Computers, Materials & Continua
DOI: 10.32604/cmc.2023.032373
Article

Social Engineering Attack Classifications on Social Media Using Deep Learning

Yichiet Aun[1,*], Ming-Lee Gan[1], Nur Haliza Binti Abdul Wahab[2] and Goh Hock Guan[1]

[1]Faculty of Information and Communication Technology, Universiti Tunku Abdul Rahman, Perak, Malaysia
[2]Faculty of Engineering, Universiti Teknologi Malaysia, Johor, Malaysia
*Corresponding Author: Yichiet Aun. Email: aunyc@utar.edu.my
Received: 16 May 2022; Accepted: 13 September 2022

Abstract: In defense-in-depth, humans have always been the weakest link in cybersecurity. However, unlike common threats, social engineering poses vulnerabilities not directly quantifiable in penetration testing. Most skilled social engineers trick users into giving up information voluntarily through attacks like phishing and adware. Social Engineering (SE) in social media is structurally similar to regular posts but contains malicious intrinsic meaning within the sentence semantic. In this paper, a novel SE model is trained using a Recurrent Neural Network Long Short Term Memory (RNN-LSTM) to identify well-disguised SE threats in social media posts. We use a custom dataset crawled from hundreds of corporate and personal Facebook posts. First, the *social engineering attack detection pipeline* (SEAD) is designed to filter out social posts with malicious intents using domain heuristics. Next, each social media post is tokenized into sentences and then analyzed with a sentiment analyzer before being labelled as *an anomaly* or *normal* training data. Then, we train an RNN-LSTM model to detect five types of social engineering attacks that potentially contain signs of information gathering. The experimental result showed that the *Social Engineering Attack* (SEA) model achieves 0.84 in classification precision and 0.81 in recall compared to the ground truth labeled by network experts. The experimental results showed that the semantics and linguistics similarities are an effective indicator for early detection of SEA.

Keywords: Social engineering attack; cybersecurity; machine learning (ML); artificial neural network (ANN); random forest classifier; decision tree (DT) classifier

1 Introduction

Facebook has become the first highest user in social media applications which garners billions of site visits daily. In addition, the use of social media skyrocketed during the pandemic for online businesses and to stay in touch during physical restrictions. The spike in social media use motivates many intruders to exploit the security vulnerabilities to retrieve user information [1]. Since social

media is people-oriented, 'breaking' into the system involves exploiting human factors using social engineering. One common avenue is by posing as friends or bots in chat boxes to gather lucrative personal information [2]. In addition, any intruder skilled enough can send a message without permission posing as the website's administrator to anyone around the world. For example, intruders can pose as banks and send users spam messages to get their bank accounts or passwords. Besides that, intruders nowadays can easily monitor human user activities carried out on social websites using simple application programming interfaces (API) calls. SE attacks often start with reconnaissance. The attacker spends a long time studying user behaviours, like their favourite products and habits, before launching scathing attacks which sounds convincing to the victims [3].

Despite many studies, there is little understanding in identifying SE attacks due to the subjectivity of social media posts. In addition, it is challenging for a machine to recognize sentiments and read between the lines of social posts that may be a SE threat. Common network security appliances like firewalls only detect illegitimate traffic at flow levels. Meanwhile, intrusion detection system/intrusion prevention system (IDS/IPS) needs to be heavily customized with intelligent rules to read application data [4]. In recent years, there were many works focusing to train deep learning models for threat intelligence using network properties, but SE attacks are mostly textual data that requires natural language processing (NLP) integration [5].

In this paper, we train a RNN-LSTM model to identify variants of SE attacks in social media posts. Firstly, we crawl for social media posts on Facebook instead of using the existing Social Computing Data Repository, Stanford Large Network Dataset Collection (SNAP), and Network Repository because the datasets are from older services and tweets which are insufficient to provide contexts. Next, we design a *social engineering detection pipeline* (SEAD) for data pre-processing. In SEAD, each post is labelled with a sentiment score based on a set of criteria like keyword matching, source screening and social graph to look for posts that show intention of information gathering. Then, we lemmatize and tokenize these malicious social posts and train the RNN-LSTM to model the spatial and temporal linguistic structure of the sentences for SE attack classifications.

2 Human Factor & Social Engineering Landscape

Social engineering prey on human weakness, and every attack is different based on the specific behavioural characteristics of the victims. For example, users are more inclined to act impulsively out of the excitement of victory, fear of authority and fear of loss. Symantec Security Response reported that technical flaws and software exploitation methods drive only 3% of security threats. Meanwhile, the other 97% of attempts are social engineering-driven. These red flags mean additional fortification of data security, application security on-device security and network security only safeguards a small minority of attacks, regardless of the operating system and types of machines [6].

In 2019, HBGary neglected the content management system (CMS) bug that leads to unauthorized shell access due to wrong secure socket shell (SSH) configurations, which otherwise is a relatively standard and straightforward security protocol. The problem stems from carelessness, a common human factor due to fatigue or lack of experience. After conducting forensics, it was discovered that the network administrator ignored the principles and policies governing safe SSH connection and did not secure the authorized_keys used in SSH authentication, resulting in information leak [7].

Attackers designed SE to be pervasive and well disguised [8]. Unfortunately, there is no magic bullet to combat SE like an antivirus to detect viruses. For example, as the name implies, baiting attacks use a false promise to spark a victim's avarice or curiosity. Then, the consumers are tricked into falling into a trap in which their personal information is stolen, or their computers are infected

with malware. Meanwhile, scareware haunts the infected hosts with false alerts and bogus threats. Users are duped into believing their system is infected with malware, prompting them to install software that has no purpose (other than to benefit the attacker) or is a malware. Deception software, rogue scanning software, and fraudware are all terms used to describe scareware. In a pretexting attack, a perpetrator may start the scam by professing to need sensitive information from a victim to complete an essential assignment [9]. The attacker frequently begins by impersonating co-workers, police, bank or tax officials, or other people with right-to-know authority to gain trust from their victim. The pretext poses questions used to validate the victim's identification and obtain sensitive personal information. Phishing scams, which are email and text message campaigns aiming to instil a sense of urgency, curiosity, or terror in victims, are among the most common social engineering attack types [10]. It then pressures people into disclosing personal information, visiting fraudulent websites, or opening malware-infected attachments. A more focused variation of the phishing scam is spear phishing, in which the perpetrator targets specific people or businesses. They then personalize their messages based on their victims' traits, work titles, and contacts to make their attack less obvious.

Machine learning (ML) has been engineered into networking intelligence like software-defined networking SDN and Cisco digital network architecture (DNA) for intrusion detection and traffic filtering [7]. However, there is no existing work to our best knowledge that uses ML to classify SE threats due to the subjectivity of these entities [11]. Collecting datasets for SE is also less straightforward due to privacy concerns and the types of data involved. In our case, social media posts, mainly texts, are in different languages rather than a packet datagram unit. Taif University of Saudi Arabia built datasets containing 300 websites and divided them into two classes: (a) phishing websites and (b) regular websites. Data collection must be continuous (not in stages) because human behaviours and human factors are constantly changing on social media to model temporal behaviours [12]. Data analysis requires NLP that includes dependency parsing [13] to examine words of a sentence from a chat or a post on social media, in order to extract relations of the conversation and flag the possible occurrence of an SE attack.

3 Related Work in Threat Intelligence

The artificial neural network (ANN) has been widely used in computer networking for threat detection [14]. In [15], the authors implemented the constant removal and recursive feature elimination, feature selection techniques for injection/intrusion attack detection in IoT applications. While the study in [16] identified as much as 111 features such as uniform resource locator (URL), domain, file name and parameters, extracted from websites to propose an ANN based classification model for the detection of phishing websites.

Staudemeyer [17] proposes using time-series characteristics of known malicious behaviour and network traffic to improve the classification accuracy of network threats and attacks detection algorithms. They design a four-memory blocks (neural) network, each of which contains two cells. Their experimental results indicate that the proposed LSTM model is better than previously published methods since LSTM could learn to backtrack and correlate continuous connection records in a time-varying manner. Similarly, we can look at the sequence of individual words in sentences as timestep, thus making them feasible to be trained on LSTM.

Meanwhile, Krishnan et al. [18] designed an RNN for Intrusion Detection. Of course, there are more straightforward methods for such threats intelligence, like simple network management protocol (SNMP) monitoring and simpler ML variants like a random forest (RF) classifier. However, the

proposed RNN classify similar threats more accurately and train faster despite working with large datasets.

Similarly, Yin et al. [19] trained another RNN-LSTM for IDS and benchmarked against support vector machine (SVM), ANN, and RF using the NSL-KDD dataset and found improved accuracy. Kim et al. [20] later found that such improvement gain is possible because LSTM solves the long-term dependency problem and overcomes the vanishing gradient drop during training of network data. Le et al. [21] built an LSTM classifier to detect intrusion to optimize the network with hyperparameter tuning and found that the Nadam optimizer is the most effective one. Still, we argue these results can be dataset-specific. In 2019, Zhang et al. [22] directly integrated the improved leNet-5 and LSTM neural network structure to model network threats' spatial and temporal cues. Cybersecurity researchers are long inspired by deep learning for threat intelligence, but these models do not detect social engineering attacks stripped of network parameters. Instead, SEA disguised as social media posts is profiled by semantic sentences, a cross between network and NLP domain.

4 Methodology-Social Engineering Attack Detection (SEAD)

The social engineering attack detection (SEAD) is designed to determine the malicious *factor* of social media posts (see Fig. 1). We define *malicious* as showing sign of information gathering, pretexting, accusative and imperative. Firstly, social media posts are crawled using Spyder from public posts of random users and demographics from personal Facebook accounts. Then, an **entity recogniser** is developed to extract three main entities from text-based posts: (i) perpetuator, (ii) target victim and (iii) the attack target. The entitiy recognition runs on natural language toolkit (NLTK) and SpaCy framework to classify texts into predefined categories like people, location, organisation, everyday objects (digital), device types and actions. For example, in a Facebook post, "Public bank **customer service** has detected a change in the **password** for your user account **Alice**". The social media posts are decoded into three important entities into the tuplet as {subject: customer service; victim: Alice; and target: passwords}.

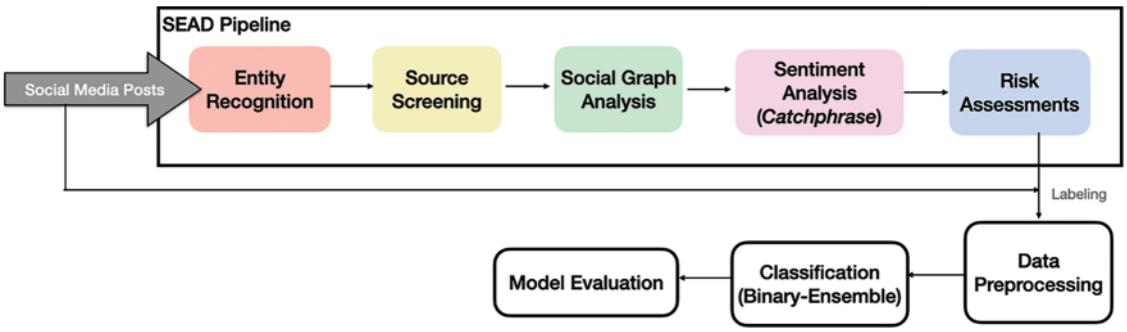

Figure 1: The social engineering attack classifications pipeline

4.1 Source Screening

The Social Engineering Attack Detector (SEAD) operates on *a guilty until proven innocence* basis. SEAD is configured to blacklist SEA threats at onset, using tools like Tesseract for subject screening and filtering. The **Source Screening** is similar to packet filtering in stateful firewalls. The *subjects*/*perpetrator* identified prior is vetted using known IP addresses and user accounts of rogue users and bots housed in a growing blacklist database. However, unlike firewalls, source screening

scans for adversaries at the application layer to catch compromised accounts with malicious intents. This provides more leeway and insights to detect a malicious source rather than relying only on network headers like IP addresses that can easily be spoofed. In addition, source screening understands conversation directionality; for example, a suspicious source is less likely to raise red flags if a legit user had started the conversion earlier and the source replied to an earlier message (see Fig. 2).

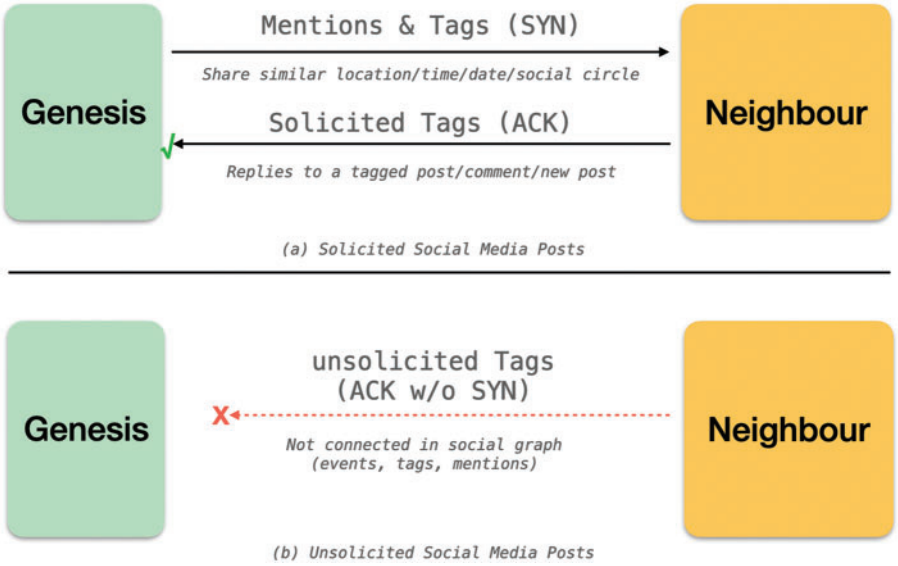

Figure 2: Determining the malicious index of social media post based on interaction state. In (a), a legit post must be soliciting some earlier interactions. In (b), a post with similar semantics is red flagged if there are no prior interactions among the circles

4.2 Social Graph Analysis

Social graph is a graph that represents social relations between people and entities [23]. The social graph analysis detect suspicious source based on a global mapping of everybody and how they are related. SEAD leverages on Facebook Social Graph through HTTP REST API. The social graph classified entities into nodes, edges and fields. Nodes are everything from user, a photo, a comment, a page on Facebook. Meanwhile, edges defines the relationships between these nodes; for example: *likes* and *shares*. SEAD emphasis on the source credibility rather than the message itself to detect for SE attacks; meaning that two similar posts can be flagged as legit and malicious depending on the social graph analysis results. Consider an example: Alice visited location {A, B, C, D}; Bob visited location {B, D, E} and Charlie visited location {X, Y, Z}. Suppose that Charlie and Bob both posted on Alice wall asking for a photo of her. Using social graph, SEAD reasons that Bob post is not suspicious due to their location co-occurrences (close acquaintances) while Charlie post is a SE attack since Alice and Charlie has never met. SEAD also evaluates nodes-to-nodes distance; such that adjacent nodes (two close friends) is less likely to invoke a SE attack compared to nodes that are few hops away. Fig. 3 shows how SEAD determine the sensitivity of red flag depending on inter-nodes relationships.

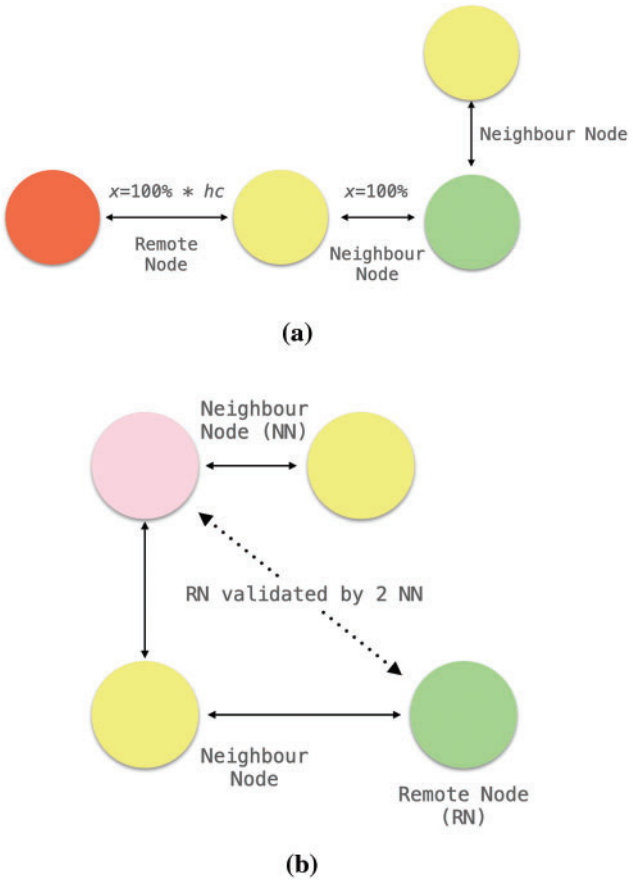

Figure 3: (a) Nodes that are closer to each other has higher trustworthy rate. Each extra hop away from source node are penalized by the hop count value. (b) A remote node (green) is considered trustworthy when verified by two mutual nodes to the source host

4.3 Sentiment Analysis

Sentiment analysis is a commonly used form of measure. We use Google AutoML to build a sentiment analyser to detect SE attacks on social media post. The main reason we need to build a custom sentiment analysis model is that existing pre-trained sentiment express "positive" using positive adjectives and "negative" based on negative adjectives [24]. For example, for a sentence 'borrowing your account for emergency', although it should; will not be flagged as SE attack using current sentiment model. SEAD use a custom Name-based entity recognition (NER) to train for a set of keywords that has malicious motivations. At the heart of NER there are two steps. First, the NER starts to detect word(s) that form an entity. Inside-outside-beginning tagging is a common way to indicate the onset and offset of an entity [25]. Secondly, NER then categorise the detected entities into important categories like person, organisation, location and in our case, actions that spruce malicious intents. NER then analyze sentences morphology to determine the tonal properties of sentence such as *imperative* and *accusative* (see Fig. 4).

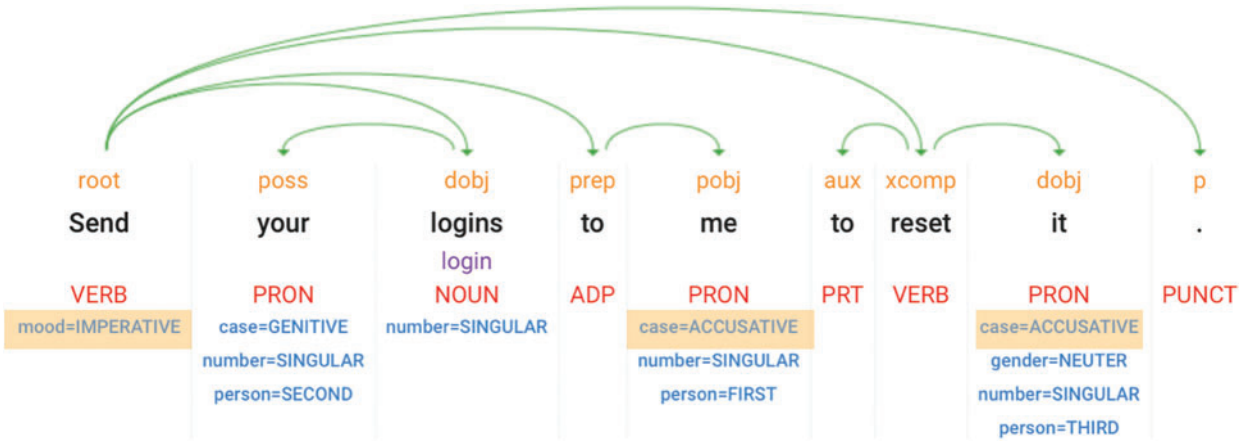

Figure 4: Analysing sentence semantics and morphology in social media posts

Each post (an instance) in the dataset is labelled as '0', '1' by two experts based on morphology terminology. Label '0' indicates a social post that is neutral; like 'good weather to hang out', while label '1' indicate a possible SE attack; like 'borrow your account'.

Table 1 shows a snippet of the dataset. The training set, train, validate, and test columns are in the left column. The training set is used to develop the model in AutoML. While searching for patterns in the training data, the model tries a variety of algorithms and parameters. The validation set is used by the model to test algorithms and patterns as it detects patterns. By default, AutoML Natural Language divides training data into three sets at random: 80% for training, 10% for validation, and 10% for testing. If you leave a column empty, AutoML Natural Language divides the document into three sets automatically.

Table 1: Some examples of annotated social media posts to determine if a post is 'benign or 'malicious'. The sentences are lemetized using NLTK

TRAIN	Hey I hope you still remembers me, can you borrow me your account I want to share you something	1
TRAIN	I saw your pictures posted elsewhere I suggest you send me your password so that I can reset it for you	1
TRAIN	Today has been great, we should do this more often	0
TRAIN	Click on the link and share this post for good luck	0
TEST	Can you share your account password so that I can be the second factor authenticator	1
VALIDATION	Today hang out was great, share me some of the photos we had earlier	0

4.4 Risk Analysis and Data Labeling

In *Risk Analysis*, SEAD averages the threat factor across three detection components to determine the 'integrity' of SEA in each social media posts [26]. Each individual component is first rated from scale of 0 to 1 based on heuristics. All three components are equally weighted. In *source screening*,

each instance of social media post is labelled *true* (1) if the integrity is compromised; like posted by an account or containing mentions of accounts with poor standing; vice versa. Meanwhile using *social graph*, the post is labelled *true* (1) to indicate an indirect post that includes posts that are not a response to a previous interactions or mentions from unassociated personal and business accounts. Lastly, the sentiment analysis calculates the sentiment score (1 for positive and 0 for negative) for each post based on expert keywords. SEAD then determine if a post contain SEA elements based on these combined scores; where a score <0.5 is labelled as safe (0) while a score >0.5 is labelled as malicious (1). Table 2 shows some examples of risk analysis of social media posts.

Table 2: Examples of risk analysis factor using SEAD

Posts	Source screening	Social graph	Sentiment	Risk analysis
{charlie→alice, ss: snortbl, 6:12:08 UTC, n2n: 1 (n degree)}, "hey can you share me your account password"	1	1	1	$1+1+1/3=1$
{bob→alice, ss: clean, 6:12:08 UTC, n2n: 1 (first degree)}, "hey it was a great day, we should hang out more"	0	0	0	$0+0+0/3=1$
{don→alice, ss: clean, 6:12:08 UTC, n2n: 1 (n degree)}, "hey can you share the photo we took together earlier"	0	1	0	$0+1+0/3=0.33$

5 Datasets and Attack Classes

The linguistic nature for SEA in social posts are deterministic; thus, we trained a five-classes ML model to classify variants of SEA threats found in Facebook posts. Since SEA threats revolves around the components outlined in SEAD; we can use the risk analysis in SEAD to filter for red flag instances to subsequently train the ML model. We use 5,000 selected Facebook posts that has risk analysis score >0.5 for the model training. The dataset is labeled with 5 classes; that is *pretexting*, *phishing*, *scareware*, *click baits*, and *Quid Pro Quo (QPQ)* by two annotators [27]. All attack classes have equal 1000 instances each; to prevent data imbalance. The two datasets are checked for consistency; any label discrepancies are discussed and determined based on mutual consensus of the experts. The definition of the classes is defined as:

- *Pretexting (PT)*-Social media posts in which the author impersonates co-workers, police, bank and tax officials, or other persons with right-to-know authority. The pretexter asks questions that are ostensibly required to confirm the victim's identity. They gather necessary personal data that can later be helpful by crafting wordlists for password guessing and cracking.
- *Phishing (PH)*-Phishing scams are email and text messages sent by attackers claiming to be from a reputable and trusted source. These campaigns prey on the curiosity or fear in victims to react irrationally against claims like stolen credit cards, leaked photos and other sensitive information that are sentimental. In most cases, the victims are manipulated to click on links to malicious websites or open attachments containing malware.
- *Scareware*-Scareware is disguised as pop up alerts when browsing to alert users with false alarms that their system or user accounts have been compromised. Users are deceived and counter-react by installing recommended anti-threat tools, often threats themselves. Unlike phishing,

Scareware is highly impactful as they are more relatable to real-time user actions and contexts, which trick users and lower their suspicious guard.

- *Click baits (CB)*-Baiting puts something enticing or curious in front of the victim to lure them into the social engineering trap. For example, carefully crafted email titles, free music download or survey that offers gift card are rewarding and warrant a few clicks. The rewards are more often or not more than meet the eyes, like getting a free mp3 that is embedded with malware; or free wallpapers that contain the recently popular mining-ware during this crypto craze. Click baits are dangerous as they often prevail against the weakened guard when users are presented with offers that are sometimes too good to be true.
- *Quid Pro Quo*-A social engineering tactic where the attacker attempts a trade of service for information. These attacks are targeted at less tech-literate people; and take advantage of human factors like curiosity and anxiety. For example, end-users are more likely to comply with IT support requests when facing technical issues and give up credentials freely for quick fixes. In recent days, remote login has become more prevalent for working from home; but few security landscapes are studied well enough to understand the potential threats.

In data preprocessing, the training data are pre-processed to reduce input noise. Firstly, each instance is truncated to 250 characters or 30 words. Then, the stop words and arbitrary digits in the text are removed (except common digits like date month years to preserve temporal context). The text is further transcoded into a common Unicode format that supports emoji commonly found in social posts for iOS, Android, and Windows. Lastly, some popular *Internet Slang* words is reconstructed to restore their linguistic meaning. For example, *AFAIK* to 'as far as I know'; *AMA* to 'ask me anything'. Some doodled words like '*w@t3r'* is converted back to '*water'* in ASCII. The final dataset contains formatted 5,000 social posts; we split at 7:3 ratio for model training and testing. Table 3 summarized the dataset attributes.

Table 3: Dataset attributes (Train:Test split at 8:2)

SEA types	Training		Testing	
	Instance count	Word count	Instance count	Word count
Pretexting	800	9626	200	2288
Phishing	800	12077	200	1737
Scareware	800	7931	200	1642
Clickbaits	800	9913	200	2311
Quid Pro Quo	800	9102	200	1989

6 RNN-LSTM for Social Engineering Classification

Traditional non-neural network classification treats multiple words as separate inputs; thus, they lack contextual information found in an actual sentence. Since social media posts can grow to 63,206 characters space, we trained a classification model using LSTM to accurately predict SEA based on linguistic meaning rather than relying on statistical odds. In LSTM, we can use a multiple word string to find out the class it belongs to. The LSTM contains three hidden layers to properly decode the properties of the actual meaning of characters, strings, words, and phrases in social media posts. The LSTM has 5 layers; the first layer is the embedded layer; followed by 3 hidden layers and lastly the output layer (see Fig. 5). We find that adding additional hidden layers do not improve validation

accuracy. In the first layer, we use 100 length vectors to represent each word. The position of words in the vector space is determined by the words that precedes and supersedes it. For example, the term 'money' is more likely associated to 'bank account' and 'spam' is more likely associated to an 'email account'. The next layer in the LSTM is the dense layer with 128 neurons. Each social post is first vectorized into a sequence of integers. These inputs are one-hot-encoded and set to the same length before the forward/backward propagation. The LSTM uses *softmax* for activation function and *Adam* as optimizer, widely used for multiclass classification. *Adam* optimizer is the better optimizer for handling sparse gradients and noisy problems. For loss function, the LSTM use *sparse_categorical_crossentropy* since some of the classes are mutually exclusive, i.e., when each sample belongs to precisely one class. A dropout layer is added after the first hidden layer to regulate the network and keep it from any bias possible. Then, another LSTM layer with 128 cells was followed by some dense layers. Lastly, the network converged to the output layer, which has five values representing the five different SEA categories. Fig. 6 outlines the detail RNN-LSTM algorithm.

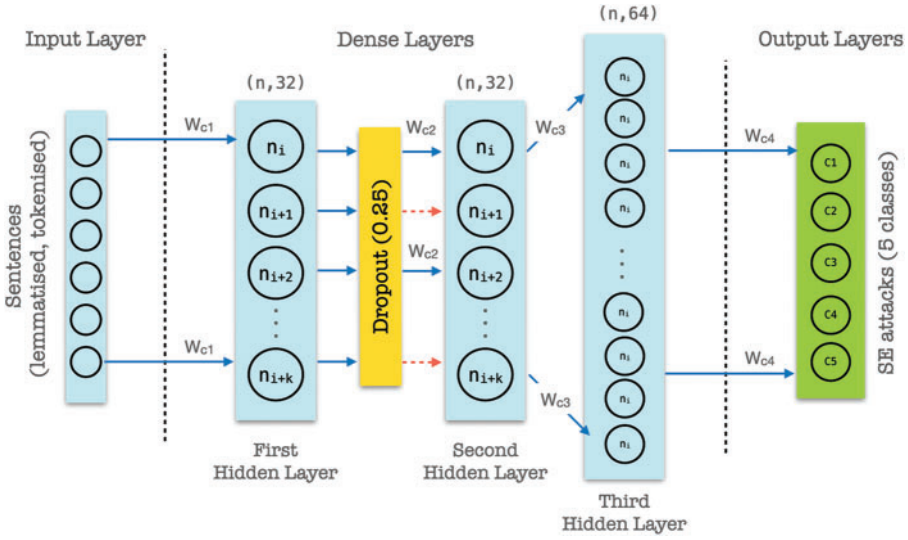

Figure 5: RNN-LSTM network architecture consisting of 3 hidden layers with dropout at 0.25

Function: RNN LSTM (training)
    def previous hidden state as prev_ct
    def current input as input
    def output as ot

Configure lstm_architecture
(n input layers, 3 hidden layers, 5 output layers, dropout=0.25, learn rate = 0.01)

Function: **RNN LSTM (training)**
concat *prev_ht* and *input* **into** *combine*
push *combine* **to** *forget layer* (candidate layer)
push *combine* **to** *input layer* (it)
    Calculate ct = cell state with *prev_ct * ft + candidate * it*
    Get ot = forward/backward propogate (*combine)*
Compute new *hidden state* (pointwise multiply *ot* and *ct*)

Function: RNN LSTM (prediction)
For all unseen samples, *n* to *ni*
    Forward propagate *n*, increment *i* by 1
    Classify class instances
Compute confusion matrix

Figure 6: RNN-LSTM algorithm

7 Model Optimisation using Pytorch's Adaptive Experimentation (AX)

Notice we do not specify any hyperparameter in the LSTM network. For model optimisation, we use Pytorch AX library for automatic hyparameter tuning. AX automatically experiment and find the most optimal hyperparameters configurations based on the lowest validation loss. For validation, we use stratified k-fold cross-validation to randomize the testing data and validation data. The AX configurations are shown in Table 4.

Table 4: Hyperparameter tuning for LSTM model

Learning rate	0.001 to 0.2
Dropout rate	0.1 to 0.5
Alpha	2 to 10
LSTM layers (hidden)	2/3
Number of epochs	10 to 25
Optimizer	Adam/RmsProp/stochastic gradient descent (SGD)

The tuning phase consists of two loops; (1) the outer loop is set to exit at the 25th iteration; (2) the inner loop has 10 iterations (n_splits = 10). At the outer loop, each iteration will split the training data into ten parts, and each combination of training and validation data is passed to the inner loop once. At the inner loop, the *get_next_trial( )* built-in function in Ax Client returns a set of trial parameterization used in the training and a trial index. Then, AX uses hyperparameters such as *alpha* and *the number of LSTM layers* to build a new model architecture for every trial. The formula on how the number of neurons in each layer is computed is shown as follows:

$$N_h = \frac{N_s}{alpha * (N_i + N_0)} \tag{1}$$

where Ns represents the number of samples in the training data, N_i represents the number of input neurons, and N_0 means the number of output neurons. Thus, N_i and N_0 are 20 and 1, respectively, and *alpha* is a factor between 2 and 10 where Ax Client selects it in each new trial. Meanwhile, we set the number of neurons in each hidden layer to be 2/3 of the input layer's size and add with the output layer's size (Heaton Research 2017). Since going deeper into the neural network (NN) is insignificant, we stop at 3 hidden LSTM layers. There are 250 trials in the experiment, resulting in 250 different combinations of hyperparameters sets. The trial results are saved into a JSON file, and the program will get the best hyperparameters from 250 trials using the built-in function get_best_parameters(). Then, the 'actual' model training starts using the best hyper-parameters found by AX. Next, we use *ModelCheckPoint* to monitor the validation loss and save the model with the lowest validation loss across multiple epochs. Finally, we apply *EarlyStopping* to stop the training process at the epoch before the val_accuracy starts to decline (a sign of overfitting).

8 Results & Performance Evaluation

To evaluate if the model generalizes well, we measures the *precision* and *recall* of the LSTM model on a synthetic data against several popular machine learning algorithms as there are no related benchmark. We use 1000 unseen samples annotated by experts for model testing as the ground truth; since popular dataset like KDDCup 99 and NSL-KDD do not contain the required features set. Table 5 shows the dataset attributes for model training and testing before pruning. Meanwhile,

precision is a solid indicator for correctly classifying *SEA* types while high *recall* indicates not missing out on too many correct instances. Table 5 compares the model performance using several popular ML classifier.

Table 5: Comparing classification precision and recall of SEA on several popular classifiers

Algorithm	Precision	Recall
KNN	0.71	0.62
DT (j48)	0.74	0.68
DBN	0.52	0.51
PCA	0.46	0.39
RF	0.78	0.71
MLP	0.78	0.73
DNN (LSTM)	0.84	0.81

Table 5 shows that the proposed (deep neural network) DNN-LSTM performs better by all metrics than the other models, scoring 0.84 in precision and 0.81 in recall. DNN-LSTM improved threats detection by 0.06% when compared to the nearest ML techniques; which translates to 60 more threats classified in every 1000 random samples. DNN-LSTM has a recall that is 0.08% higher than multi layer perceptron (MLP); which means it is more robust and can detect more variants of threats. The recall rate is somewhat lower across the board, which is not uncommon for multi-class classification for unstructured long text. Most traditional ML predictions are based on term frequency and bag of words, so they struggle with longer sentences. Surprisingly, the performance gap on standard ML-like k-nearest neighbours (KNN), decision tree (DT) and RF against neural networks is marginal, albeit more lightweight and faster trained. KNN is a clustering technique that does not require large training data; DT is always known to work well with simple text data classification problems. RF is an ensemble that combines multiple DT to improve accuracy further. Meanwhile, the principal components analysis (PCA) and deep belief network (DBN) gap are more profound; the algorithms classify words as separate entities in a sentence, thus losing some spatial cues of the sentence's linguistic properties. Like LSTM, MLP learns the semantic using forward/backward propagation on a neural network and is slightly less accurate. We compare an optimized LSTM and an MLP with the best NN configurations and hyperparameters. We ramify that there might be helpful temporal information in sentence structure, like words' choice and relative order of occurrences. Although MLP converged faster, adding the memory cell in LSTM enables the network to encourage desired behaviour from the error gradient on every step of the learning process. Since the NN is a black box, we can only imply that the NN model mainly benefits from training each sentence as an entity rather than individual words in standard ML. Explainable Ai is out of the scope of this paper. Using LSTM, we ramify a near-perfect model is somewhat tricky when intentions are implied from words. First, we must consider the gap in linguistic literacy to intrinsic SEA intentions expressed in words. Detecting SEA on social posts also lacks contexts like timing, topics, author's criminal history, the topic of a post, and the political and cultural landscape of the people involved.

The confusion matrix in Table 6 shows the per-class classification accuracy for SEA. We measure using True Positive Rate (TPr); which is commonly used to indicate the number of correctly classified instances in a multiclass-classifications. DNN-LSTM predicts 'pretexting' most accurately 0.905 TPr; while 'quid pro quo' is somewhat lower at 0.755 TPr. Most of the true negatives come from SEA

being misclassified as pretexting. We ramify that some posts that contain identity indicators in QPQ attack inadvertently confuse the model; since the same instance contains both the element of PT and QPQ. Meanwhile, 'Scareware' (SW) which contains wordings like 'your system' or 'your IP address is infected' often shares similar semantics with PT and PH, thus, it is occasionally misinterpreted as other attack classes. We find the 'Click baits (CB)' class to have the highest TPr, mainly due to the choice of words with phrases like 'you have won', 'congratulations', 'you are selected as winners' which is unique to the corresponding class. We conclude that the high TPr for PT and PH made our mode a robust one; since more than 80% of SEA originates from these two classes.

Table 6: Confusion matrix for social engineering attack classifications

	PT	PH	SW	CB	QPQ	TPr
Pretexting (PT)	181	5	2	7	5	0.905
Phishing (PH)	19	169	6	2	4	0.845
Scareware (SW)	37	2	157	1	3	0.785
Clickbaits (CB)	10	4	2	182	2	0.91
Quid Pro Quo (QPQ)	23	5	11	10	151	0.755

Our DNN-LSTM has not been tested with catchphrases that are specially curated to beat the system; for example, using words that carry positive sentiment but with bad intentions. Semantically, a social media post is geared toward a first-person narrative and is more casually toned. Meanwhile, a phishing email is structurally more rigid and formal than conversational. The scope of phishing here refers to social media post that is fishing for sensitive information, rather than other common phishing emails. Thus, any dataset with different tonal, thematic, and linguistic properties has to be retrained for a more robust, all-around classification. Note that the SEAD method, as of writing, only supports sequence to sequence English words. Other NLP toolkit is similar to NLTK in that understanding foreign linguistic structure is required to port SEAD for other languages' use. Our DNN-LSTM is not sensitive to future contexts in a sentence; as compared to a (bidirectional) Bi-LSTM that makes use of sequence information in both forward and backward directions. For example, consider DNN-LSTM modelled on a sentence "share your number with us . . . "; without the follow-up sentence "we ensure all your banking details are encrypted". The DNN-LSTM lacks the future contexts here; whereby a Bi-LSTM can leverage on 'banking details' in the second sentence to understand the 'number' refers to bank account details. Bi-LSTM, however, requires more compute for bidirectional propagation, unlike simpler variants like gated recurrent unit (GRU) which train faster than both DNN-LSTM and Bi-LSTM at the cost of prediction accuracy. We have not tested the impact of adding future contexts to SEA classifications, although the dataset is time-series ready and can be easily retrained on Bi-LSTM.In short, some social engineering attacks are not expressed through words and are never captured in any linguistic semantics.

9 Conclusion

Social engineering attacks (SEA) are finding their way into social media posts. They prey on victims' insecurities and fear to bait them into clicking on malicious links and giving up sensitive information involuntarily. Recently, attackers cleverly craft social posts to sound like most regular posts but carry the intrinsic motivation, becoming more intimate and personal to minimise suspicion. We designed the SEAD pipeline to automatically label a social media post as malicious or usual based

on source screening, social graph analysis and sentiment analysis. We find that most SEA can be stopped at the source level, scrutinising posts made by suspicious accounts. Leveraging on Meta's public API, we add weighted bias to social posts made by account with poor standings for a potential threat. With SEAD, we cut the amount of 'useful' samples (posts that are likely SEA) before labelling them to train the RNN-LSTM model for SEA classifications. For dataset integrity, the dataset is annotated by two experts similar to the guideline outlined in the VSD2014 dataset. We trained RNN-LSTM to identify five SEA variants: phishing, pretexting, scareware, clickbait, and quid pro quo. The RNN-LSTM outperforms other ML classifiers; it scored 0.84 in precision and 0.81 in recall rate. We imply that SEA threats can be determined based on the sentence semantics; since posts that gather information often contains wordings and phrases that share similar linguistics properties. We do not investigate a per-class performance among the SEA types, but further studies can shed light on some other SEA variants or composite SEA types and possibly SEA that are not directly overt using linguistic features.

Funding Statement: The authors acknowledge the funding support of FRGS/1/2021/ICT07/UTAR/02/3 and IPSR/RMC/UTARRF/2020-C2/G01 for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Algarni, Y. Xu, T. Chan and Y. -C. Tian, "Social engineering in social networking sites: Affect-based model," in *8th Int. Conf. for Internet Technology and Secured Transactions (ICITST-2013)*, London, UK, 2013.

[2] M. Huber, S. Kowalski, M. Nohlberg and S. Tjoa, "Towards automating social engineering using social networking sites," in *2009 Int. Conf. on Computational Science and Engineering*, Vancouver, BC, Canada, 2009.

[3] S. Dasgupta, A. Piplai, A. Kotal and A. Joshi, "A comparative study of deep learning based named entity recognition algorithm for cybersecurity," in *2020 IEEE Int. Conf. on Big Data (Big Data)*, Atlanta, GA, USA, 2020.

[4] S. Li, X. Yun, Z. Hao, X. Cui and Y. Wang, "A propagation model for social engineering botnets in social networks," in *2011 12th Int. Conf. on Parallel and Distributed Computing, Applications and Technologies*, Gwangju, Korea (South), 2011.

[5] C. Lorenzen, R. Agrawal and J. King, "Determining viability of deep learning on cybersecurity log analytics," in *IEEE Int. Conf. on Big Data (Big Data)*, Seattle, WA, USA, 2018.

[6] D. Gumubas, T. Yildirim, A. Genovese and F. Scotti, "A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems," *IEEE Systems Journal*, vol. 15, no. 2, pp. 1717–1731, June 2021.

[7] O. Jaafor and B. Birregah, "Multi-layered graph-based model for social engineering vulnerability assessment," in *2015 IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, Paris, France, 2015.

[8] A. Algarni, Y. Xu and T. Chan, "Social engineering in social networking sites: The art of impersonation," in *2014 IEEE Int. Conf. on Services Computing*, Anchorage, AK, USA, 2014.

[9] H. Wilcox and M. Bhattacharya, "A framework to mitigate social engineering through social media within the enterprise," in *2016 IEEE 11th Conf. on Industrial Electronics and Applications (ICIEA)*, Hefei, China, 2016.

[10] S. Gupta, A. Singhal and A. Kapoor, "A literature survey on social engineering attacks: Phishing attack," in *2016 Int. Conf. on Computing, Communication and Automation (ICCCA)*, Greater Noida, India, 2016.

[11] F. Mouton, M. Teixeira and T. Meyer, "Benchmarking a mobile implementation of the social engineering prevention training tool," in *2017 Information Security for South Africa (ISSA)*, Johannesburg, South Africa, 2017.
[12] G. Karatas, O. Demir and O. Koray Sahingoz, "Deep learning in intrusion detection systems," in *2018 Int. Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, Ankara, Turkey, 2018.
[13] H. Sun and R. Grishman, "Lexicalized dependency paths based supervised learning for relation extraction," *Computer Systems Science and Engineering*, vol. 43, no. 3, pp. 861–870, 2022.
[14] M. Macas and C. Wu, "Review: Deep learning methods for cybersecurity and intrusion detection systems," in *2020 IEEE Latin-American Conf. on Communications (LATINCOM)*, Virtual Conference, 2020.
[15] T. Gaber, A. El-Ghamry and A. Hassanien, "Injection attack detection using machine learning for smart IoT applications," *Physical Communication*, vol. 52, no. 101685, pp. 1–14, 2022.
[16] S. Salloum, T. Gaber, S. Vadera and K. Shaalan, "Phishing website detection from URLs using classical machine learning ANN model," *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 399, pp. 509–523, 2021.
[17] R. Staudemeyer, "Applying long short-term memory recurrent neural networks to intrusion detection," *South African Computer Journal*, vol. 56, no. 1, pp. 136–154, 2015.
[18] R. Krishnan and N. Raajan, "An intellectual instrusion detection system model for attacks classification using RNN," *International Journal of Pharmaceutical Technology and Biotechnology*, vol. 8, no. 4, pp. 23157–23164, 2016.
[19] C. Yin, Y. Zhu, J. Fei and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
[20] J. Kim, J. Kim, H. Thu and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *2016 Int. Conf. on Platform Technology and Service (PlatCon)*, Jeju, Korea (South), February 2016.
[21] T. Le, J. Kim and H. Kim, "An effective intrusion detection classifier using long short-term memory with gradient descent optimization," in *2017 Int. Conf. on Platform Technology and Service (PlatCon)*, Busan, Korea (South), February 2017.
[22] Y. Zhang, X. Chen, L. Jin, X. Wang and D. Guo, "Network intrusion detection: Based on deep hierarchical network and original flow data," *IEEE Access*, vol. 7, pp. 37004–37016, 2019.
[23] Z. Luo, W. Cai, Y. Li and D. Peng, "The correlation between social tie and reciprocity in social media," in *Proc. of 2011 Int. Conf. on Electronic & Mechanical Engineeing and Information Technology*, Harbin, China, 2011.
[24] H. AlSalman, "An improved approach for sentiment analysis of arabic tweets in twitter social media," in *2020 3rd Int. Conf. on Computer Applications & Information Security (ICCAIS)*, Riyadh, Saudi Arabia, 2020.
[25] J. Hu, M. Liu and J. Zhang, "A semantic model for academic social network analysis," in *2014 IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, Beijing, China, 2014.
[26] Y. Kano and T. Nakajima, "Trust factor of social engineering attacks on social networking services," in *2021 IEEE 3rd Global Conf. on Life Sciences and Technologies (LifeTech)*, Nara, Japan, 2021.
[27] P. Leonov, A. Vorobyev, A. Ezhova, O. Kotelyanets, A. Zavalishina *et al.,* "The main social engineering techniques aimed at hacking information systems," in *2021 Ural Symp. on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, Yekaterinburg, Russia, 2021.