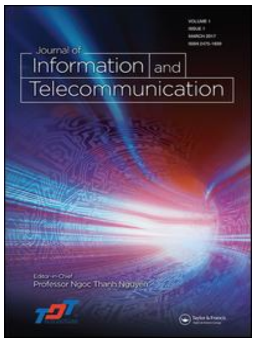

Journal of Information and Telecommunication

ISSN: 2475-1839 (Print) 2475-1847 (Online) Journal homepage: https://www.tandfonline.com/loi/tjit20

SEADer++: social engineering attack detection in online environments using machine learning

Merton Lansley, Francois Mouton, Stelios Kapetanakis & Nikolaos Polatidis

To cite this article: Merton Lansley, Francois Mouton, Stelios Kapetanakis & Nikolaos Polatidis (2020): SEADer++: social engineering attack detection in online environments using machine learning, Journal of Information and Telecommunication, DOI: 10.1080/24751839.2020.1747001

To link to this article: https://doi.org/10.1080/24751839.2020.1747001

JOURNAL OF INFORMATION AND TELECOMMUNICATION
https://doi.org/10.1080/24751839.2020.1747001

SEADer++: social engineering attack detection in online environments using machine learning

Merton Lansley[a], Francois Mouton [b,c], Stelios Kapetanakis[c] and Nikolaos Polatidis [d]

[a]Department of Informatics, University of Sussex, Brighton, UK; [b]Department of Applied Science, Noroff University College, Oslo, Norway; [c]Department of Computer Science, University of the Western Cape, Bellville, South Africa; [d]School of Computing, Engineering and Mathematics, University of Brighton, Brighton, UK

ABSTRACT
Social engineering attacks are one of the most well-known and easiest to apply attacks in the cybersecurity domain. Research has shown that the majority of attacks against computer systems was based on the use of social engineering methods. Considering the importance of emerging fields such as machine learning and cybersecurity we have developed a method that detects social engineering attacks that is based on natural language processing and artificial neural networks. This method can be applied in offline texts or online environments and flag a conversation as a social engineering attack or not. Initially, the conversation text is parsed and checked for grammatical errors using natural language processing techniques and then an artificial neural network is used to classify possible attacks. The proposed method has been evaluated using a real dataset and a semi-synthetic dataset with very high accuracy results. Furthermore, alternative classification methods have been used for comparisons in both datasets.

ARTICLE HISTORY
Received 23 December 2019
Accepted 22 March 2020

KEYWORDS
Social engineering; online chat environments; natural language processing; machine learning; neural networks; ensemble learning

1. Introduction

There have been various research projects demonstrating the need for an automated system to recognize Social Engineering (SE) attacks. Many of these projects show proof of concept models using a variety of different techniques, most commonly Natural Language Processing (NLP) and Machine Learning (ML) methods like Artificial Neural Networks (ANN). Whilst these exist, there is a lack of evidence that the theory has been implemented and proved working, with very little evidence of the rate of success (Mouton et al., 2018; Peng et al., 2018; Sawa et al., 2016; Tsinganos et al., 2018).

Human Psychology plays a large part in the creation of NLP software as it is important to understand how people socialize and behave with others. A common and well-regarded psychologist, Dr. Robert Cialdini proposed the 6 principles of persuasion (Resnik, 1986): Authority, Scarcity, Liking/Similarity, Reciprocation, Social Proof and Commitment/Consistency. Further studies by psychologists also follow a similar consensus that these are the

CONTACT Nikolaos Polatidis N.Polatidis@Brighton.ac.uk School of Computing, Engineering and Mathematics, University of Brighton, Brighton, UK

key factors in defining a persuasive person. Using these key features as a base, Computer Scientists theorized and attempted to build systems that recognized these traits. The research projects on this area can be loosely broken down into three stages, Data Pre-Processing, Feature Extraction and Aggregation of Results as shown in Figure 1.

Most research approaches into this problem domain have some relevant data and these data are usually pre-processed. The method of pre-processing differs per system, but generally involves the same goal; preparing the data for classification. The contextual, or meta data, such as time, date and IP addresses are captured in a relationship with the original data. This can be used for correlating different dialogues with one another, one by matching the IP as an internal or external agent and secondly by highlighting potential reoccurring adversaries that may persist under different aliases. The dialogues are sanitized to remove any erroneous content such as HTML tags or corrupt texts. The data can then be parsed using a Natural Language Processing (NLP) parser such as the Stanford CoreNLP (Manning et al., 2014). NLP parsers provide a vast array of tools that are able to dissect the human language into dependency trees, define grammatical relations and their Parts of Speech (POS) to name a few common uses. Using these libraries, it allows programmers to create algorithms which are able to classify conversations based on the linguistic features the parser extracts.

This paper extends our previous conference version found in Lansley et al. (2019) and is based on the concept that a dialogue is taking place in an online chat environment. To determine if a dialog between two interlocutors is a Social Engineering attack, certain criteria, or else known as features, need to be chosen, a process known as feature selection. The features are selected before the three stages and are commonly based on the principles of persuasion, common phishing tactics like malicious links and the history of the attacker. We can then classify the data based on these features and produce a 'score' of how strongly the selected dialog matches the criteria. Each implementation designed to detect each feature will use a variety of classification techniques, such as: Fuzzy Logic, Topic Blacklists, Decision Trees, Random Forest and Neural Networks depending on what needs to be extracted. Furthermore, an automated system requires the output of a clear decision, albeit a probabilistic decision, this can be done by

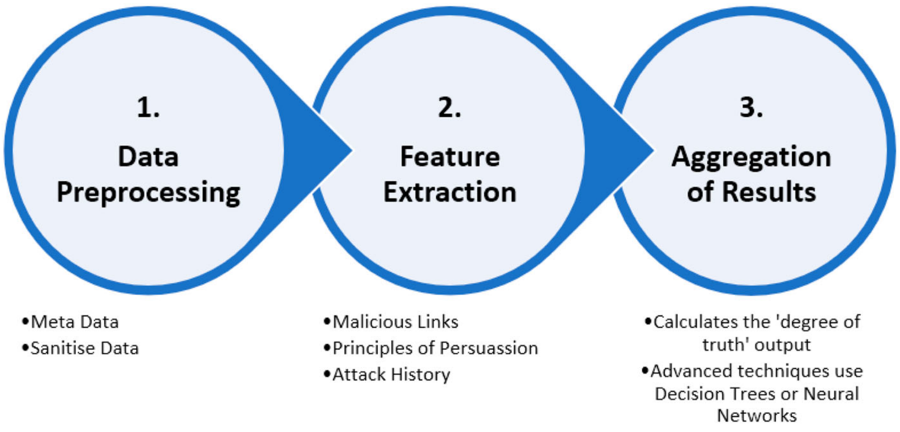
1.
Data Preprocessing
2.
Feature Extraction
3.
Aggregation of Results
•Meta Data
•Sanitise Data
•Malicious Links
•Principles of Persuassion
•Attack History
•Calculates the 'degree of truth' output
•Advanced techniques use Decision Trees or Neural Networks

Figure 1. The data processing pipeline.

aggregating the outputs from the Feature Extraction process. The results of each feature are weighted by importance, and at basic level can be averaged to give a Fuzzy logic prediction. By using more advanced techniques such as decision trees or neural networks, the weight of each feature can be calculated programmatically to determine which features carry the most importance regarding whether or not a social engineering attack is taking place.

The following contributions are delivered:

(1) A method for detecting social engineering attacks in online chat environments is proposed.
(2) The proposed method has been evaluated using a real and a semi-synthetic dataset with the results validating our approach.

The rest of the paper is organized as follows: Section 2 presents the related work, Section 3 delivers the proposed method, Section 4 contains the experimental evaluation and Section 6 is the conclusions part.

2. Related work

An early implementation of SE detection in real-time telephone systems is SEDA (Social Engineering Defense Architecture) (Hoeschele & Rogers, 2005). The researchers mainly focused on identifying repeat callers using their voice signatures. Later a proof of concept model of SEDA (Hoeschele & Rogers, 2005) was produced being able to correctly identify all attacks in their dataset. A method of detection used by Sawa et al. (2016) is to identify the questions requesting private information and commands requesting that the user perform tasks they are not authorized to perform. This technique uses a manually derived topic blacklist of verb-noun pairs for which they state should be built around security policies associated with a system. This work is taken further in Peng et al. (2018) by identifying 4 main attack vectors; the urgency of the dialog, negative commands and questions, whether the message is likely automated identified by a generic greeting. Finally, they use a reputable cyber security service, Netcraft, to check the safety of a URL. Instead of manual blacklist creation, they use a large corpus of phishing emails to generate a topic blacklist using a Naive Bayes classier. Some approaches like SEADMv2 (Mouton et al., 2018) and MPMPA (Jamil et al., 2018) use complex state machines in order to map out the pathways that can be followed as a checklist-type system to mitigate an attack. This proposal is suited where there are multiple authorization layers that might prevent a request from being carried out. These two separate machines provide some overlap, but the explicit state definition required could be limiting to where these can be applied. The nature of SE attacks is unpredictable meaning that new methods of attacks are always being introduced. The SEADM versions state machines still rely on the user input for changing state, which means the chance for user error and naivety is still present. The works of Tsinganos et al. (2018) covers an extensive overview of the existing systems and provide a comprehensive recognition of subsystems for their detection architecture; in influence, deception, personality, speech act and past experiences. The work of Bhakta and Harris (2015) provides a semantic-based approach of dialogues to detect social engineering attacks. In Heartfield and Loukas (2018) the authors show that the human factor is the weakest link in social engineering attacks and based on a human study the prove that. In Bezuidenhout et al. (2010) and Mouton et al. (2015) the authors provide a theoretical foundation that potentially could be used in real systems to detect attacks. In the works of Nicholson et al. (2017) it is explained how social engineering attacks can potentially be detected, whereas in Krombholz et al. (2015) and Mouton et al. (2016) different examples of attacks with scenarios are presented. In addition to the works mentioned, there are numerous other articles from relevant domains that can be useful such as the one in Nguyen and Nguyen (2016) where the authors performed an influence analysis of the number of members on the quality of knowledge in a collective, the work in Saadat Javad and Koofigar (2017) where a neural network is used along with harmony for searching, the one in Precup and David (2019) where nature inspired optimization algorithms for fuzzy control servo systems are discussed and the one found in Abed-alguni (2019) where an Island-based cuckoo search algorithm with highly disruptive polynomial mutation is proposed.

3. Proposed method

The proposed method consists of several steps to preprocess the dialogs into a dataset for classification. The last steps are applying the classifier. All the steps explained below, have been written in the Python programming language. The SymSpellpy library (a Python port of SymSpell) was used for spelling, the Web of Trust (WOT) Application Programming Interface (API) was used to check any links. In Section 3.1 the dataset pre-processing steps are described, which are necessary to create a new dataset with numerical values that can be used for classification purposes. In Section 3.2 an artificial neural network multi-layer perceptron (MLP) classifier is applied to the dataset, whereas in Section 3.3 an ensemble learning machine learning method has been developed.

3.1. Data pre-processing and preparation

Steps 1 to 5 check for malicious links, steps 6 and 7 determine the spelling quality, and steps 8 to 11 determine the intent of the text, using a pre-defined blacklist. The score of each feature can be given by: Link score, S_L, Spelling score, S_{SP}, and Intent score, S_I. Each score is then scaled down to between 0 and 1. After the pre-processing of the dialogues (steps 1–11), the classification dataset has the following 4 labels: (1) Intent, (2) Spelling, (3) Link and (4) attack or no attack.

The steps are as follows:

(1) Extract all URLs from the dialog text using a regex pattern matcher.
(2) If the text contains URLs, send the link/s to the WOT API to evaluate if the web link is malicious.
(3) The WOT API returns the reputation of the site (value between 0 and 100), the confidence of the given reputation (value between 0 and 100) and the identifying categories (17 in total) that identify the nature of the website. The broad categories and example subcategories are as follows:
    - 1XX Negative (101 Malware, 103 Phishing, 104 Scam, 105 Potentially illegal etc.)

- 2XX Questionable (201 Misleading claims or unethical, 205 spam, 207 ads / popups etc.)
- 3XX Neutral (301 Online tracking, 302 controversial, 303 political etc.)
- 5XX Positive (501 A good site)

(4) If the returned category is of group 1XX or 2XX, then $S_L = 1$.

(5) Otherwise, divide the reputation by 100 and take it away from 1 as shown in Equation (1).

$$S_L = 1 - \frac{\text{reputation value}}{100} \tag{1}$$

(6) Check for spelling using the SymSpellpy library.

(7) the best suggested spelling correction, determine the number of misspelled words, given by x. This number is then scaled between 0 and 1 by applying Equation (2). The value of represents the spelling quality, where higher values represents poorer spelling. Rather than a linear function, an exponential function is used to rate a higher number of spelling mistakes more severely. To adjust the rate at which the score tends towards 1, the constant α can be varied to affect how harsh you want to be on spelling errors. After extensive testing it was identified that 0.5 allowed the text to contain a small number of mistakes without creating a high score. For example, if α is set to 0.5 and x=1 then $S_{SP} = 0.39$, if x=5 then $S_{SP} = 0.92$.

$$S_{SP} = 1 - e^{-\alpha x} \tag{2}$$

(8) The next step uses the corrected spelling of the dialog, and checks it against a blacklist, derived from 48 security policy style words. This can be easily populated with company or environment-related words such as: credentials, passwords, database etc. The number of blacklist matched words is given by M_B.

(9) The algorithm at this step checks for intent verbs and adjectives such as need, must, urgent etc. This value is given by M_I

(10) To tune the results, values M_B and M_I are multiplied by the weights W_B and W_I, weighted at values of 2 and 1 retrospectively. This step has been added as an equation in case someone wants to change the weight of this step, if it is considered more important. The value x can hence be given by Equation (3).

$$x = (M_B \times W_B) + (M_I \times W_I) \tag{3}$$

(11) Then, the value of x is normalized in Equation (4), using the same exponential function as Equation (2). Where $\alpha = 0.4$ to give the best output. A higher value of S_I indicates a higher concentration of blacklisted words in the text.

$$S_I = 1 - e^{-\alpha x} \tag{4}$$

(12) At this step the original dialogue dataset is being checked to identify which dialogue was indeed an attack and assign the true (1) or false (0) value to the new dataset used for classification.

In summary, the above steps are used to read natural language and convert it into numerical values with the labels (1) Intent, (2) Spelling, (3) Link and (4) attack or no attack. Then the new dataset with these values is used with a machine learning classification algorithm to identify attacks. Sections 3.2 and 3.3 describe an MLP and an ensemble learning classifier, respectively.

3.2. Classification of attacks using an artificial neural network

After the dataset is populated and the MLP classifier is applied. If the output is high, then it is considered an attack otherwise it is not considered an attack. Initially, we define an activation function as: $g(z)$ with x input values and w weights as input. The activation function is shown in Equation (5).

$$z = w_1x_1 + w_2x_2 + \cdots + w_mx_m \tag{5}$$

If $g(z)$ is greater than a given threshold θ, the output is 1 or -1 otherwise, as shown in (6).

$$g(z) = \begin{cases} 1, & \text{if } z > \theta \\ -1, & \text{else} \end{cases}$$
$$\text{where} \quad z = w_1x_1 + w_2x_2 + \cdots + w_mx_m = \sum\nolimits_{j=1}^{m} x_jw_j = w^Tx \tag{6}$$

For the next step, we use Rosenblatt's perceptron rule to update the weights as follows: (a) Each weight was initialized with small random numbers and (b) for each iterative training step for each input x the output value was calculated, and the weights were updated.

The output update is defined in Equation (7), with $\Delta w_j = e(target(i) - output(i))x_j^i$ and e is the learning rate, *target* the actual (true) class label and *output* the predicted label. Weight updates were iterative, and updates were performed simultaneously.

$$w_j^+ = w_j + \Delta w_j \tag{7}$$

3.3. Classification of attacks using ensemble learning

For the ensemble learning approach a Gaussian Naïve Bayes classifier has been used along with a Decision Tree classifier and a Random Forest classifier based on a soft voting approach. Initially the three algorithms are explained and then a pseudocode of the ensemble algorithm is shown. The Gaussian Naïve Bayes defined in Equation (9) implements the Naïve Bayes algorithm and is defined in Equation (8), while it assumes that the likelihood of features is Gaussian. Naïve Bayes gives a probability P for a class y, belonging or not to X as shown in Equation (8). In Equation (9) where the Gaussian Naïve Bayes is defined, and the parameters σ_y and μ_y are estimated using the maximum likelihood. μ is the mean and σ is the standard deviation.

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \tag{8}$$

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}}\exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right) \tag{9}$$

Then a Decision Tree classifier is defined with unlimited leaf nodes. A Decision Tree is a classifier where a tree is drawn upside down with its root at the top which is being a social engineering attack or not. Then the attributes or else known as features found in the dataset are used to make decision by setting values and using *IF/THEN/ELSE* statements. For example, in the dataset the first attribute is the intent. After the root is set as attack at the final leaf a *YES/NO* decision needs to be made. Subsequently, the numerical values found in this attribute are used to decide which number should be the cut-off point for a yes or no decision. The same continues for all attributes until a final decision is made.

Following that a Random Forest classifier is defined, comprised of 10 Decision Tree classifiers with different cut-off points. The Random Forest returns a yes or no decision based on majority of answers received by the decision trees.

Algorithm 2. Ensemble learning pseudocode

```
  Input: Social engineering classication dataset
  Output: Classification as an attack or not
1: LOAD data
2: positives ← 0
3: negatives ← 0
4: attack ← 0
5: for entry of the dataset do
6:     LOAD Gaussian NB classifier
7:     result ← GaussianNBclassification
8:     if result is positive then
9:         positives ← positives + 1
10:    else
11:        negatives ← negatives + 1
12:    end if
13:    LOAD Decision Tree classifier
14:    result ← Decision Tree Classifier
15:    if result is positive then
16:        positives ← positives + 1
17:    else
18:        negatives ← negatives + 1
19:    end if
20:    LOAD Random Forest classifier
21:    result ← Random Forest Classifier
22:    if result is positive then
23:        positives ← positives + 1
24:    else
25:        negatives ← negatives + 1
26:    end if
27: end for
28: if positives > negatives then
29:    return attack = 1
30: else
31:    return attack = 0
32: end if
```

4. Experimental evaluation

The experimental evaluation took place in an offline environment using a real dataset and a semi-synthetic dataset, which we call the standard dataset and the compound dataset. Following from Section 3, we have built both datasets using a social engineering attack dataset with 147 entries classified as an attack or not. After the preprocessing steps followed in section three, we derived the following four classification labels: (a) intent (b) spelling (c) link (d) is attack.[1] Furthermore, the standard dataset contains 147 entries obtained from Bezuidenhout et al. (2010), while the compound dataset is based on the 147 entries plus 600 entries from customer support-based tweets from Twitter, none of which are classified as attacks. Both dataset text entries have been converted to numerical-based classification datasets with four entries each. Three labels for the respective data and one with a yes or no (1 or 0) value of a conversation being a social engineering attack or not. To both datasets a small number of links to websites have been added, with some being malicious. Furthermore, it should be noted that the dataset is imbalanced with most entries not being an attack.

The accuracy metric has been used for the evaluating the classification models. The metric calculates the fraction of the prediction that each model got right. Equation (10) represents the accuracy where TP stands for true positives, TN for true negatives, FP for false positives and FN for false negatives.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{10}$$

The MLP classification model described in Section 3.2 is evaluated initially using the accuracy metric and compared to the Decision Tree and Random Forest classifiers. The results of the evaluation as shown in Tables 1–7 are based on the standard and compound datasets, respectively, and a 5-fold cross-validation approach has been used. In Tables 1–6 the MAX and MIN values represent the maximum and minimum values of the 5-fold returned after each time the algorithm ran and the MEAN is the average value. Each algorithm was executed 10 times. Table 7 presents the average mean results for the standard and compound datasets, respectively.

Three algorithms from the SciKit machine learning library have been used for comparison purposes: Decision Tree, Random Forest and Neural Network multi-layer perceptron. The settings used for the algorithms were the following:

Decision Tree: SciKit learn settings with default settings which included unlimited leaf nodes.

Table 1. Decision Tree results for the standard dataset.

MEAN	MAX	MIN
0.736231884	0.826086957	0.583333333
0.692885375	0.782608696	0.625
0.68442029	0.833333333	0.458333333
0.650362319	0.695652174	0.583333333
0.656347826	0.76	0.608695652
0.659914361	0.826086957	0.5
0.694762846	0.826086957	0.416666667
0.647463768	0.791666667	0.5
0.65	0.695652174	0.583333333
0.741798419	0.833333333	0.652173913

Table 2. Random forest results for the standard dataset.

MEAN	MAX	MIN
0.683333333	0.791666667	0.608695652
0.631521739	0.708333333	0.565217391
0.691304348	0.791666667	0.608695652
0.657608696	0.75	0.565217391
0.651811594	0.782608696	0.333333333
0.700724638	0.75	0.666666667
0.647826087	0.75	0.5
0.736067194	0.782608696	0.708333333
0.699130435	0.8	0.608695652
0.72826087	0.826086957	0.583333333

Table 3. Multi-layer perceptron results for the standard dataset.

MEAN	MAX	MIN
0.709057971	0.75	0.652173913
0.74384058	0.826086957	0.652173913
0.673043478	0.8	0.565217391
0.640217391	0.782608696	0.47826087
0.75273386	0.833333333	0.625
0.65085639	0.739130435	0.541666667
0.751449275	0.826086957	0.565217391
0.659826087	0.739130435	0.56
0.68173913	0.8	0.52173913
0.65	0.695652174	0.583333333

Table 4. Decision tree results for the compound dataset.

MEAN	MAX	MIN
0.908050847	0.916666667	0.9
0.921481816	0.957983193	0.900826446
0.921386555	0.941176471	0.890756303
0.923023127	0.95	0.890756303
0.921470588	0.957983193	0.9
0.923120703	0.941176471	0.909090909
0.909774011	0.93220339	0.891666667
0.919747899	0.932773109	0.908333333
0.913009121	0.925619835	0.899159664
0.914705882	0.933333333	0.890756303

Table 5. Random forest results for the compound dataset.

MEAN	MAX	MIN
0.902910619	0.95	0.857142857
0.919787138	0.941176471	0.900826446
0.909703336	0.924369748	0.899159664
0.92640056	0.941666667	0.915966387
0.921414566	0.95	0.883333333
0.919803922	0.941176471	0.891666667
0.916468927	0.940677966	0.883333333
0.911358543	0.933333333	0.883333333
0.931468273	0.941666667	0.917355372
0.911372549	0.916666667	0.907563025

Table 6. Multi-layer perceptron results for the compound dataset.

MEAN	MAX	MIN
0.929803922	0.949579832	0.908333333
0.921442577	0.932773109	0.908333333
0.926414566	0.941666667	0.915966387
0.923107345	0.933333333	0.908333333
0.936442577	0.949579832	0.915966387
0.919789916	0.949579832	0.891666667
0.923120934	0.949579832	0.899159664
0.924733894	0.941176471	0.899159664
0.924747899	0.941666667	0.907563025
0.921398477	0.949579832	0.907563025

Table 7. Comparison of algorithms for the standard and the compound datasets.

(a)		(b)	
ALGORITHM	RESULT	ALGORITHM	RESULT
Decision tree	0.681	Decision tree	0.918
Random forest	0.683	Random forest	0.917
Multi-layer perceptron	0.691	Multi-layer perceptron	0.925

Random Forest: SciKit Random Forest classifier with 50 Estimators.
Neural Network: SciKit MLP classifier with lbfgs solver,$\alpha = 1e - 5$ and hidden layer size = (15,15,15).

The above results show that the MLP classifier is more accurate compared to the others. However, this doesn't mean that it is the best solution for the social engineering attack detection problem due to the imbalanced dataset. Solutions which assume that most entries are negative can produce high accuracy. Therefore, alternative metrics have been used to show that despite of the high accuracy of the MLP classifier an ensemble method is more appropriate when it comes to classification of attacks.

Moreover, an 80%/20% training/testing approach has been used for splitting the dataset for this part of the evaluation since the size of the dataset is considered small for 5 or 10 cross fold validation. To maximize the validation of the results each test has been executed ten times with the results of each execution and the mean presented. The metrics used are the accuracy as defined in Equation (10), the F1 measure defined in Equation (13) which is based on the precision and recall defined in Equations (11) and (12), respectively. F1 is the weighted average of precision and recall and takes into consideration both false positives and false negative into account. It is a useful metrics for imbalanced datasets. Finally, the Area Under the Curve (AUC) has been used. This is a metric based on the True Positive rate (Sensitivity or recall) defined in Equation (14) and on the False Positive Rate (Specificity) defined in Equation (15). It can be used for binary classification problems and it returns the probability of when selecting a random entry, this will be classified correctly. The AUC represents the area under the curve of a graph plotted with the False Positive rate against the True Positive rate in a 0 to 1 scale in different points between as shown in Figure 2, which however shows the plot only and not any random or sample curve. Last but not least it should be mentioned that in all metrics higher values are better. Especially in AUC it should be also noted that a value of 0.5 in 1 means that half of the instances are classified correctly, therefore any value ranging from 0.5 and below means that the classifier doesn't work as expected.

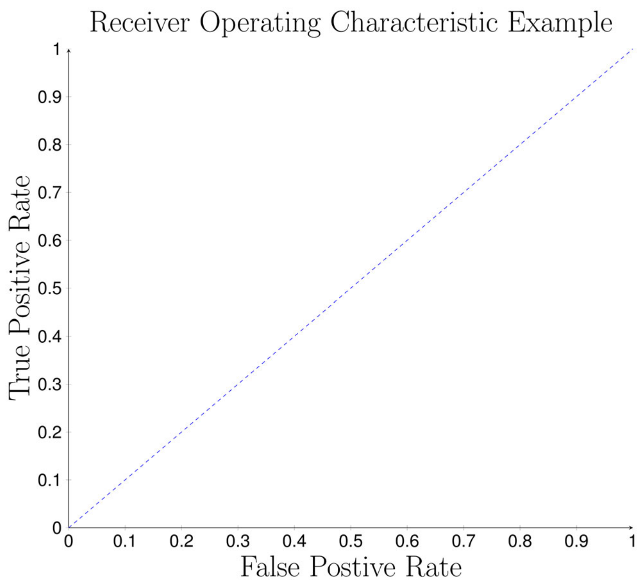

Figure 2. Area under the curve (AUC).

Lastly, all the results that follow in Tables 8–10 represent the accuracy, F1 and AUC based on the compound dataset for the MLP and Ensemble classifiers.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{11}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{12}$$

$$\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{13}$$

$$\text{True Positive Rate (Sensitivity)} = \frac{\text{TP}}{\text{FN} + \text{TP}} \tag{14}$$

$$\text{False Positive Rate (Specificity)} = \frac{\text{TP}}{\text{FP} + \text{TN}} \tag{15}$$

Table 11 provides a comparison between the mean values of the MLP and Ensemble classifiers for the Accuracy, F1 and AUC metrics respectively.

Tables 12–14 that follow provide a comparison for the Accuracy, F1 and AUC metrics, respectively, between the soft voting ensemble method used and a hard-voting method based on the same approach. The difference between soft voting and hard voting is that in the first 2 out of 3 need to be classified as attack or not but in hard voting all 3 need to be classified as attack or not.

Table 15 provides a comparison between the mean values of the Soft and Hard Ensemble classifiers for the Accuracy, F1 and AUC metrics, respectively.

Table 8. Accuracy results for the MLP and ensemble learning classifiers.

Execution	MLP	Ensemble
1	0.906	0.913
2	0.933	0.926
3	0.953	0.92
4	0.953	0.92
5	0.886	0.913
6	0.906	0.92
7	0.9	0.92
8	0.906	0.966
9	0.933	0.913
10	0.946	0.933
Mean	0.922	0.924

Table 9. F1 results for the MLP and Ensemble learning classifiers.

Execution	MLP	Ensemble
1	0.705	0.735
2	0.789	0.776
3	0.803	0.705
4	0.754	0.81
5	0.599	0.716
6	0.705	0.777
7	0.694	0.747
8	0.755	0.824
9	0.754	0.735
10	0.763	0.773
Mean	0.732	0.759

Table 10. AUC results for the MLP and ensemble learning classifiers.

Execution	MLP	Ensemble
1	0.659	0.687
2	0.732	0.718
3	0.766	0.678
4	0.682	0.774
5	0.579	0.676
6	0.665	0.739
7	0.655	0.708
8	0.713	0.805
9	0.707	0.715
10	0.705	0.726
Mean	0.686	0.722

Table 11. Comparisons between the MLP and ensemble classifiers.

(a) Accuracy comparison		(b) F1 comparison		(c) AUC comparison	
ALGORITHM	RESULT	ALGORITHM	RESULT	ALGORITHM	RESULT
MLP	0.922	MLP	0.732	MLP	0.686
Ensemble	0.924	Random Forest	0.759	Random Forest	0.722
Difference in terms of %	+0.25	Difference in terms of %	+3.7	Difference in terms of %	+5.3

5. Discussion

Initially, it is shown that a straightforward solution such as an MLP classifier provides high accuracy. Following that, a slightly more complicated method that involves the use of

Table 12. Accuracy results for the soft and hard voting ensemble learning classifiers.

Execution	Soft	Hard
1	0.913	0.913
2	0.926	0.933
3	0.92	0.94
4	0.92	0.913
5	0.913	0.946
6	0.92	0.926
7	0.92	0.893
8	0.966	0.926
9	0.913	0.946
10	0.933	0.926
Mean	0.924	0.926

Table 13. F1 results for the soft and hard voting Ensemble learning classifiers.

Execution	Soft	Hard
1	0.735	0.735
2	0.776	0.754
3	0.705	0.769
4	0.81	0.716
5	0.716	0.831
6	0.777	0.76
7	0.747	0.72
8	0.824	0.802
9	0.735	0.818
10	0.773	0.79
Mean	0.759	0.769

Table 14. AUC results for the soft and hard voting ensemble learning classifiers.

Execution	Soft	Hard
1	0.687	0.687
2	0.718	0.688
3	0.678	0.782
4	0.774	0.663
5	0.676	0.765
6	0.739	0.702
7	0.708	0.674
8	0.805	0.738
9	0.715	0.763
10	0.726	0.742
Mean	0.722	0.72

Table 15. Comparisons comparison between the soft and hard ensemble classifiers.

(a) Accuracy comparison		(b) F1 comparison		(c) AUC comparison	
ALGORITHM	RESULT	ALGORITHM	RESULT	ALGORITHM	RESULT
Soft	0.924	Soft	0.759	Soft	0.722
Hard	0.926	Hard	0.769	Hard	0.720
Difference in terms of %	−0.25	Difference in terms of %	−1.35	Difference in terms of %	+0.4

three classifiers, but not an MLP classifier within, using soft voting can provide similar accuracy. However, accuracy is not always the best method to evaluate classifier especially in cases where there is no balance between a yes or no classification label. Subsequently, this can lead to very accurate classifiers because most of the labels are either yes or no, despite of their complexity.

In terms of comparison of the results it is shown that both the MLP and the Ensemble Learning classifier provide similar accuracy, which can lead to a conclusion that any of the two can be used. Following that the F1 and AUC metrics have been used to further evaluate the classifiers which take into account the true positive and false positive rates. Moreover, we consider the AUC being slightly more important in this scenario because it is the possibility of correctly classifying correctly a randomly chosen entry. Thus, due to the result of the AUC being higher the soft voting approach has been selected against the hard voting approach, although one can argue that any of the two can be used due to the very similar results but both of these approaches are more accurate compared to the MLP classifier.

In terms of F1 and AUC however there are differences when the MLP is compared with the proposed Ensemble Learning classifier. In term of F1 there is a 3.7% difference and in terms of AUC there is a 5.3% difference. Moreover, the soft-voting method performs slightly better when compared to the hard-voting method in terms of AUC with 0.722 and 0.720, respectively, which is a 0.4% difference and the reason that it was selected against the hard voting.

Taking into consideration that the AUC is an important metric for classifying a dialogue as a potential social engineering attack or not we conclude to the fact the in online chat environments where a chat takes place between a random individual and a company employee it is vital to be as accurate as possible when it comes to social engineering. Companies nowadays provide education about such matters to employees, but it never harms to be extra cautious with a supporting online tool which can help identifying social engineering attacks.

The AUC of the proposed soft-voting Ensemble Learning method is better from both the MLP classifier and the alternative hard-voting method. Thus, making it the best solution for real environments where social engineering attack identification is essential.

Additionally, this research can also contribute to the currently existing models for social engineering attack detection, which were discussed in the related work section. Both the SEADM and MPMPA utilize state machine logic, that relies on human intervention for detection. This research can plug into these models by supplementing the human reasoning component with additional information that can be utilized as additional inputs. This allows the already existing models to improve, by incorporating this research, and ultimately this leads to improved social engineering attack detection models within the domain. This research can also be used as a baseline to determine whether the existing models can be even further automated utilizing natural language processing and artificial neural networks. Adding this research onto the existing models will aid to eliminate user error or naivety which can be introduced due to the user input into the models.

6. Conclusions

In this paper we presented a novel method for social engineering attack detection based on natural language processing and artificial neural networks. The method initially processes a dialogue and then creates a dataset that can be used for classification purposes. The proposed method has been evaluated using a real and a semi-synthetic dataset and can detect social engineering attacks with very high accuracy. Furthermore, alternative classification methods have been used for comparison in order to show the effectiveness of our method. In the future we plan to collect more data and both in terms of entries and by expanding the number of attributes necessary. Furthermore, we aim to develop a more balanced dataset and execute the algorithm in a real environment. The results will indicate if any new algorithms might be necessary to improve detection.

Note

1. The datasets can be found at https://github.com/npolatidis/seader

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Mr Merton Lansley is currently undertaking an MSc degree in Intelligent and Adaptive systems at the University of Sussex and holds a BSc (Hons) in Computer Science from the University of Brighton. He has previously published in Springer's ICCCI 2019: Computational Collective Intelligence conference. He is interested in the applications of machine learning and natural language processing in the cyber security domain.

Dr. Francois Mouton is an Associate Professor in Cyber Security at Noroff University College based in Oslo, Norway. His fields of expertise are social engineering, penetration testing and digital forensics. Francois graduated with a PhD Computer Science, in the field of social engineering, from the University of Pretoria in 2018. The focus of his PhD was on techniques on identifying and thwarting social engineering attacks by altering the human psyche. During his academic career he has also completed all the undergraduate modules provided at University of Pretoria for both psychology and accounting. He has (co)authored several international publications, mainly on topics of digital forensics readiness and social engineering.

Dr Stelios Kapetanakis is a Principal Lecturer in Business intelligence and Enterprise in the School of Computing, Engineering and Mathematics, University of Brighton. He has been the Head of Research and Development in Kare, Knowledgeware, a specialising in Natural Language processing. He has a PhD in Artificial Intelligence and an MBA in Knowledge and Innovation Management. For the past 12 years Dr Kapetanakis has been working in applied Artificial Intelligence projects with national and international vendors like the British Railways, Vivaki Nerve Centre, MediaCom, among others. His research interests include large scale Data Mining, Deep Learning, Case-based Reasoning and Spatial-Temporal reasoning.

Dr Nikolaos Polatidis since February 2019 is a Senior Lecturer in the School of Computing, Engineering and Mathematics at the University of Brighton. Prior to this, he was a Lecturer from May 2018 to January 2019 and before that a Research Fellow from March 2017 to April 2018. He has published more than 30 peer reviewed articles, served the wider community via various chairing roles and committee memberships for conferences and workshops and by reviewing for a wide range of journals. His research interests include the areas of: Artificial Intelligence, Machine Learning and Cybersecurity.

ORCID

Francois Mouton http://orcid.org/0000-0001-8804-7601
Nikolaos Polatidis http://orcid.org/0000-0003-4249-4953

References

Abed-alguni, B. H. (2019). Island-based cuckoo search with highly disruptive polynomial mutation. *International Journal of Artificial Intelligence*, *17*(1), 57–82.

Bezuidenhout, M., Mouton, F., & Venter, H. S. (2010). Social engineering attack detection model: Seadm. In *2010 Information Security for South Africa* (pp. 1–8).

Bhakta, R., & Harris, I. G. (2015). Semantic analysis of dialogs to detect social engineering attacks. In *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)* (pp. 424–427).

Heartfield, R., & Loukas, G. (2018). Detecting semantic social engineering attacks with the weakest link: Implementation and empirical evaluation of a human-as-a-security-sensor framework. *Computers and Security*, *76*, 101–127. https://doi.org/https://doi.org/10.1016/j.cose.2018.02.020

Hoeschele, M., & Rogers, M. (2005). Detecting social engineering. In Pollitt, M. and Shenoi, S. (Eds.), *Advances in digital forensics* (pp. 67–77). Springer US.

Jamil, A., Asif, K., Ghulam, Z., Nazir, M. K., Mudassar Alam, S., & Ashraf, R. (2018). Mpmpa: A mitigation and prevention model for social engineering based phishing attacks on facebook. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 5040–5048).

Krombholz, K., Hobel, H., Huber, M., & Weippl, E. (2015). Advanced social engineering attacks. *Journal of Information Security and Applications*, *22*, 113–122. Special Issue on Security of Information and Networks. https://doi.org/10.1016/j.jisa.2014.09.005

Lansley, M., Polatidis, N., & Kapetanakis, S. (2019). Seader: A social engineering attack detection method based on natural language processing and artificial neural networks. In N. T. Nguyen, R. Chbeir, E. Exposito, P. Aniorté, and B. Trawiński (Eds.), *Computational collective intelligence* (pp. 686–696). Springer International Publishing.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations* (pp. 55–60).

Mouton, F., Leenen, L., & Venter, H. S. (2015). Social engineering attack detection model: Seadmv2. In *2015 International Conference on Cyberworlds (CW)* (pp. 216–223).

Mouton, F., Leenen, L., & Venter, H. S. (2016). Social engineering attack examples, templates and scenarios. *Computers and Security*, *59*, 186–209. https://doi.org/10.1016/j.cose.2016.03.004

Mouton, F., Nottingham, A., Leenen, L., & Venter, H. S. (2018). Finite state machine for the social engineering attack detection model: Seadm. *SAIEE Africa Research Journal*, *109*(2), 133–148. https://doi.org/10.23919/SAIEE.2018.8531953

Nguyen, N. T. (2016). An influence analysis of the inconsistency degree on the quality of collective knowledge for objective case. In *Asian conference on intelligent information and database systems* (pp. 23–32). Berlin: Springer.https://doi.org/10.1007/978-3-662-.

Nicholson, J., Coventry, L., & Briggs, P. (2017). Can we fight social engineering attacks by social means? Assessing social salience as a means to improve phish detection. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)* (pp. 285–298). USENIX Association.

Peng, T., Harris, I., & Sawa, Y. (2018). Detecting phishing attacks using natural language processing and machine learning. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)* (pp 300–301).

Precup, R.-E., & David, R. C. (2019). *Nature-inspired optimization algorithms for fuzzy controlled servo systems*. Butterworth-Heinemann.

Resnik, A. J. (1986). *Journal of Marketing Research*, *23*(3), 305–306.

Saadat Javad, P. M., & Koofigar, H. (2017). Training echo state neural network using harmony search algorithm *International Journal of Artificial Intelligence*, *15*(1), 163–179.

Sawa, Y., Bhakta, R., Harris, I. G., & Hadnagy, C. (2016). Detection of social engineering attacks through natural language processing of conversations. In *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)* (pp. 262–265).

Tsinganos, N., Sakellariou, G., Fouliras, P., & Mavridis, I. (2018). Towards an automated recognition system for chat-based social engineering attacks in enterprise environments. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, ARES 2018, (pp. 53:1–53:10). ACM.