset

```
from keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words=max_feature)

tokenizer.fit_on_texts(emails_train)
```

Figure 18. Text Tokenizing

Even after cleaning and processing the dataset, the current dataset are not capable to classify email as spam or not spam. This happens because we haven't considered the basis idea behind how email detection model work. Email detection model works by analysing the structure of spam email and its dependency against some frequently used words. For a deep learning machine to able to considered how each unique word in the email affects the classification result, the whole email text must be seperate into smaller chunks of meaningful words, this process is called text tokenizing. Text tokenizing produce an array of word called token. Figure 18, shows the whole process in tokenizing each email text. I start by importing keras library because the library provide feature to tokenizing a text. Then, I create a tokenizer object which has fit_on_texts function that will automatically split each email in the dataset into smaller chunks. The next step is to transform each token into an array of integer that represent the number of times each unique words in the token occur in the whole dataset. For example, assuming a dataset consist of two tokens [a, a , b, b], [a, b, c] during the transformation the first token will transform into [3, 3] this is because the first token consist of only 2 unique words: word 'a' and 'b' both occur 3 times in the whole dataset. This process is repeated for each token in the dataset for both training and testing dataset. Figure 19, shows the process of transforming each token and example of output transformation for the first token in the training dataset.

```
emails_train_tokens = np.array(tokenizer.texts_to_sequences(emails_train))
emails_test_tokens = np.array(tokenizer.texts_to_sequences(emails_test))
print(emails_train_tokens[0])
```

```
[2, 1256, 4334, 1546, 118, 2596, 10141, 10142, 1294, 1146, 62, 20043, 1238, 396, 381, 135, 569, 517, 4833, 1295, 13157, 133, 118, 2917, 4
11, 510, 57, 425, 133, 4833, 1295, 252, 2597, 50, 3158, 269, 1321, 293, 76, 792, 188, 547, 2262, 20044, 20045, 20046, 20047, 20048, 1315
8, 8480, 10143, 20049, 10144, 20050, 20051, 13159, 10145, 13160, 6050, 13161]
```

Figure 19. Tokens Transformation

I now start creating the RNN model, to create this model two additional libraries are needed. Keras and Tensorflow libraries will help in creating layers for the model. Since RNN is a sequential neural network, we will use keras.Sequential() object and assign it to variable called model. To add layers to the model, we call .add() method and specify what type of layer we want to add. To prevent complexity of determining which layer to put, I took a sample code of RNN model from Github. So all of the initial parameters have been preconfigured by following the GitHub. Figure 20, shows the code for the RNN model.

```
import tensorflow as tf
embedding_vecor_length = 32

model = tf.keras.Sequential()
model.add(Embedding(max_feature, embedding_vecor_length, input_length=max_len))
model.add(Bidirectional(tf.keras.layers.LSTM(64)))
model.add(Dense(16, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 20. RNN model

To train the model, we can call .fit() method on the model variable. The .fit method will ask for dataset to be trained in this case it is the training dataset. The method will also asked for number of epochs, epochs stands for number of training sessions will be performed. Generally higher numbers of epochs generate better accuracy for the model. Thus, for this fit() method I am going to use 20 epochs. Lastly, fit() method will asked for validation dataset. Validation dataset is dataset that will be used to check the accuracy of the model on each training session. Based on the accuracy of the model in predicting validation dataset the hidden layers will automatically be modified by the model on each training session. I am going to use testing dataset for this parameter. Figure 21, shows the model.fit() method's parameters. The model's performance against train and test datasets on each training epoch are recorded in Figure 22.

```
history = model.fit(emails_train_tokens, target_train, batch_size=512, epochs=20, validation_data=(emails_test_tokens, target_test))
```

Figure 21. Training the RNN

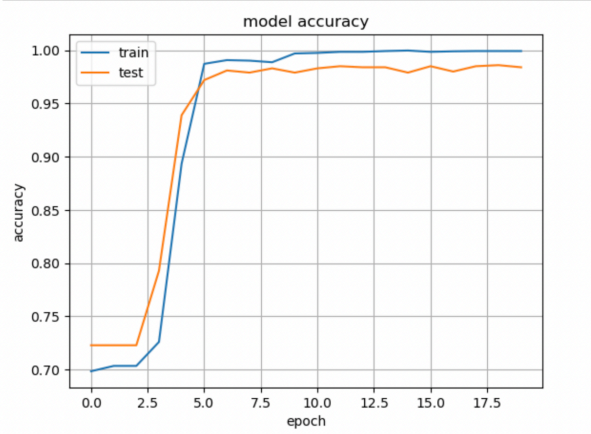

Figure 22. Model Accuracy on each Epoch

After training the model, I use the model to predicts the spam classification for each row in the testing dataset. Because the output from RNN is ranging from 0 to 1, I create a conditional where if the output is less than 0.5 it will be change into 0 and vice versa. Figure 23, shows the model predicting the testing dataset. Finally, I use sklearn library to calculate the accuracy and confusion matrix of the prediction result.

```
spam_predict  = [1 if o>0.5 else 0 for o in model.predict(emails_test_tokens)]
```

Figure 23. Model Predicting Test Data

To further see the performance of the spam detection model, I want the model to try predict a different dataset. This time the dataset is taken from my real Gmail spam box. I wanted to see if the model can predict that all of the emails in the dataset is a spam email which shown in Figure 24. The same sklearn library will then be used to calculate the accuracy and confusion matrix of the results.

	text	spam
0	Kesempatan terakhir untuk 44 David Jangan lewa...	1
1	David Shire look grand BIG Enjoy beautiful vie...	1
2	best September Sale coming Lift creativity new...	1
3	Pinocchio related candlesticks Plus tips spice...	1
4	Upgrade dapet GRATIS David Miliki Samsung TV f...	1
5	0739870 Bitcoins available Thanks filling 0739...	1
6	Withdrawal error AMOUNT USD need add withdrawa...	1
7	makeover Rp 49Mio David Redecorate home chic L...	1
8	Flash Sale Galaxy terbaru menanti Rasakan perf...	1
9	David exceptional cinema Dont wait longer best...	1
10	Promo Galaxy TERBARU cuma 3 hari Flash Sale Ja...	1
11	cant keep Switch Galaxy Z Flip4 5G greater fle...	1
12	Iris Like photo notification 7245475513 httpsd...	1
13	Dont wait September ends 1 WEEK LEFT September...	1
14	Last chance Rp 49Mio decor David Samsung got s...	1
15	Teman BARU yang Pasti Awesome Sambut Galaxy A2...	1
16	Find perfect quiz David Match right Lifestyle ...	1
17	stylish yet David Rp 69Mio Lifestyle TV Get jo...	1
18	lowest price Final Day Give best tool fight di...	1
19	Celebrate Best Moments David Let Samsung compa...	1
20	Need camera flex like Flex Galaxy Z Flip4 Z Fo...	1
21	profile going deleted id 41488676132 httpsdriv...	1

Figure 24. Model Predicting Real Emails

```
prediction  = [1 if o>0.5 else 0 for o in model.predict(real_emails_tokens)]
```

Figure 25. Model Predicting Real Emails

Analysis

After the detection model has been implemented and trained using cleaned training dataset, the model can be used to predict testing dataset. The performance of the model is recorded using Sklearn library. Figure 26, shows the confusion matrix report of the model in predicting testing dataset. The details show that the columns represent predicted labels, while the rows represent the actual labels. Out of 722 actual non spam emails, 716 of the rows were successfully predicted by the model and a type I error on 6 emails predicted as spam. Meanwhile, from a total of 277 spam emails, 267 emails was correctly predicted as a spam

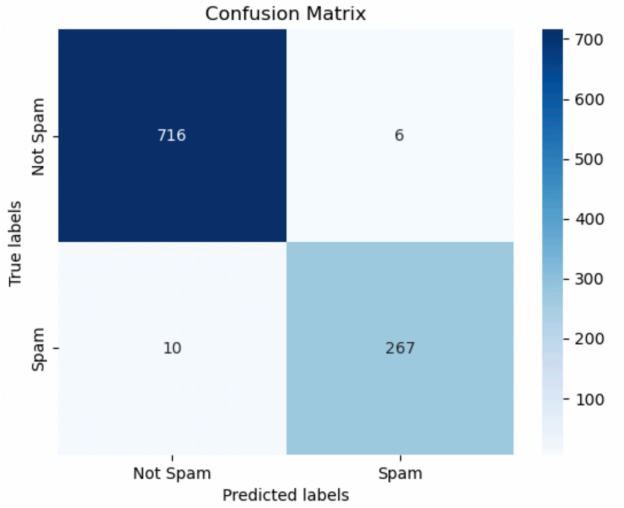

Figure 26. Confusion Matrix Predict Test dataset

and a type II error on 10 emails predicted as not spam. The performance accuracy of the model in predicting testing dataset is 98.39%, which is very good.

```
Accuracy:  0.983983983983984
```

Figure 27. Model Prediction Accuracy on Test dataset

Accuracy of the deep learning model is affected by the size of training dataset, and number of epochs assign during training. Figure 22 above, shows that in each epoch the accuracy of the model is increasing. It shows that the accuracy reach its peak performance after 5 epochs and no longer increasing, this is a crucial information that can help reducing the cost of time for training the model. Knowing that the model will achieve peak performance around 5 epochs, the next training can be perform with only 5 epochs to reduce consumption of the time.

To further analyst the performance of the model, the model is faced with new spam dataset that taken from real life Gmail Spam box. The model is expected to be able to predict that the new dataset are all spam emails. However, the same model fail to predict the new emails as spams. The accuracy performance of the model fall into 69.56 %, which is not very good. Out of 23 spam emails, 7 of the emails fails to be classified as a spam. The problem suggest that the model is overfitting. Overfitting occurs when a deep learning model performs really well against the training dataset that it fail to predict a new dataset [13]. One possible reason why the model produce bad accuracy, is due to the dataset shift problem. Dataset shift occurs when the distribution of the data change over a period of time [14]. The dataset that is used in the experiment is generated from few years ago, while the new dataset may no longer follow the same pattern due to socioeconomic factors such as spammers or attacker behaviour [14].

Once trained, the model create a word banks consisting of frequently used spam words. The moment the model has to predict an email, the model will check each word in the email and compare it with the spam words bank. If the email contains sustainable amount of spam words, it will not be able to bypass the spam filters. Knowing such defence mechanism exist, attacker or spammer will try to find the vulnerability in the model and try to exploit it to bypass the filters. One vulnerability in our model that can be exploit by attacker is the defence mechanism that only check for the used of spam words in the email. This way attacker can simply bypass the spam detection by generating phishing email with no spam words. Nowadays on internet there are a lot of websites that show guidelines and list of words to avoid in order to bypass spam filters security.

Increasing the size of the dataset and making the machine learn continuously can help to prevent the model vulnerability. However, this decision also produce another security threat for email's users. Because the model will consider the new spam words used by the attacker to bypass the security, slowly the model will used all of words available and produce higher type I error. Meaning, more legitimate emails will be considered as spam despite coming from trusted sources. Email's users are tend to ignore any emails that get into their spam box. Thus, many important conversations will be missed.

To prevent attacker bypassing the spam filter security without increasing type I error. From security professional perspective, they must start to considerate another parameters that can be used to classify email spam other than frequently used spam words. Source and purpose analysis from above experimentation provides procedures that can effectively deduce if the email legitimate or not. The procedures can help identify other factors such as the sender IP address, domain, embedding links, and attachments. However, the limitation of the process is that it consume a lot of

time and too complex to be implemented. Thus, finding a way to make a detection model that also considered email's IP address, domain, links, and attachments in addition with used of words becoming the next possible solution.

Evaluation

In conclusion, the number of users in SMTP servers will keep on increasing each year. This create a large opportunity for spammer to send bulk emails for business or fraud purpose. Over 90% of security breach is caused by workers unknowingly open phishing emails [15]. Email's phishing is one of easiest phishing method that attacker can performed because attacker can easily generate one fraud email and send it in bulk to list of email addresses. Phishing emails are usually combine together with spam activity to increase the probability of target recipient to open it. As the result, over 70% of phishing emails is open by the target [15].

As discussed in the documentation above, tools such as Python, MxToolBox, and WhoIs websites can be used to perform source and purpose analysis on an email. The objective of source and purpose analysis is to identify the credibility of the email's sender and purpose of the contents. Thus, the difference between a legitimate email and fraud email can be identified. Despite being useful, the whole process is complicated and inefficient for large emails data. Most people that are not tech savvy can't fully trace back the email to its source.

Due to that reason, most SMTP servers invented their own email spam filters engine. There are may ways of implementing spam detection model, the most common method used is machine learning. However, lately a more powerful replacement has been considered. By mimicking a human brain, deep learning allows a model to train multiple times without needing human intervention. There are many types of deep learning algorithms, but the algorithm that is suitable in processing text data is called RNN model. RNN model is implemented for spam detection engine in this report. Despite of high performance during training and testing, the model still fail to predict real life dataset. This shows that the performance of model that we implemented are far in comparison with major SMTP's engine. For an example Gmail, their engine considered variety of signals then just the used of words in the email. In determining which email marked as spam, Gmail considered the sender IP address, domain, third party service, links, and users feedback [10]

The analysis above, shows that there are two major challenges found in our spam detection model: the current model is affected by dataset shift problem and the presence of attacker in exploits the model's vulnerability. In order to solve those challenges some modification have to be done with the current RNN model. Currently, the model only classified email by analysing the used of spam words in the email. Because the model is limited to only one kind of dataset, the model is prone to dataset shift problem. To further improve the model, it must inherit the concept of source and purpose analysis procedures that have been discussed above. Using programming to collect details about the email's source and embedding links in the email, allow the model to considered more parameters than just a selection of words.

References

[1] Cisco. 2022. *What Is Spam Email?*. [online]. Cisco.com. Available at: <https://www.cisco.com/c/en/us/products/security/email-security/what-is-spam.html> [Accessed 10 October 2022].

[2] Office of the Victorian Information Commissioner. 2022. *Phishing Attacks and How to Protect Against Them*. [online]. OVIC.com. Available at: <https://ovic.vic.gov.au/privacy/resources-for-organisations/phishing-attacks-and-how-to-protect-against-them/> [Accessed 10 October 2022].

[3] Awati, R., 2021. *Email Virus*. [online]. TechTarget. Available at: <https://www.techtarget.com/searchsecurity/definition/email-virus> [Accessed 10 October 2022].

[4] Statista. 2022. *Number of e-mail users worldwide from 2017 to 2025*. [online]. Statista. Available at : <https://www.statista.com/statistics/255080/number-of-e-mail-users-worldwide/> [Accessed 11 October 2022].

[5] Moorthy, J., 2022. *23 Email Spam Statistics to Know in 2022*. [online]. Mailmodo. Available at: <https://www.mailmodo.com/guides/email-spam-statistics/> [Accessed 11 October 2022].

[6] Brownlee, J., 2020. *Why Do I Get Different Results Each Time in Machine Learning?*. [online]. Available at: <https://machinelearningmastery.com/different-results-each-time-in-machine-learning/> [Accessed 10 October 2022].

[7] Brownlee, J., 2019. *What is Deep Learning?*. [online]. Available at: <https://machinelearningmastery.com/what-is-deep-learning/> [Accessed 12 October 2022].

[8] Dettmers, T., 2015. *Deep Learning in a Nutshell: Core Concepts*. [online]. nvidia.com. Available at: <https://developer.nvidia.com/blog/deep-learning-nutshell-core-concepts/> [Accessed 13 October 2022].

[9] Pai, A., 2020. *CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning*. [online]. Analyticsvidhya.com. Available at: <https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/> [Accessed 14 October 2022].

[10] Kumaran, N., 2022. *Understanding Gmail's spam filters*. [online]. Google Cloud. Available at: <https://cloud.google.com/blog/products/workspace/an-overview-of-gmails-spam-filters> [Accessed 14 October 2022].

[11] Gmail. 2022. *Prevent mail to Gmail users from being blocked or sent to spam*. [online]. Gmail Help. Available at: <https://support.google.com/mail/answer/81126?hl=en> [Accessed 15 October 2022].

[12] Python Tutorials. 2022. *NLTK stop words*. [online]. Available at: <https://pythonspot.com/nltk-stop-words/> [Accessed 15 October 2022].

[13] IBM Cloud Education. 2021. *Overfitting*. [online]. IBM. Available at: <https://www.ibm.com/cloud/learn/overfitting> [Accessed 17 October 2022].

[14] Ayuya, C., 2020. *Correcting Dataset Shift in Machine Learning*. [online]. Section.com. Available at: <https://www.section.io/engineering-education/correcting-data-shift/> [Accessed 17 October 2022].

[15] Cveticanin, N., 2022. *Phishing Statistics & How to Avoid Taking the Bait*. [online]. Dataprot.com. Available at: <https://dataprot.net/statistics/phishing-statistics/> [Accessed 17 October 2022].