Proceedings of the Workshop on Social Threats in Online Conversations: Understanding and Management (STOC-2020), pages 48–55
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

Email Threat Detection Using Distinct Neural Network Approaches

Esteban Castillo[1], Sreekar Dhaduvai[2], Peng Liu[2], Kartik-Singh Thakur[2], Adam Dalton[3] and Tomek Strzalkowski[1]
[1]Rensselaer Polytechnic Institute, Troy, NY, USA, {castie2, tomek}@rpi.edu
[2]State University of New York at Albany, Albany, NY, USA, {sdhaduvai, pliu3, kthakur}@albany.edu
[3]IHMC, Ocala, FL, USA, adalton@ihmc.us

Abstract

This paper describes different approaches to detect malicious content in email interactions through a combination of machine learning and natural language processing tools. Specifically, several neural network designs are tested on word embedding representations to detect suspicious messages and separate them from non-suspicious, benign email. The proposed approaches are trained and tested on distinct email collections, including datasets constructed from publicly available corpora (such as Enron, APWG, etc.) as well as several smaller, non-public datasets used in recent government evaluations. Experimental results show that back-propagation both with and without recurrent neural layers outperforms current state of the art techniques that include supervised learning algorithms with stylometric elements of texts as features. Our results also demonstrate that word embedding vectors are effective means for capturing certain aspects of text meaning that can be teased out through machine learning in non-linear/complex neural networks, in order to obtain highly accurate detection of malicious emails based on email text alone.

1. Introduction

Email messages are the dominant way of communication for many users around the world (Dada et al., 2019). Among the massive daily email traffic, unsolicited and unwanted messages[1] have become a growing nuisance and increasingly posing serious threats to users' privacy and security by distributing false information, deceptive requests, as well as malicious links and attachments.

A number of approaches have been used for detection and removal of malicious messages from email feeds (Mujtaba et al., 2017). For example, extraction of harmful content (payload) has solved many obvious problems, as did the analysis of email headers for sender addresses and delivery paths, but most of these techniques fail to understand the content of the message itself: does the message contain a request (explicit or implicit) for the addressee to perform an action that would harm them or their organization, e.g., by divulging private information? In other words, the message itself, and not necessarily any associated metadata, becomes a threat because it attempts to break the last line of defense: the user.

Given the challenging nature of the task, we propose a novel technique to identify suspicious emails based on the analysis of email textual content. *Our main contribution* is the evaluation of multiple neural network architectures applied to pre-trained word embedding representation to automatically acquire accurate indicators of malicious emails. *The papers main hypothesis* is that different non-linear models (neural networks architectures) can learn hidden correlations between text elements (represented as word embeddings) that are characteristic of malicious messages and do so more reliably than classic supervised learning approaches (bag of words, TD-IDF etc.). *Our motivation* is to create reliable content-based models that can classify email and other types of messages (such as SMS) as suspicious (spam, phishing, malware, propaganda, etc.) as a first line of defense against social engineering attacks (Sawa et al., 2016).

The remainder of this paper is organized as follows: in Section 2. current approaches to detection of suspicious email are reviewed. Sections 3. to 6. provide details of our design and implementation of the neural network architectures. In Section 7. the experimental results are presented and discussed. Finally, implications and conclusions derived from this work thus far are discussed in Section 8..

2. Related Work

In this section, we briefly review the most relevant recent work in the email analysis and classification, specifically those that use machine learning, highlighting their main features and performance.

(Diale et al., 2019) implemented a Support Vector Machine (SVM), Random Forest and decision tree algorithms for spam detection with a vector size reduction approach to eliminate excessive number of features. A distributed bag of words representation was used for fixed length embedding of email samples. Dimensionality reduction was utilized to capture word ordering and basic semantic meaning from text messages. Experimental results show an overall spam detection accuracy of 97% over the Enron dataset.

(Abu-Nimeh et al., 2007) compares distinct supervised learning algorithms (logistic regression, random forest, SVM, etc.) for detecting phishing emails. The approach considers a bag of words model as text representation with TF-IDF weights for detecting best features in the body of emails. Experimental results show an average accuracy of 92% over a manually annotated phishing dataset and identify the logistic regression and SVM algorithms as best options when text frequency distributions are analyzed.

(Abiodun et al., 2019) used a SVM and Naïve Bayes algorithm alongside a feature analysis process to detect phishing messages. Multiple content and header features were considered incrementally in order to find the optimal set of features that maximizes classification accuracy. Experimental

[1] Social engineering attacks, spam, phishing, malware, propaganda among others.

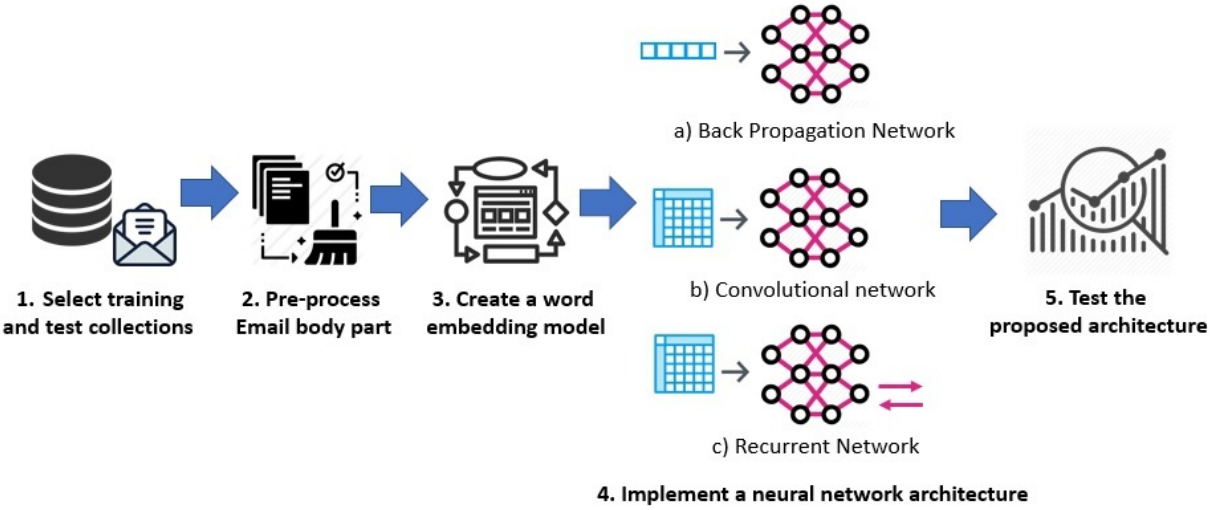


Figure 1: Email threat detection process.

results show accuracy around 98% for detecting phishing texts over messages that contain a verified set of phishing messages and URLs reported by volunteers (Alexa and PhishTank[2] collections).

(Varol and Abdulhadi, 2018) presented an heuristic spam filtering approach in which different string matching metrics (Levenshtein Edit-Distance, Longest Common Subsequence, etc.) are used to compare email text sentences over manually selected phrases related to spam and propaganda. Emails are labelled as friend or foe if most comparisons surpass a predefined numeric threshold. Experiments were run against the Enron and CSDMC2010 datasets showing spam detection accuracy around 98%.

(Bahgat et al., 2018) employed several supervised learning algorithms (SVM, Bayesian Logistic Regression) alongside an ontology and text similarity measures for detecting malicious email messages (spam and propaganda). The proposed approach employs the WordNet ontology to eliminate words with a similar meaning, then benign/non-benign emails are compared between each other using string matching measures where emails are label as foe if are similar to many malicious messages. Experimental results show an accuracy above 90% over the Enron dataset.

Finally, (M et al., 2018) describes a set of experiments for detecting phishing on emails by using a convolutional neural network with a word embedding approach over email headers or the messages itself (payload). The experiments obtained an accuracy around 96% which shows the importance of neural networks for detecting malicious attacks and demonstrate that word embeddings are a suitable for detecting fine-grained patterns of users writing style. Important to mentioned that this paper helps to see that a single model over spam, phishing, malware, etc. could be created by using word embedding and neural networks as platform.

Most of the above approaches work reasonably well, although some recent experiments using neural networks (Smadi et al., 2018; Roy et al., 2020) have been more successful, especially for generalizing models across different threat types and associated topics.

3. Threat Detection Process

In this section we discuss several variants of a new method for detecting multiple forms of malicious email that include phishing, spam, malware, propaganda, and also sophisticated forms of social engineering. Our method is tested on email, but it is general enough to apply to other types of messaging, including social media private messaging and chat channels. Figure 1 shows the overall approach with three alternative training modes with different neural network architectures. The process is explained below:

1. Select appropriate email collections for training and testing of the prediction models (see Section 4.).
2. Pre-process textual information in the body of emails. This task includes word tokenization, elimination of punctuation and special symbols, and converting all text to lowercase.
3. Create word embedding models taking as input all training email collections (see Section 5.).
4. Implement a neural network architecture that takes word embeddings obtained in the previous step as input and learns to classify emails into friend/foe or (in future experiments) more categories (e.g., friend, foe, undecided; as well as subtypes of foe messages) (see Section 6.).
5. Evaluate the trained models using set-aside test collections (see Section 7.).

4. Datasets Used

The document collections used for training and testing include benign and malicious email samples obtained from employees of public companies and government departments.

Benign emails correspond to internal interactions among users on day-to-day work issues. On the other hand, most of

[2] https://www.phishtank.com/

suspicious emails are obtained from employees spam boxes and specific email threat repositories (like APWG dataset). All emails have been manually labeled at source following the conventions of the data providers. For training purposes we converted these into binary suspicious/non-suspicious labels, but also kept the original labels as additional features (threat type).

In this application we only consider the (textual) body of emails; header and other metadata was not used.[3]. We also note that all personally identifiable information (PII) has been removed or replaced in the data. Attachments are kept in most cases, but these containing malware are eliminated to protect users.

Table 1 summarizes the key details of each collection. Enron and APWG (among other collections) are used for training purposes while Non-public datasets called dry-run 1 and dry-run 2 are used for testing.

Dataset name and/or type	Feature	Training	Testing
Enron	Used for word embeddings	✓	
(Klimt and Yang, 2004)	Collection type	Public available	
Benign emails	Number of emails	84111	NA
APWG (Oest et al., 2018)	Used for word embeddings	✓	
Phishing/Malware	Collection type	Public available	
Non-benign emails	Number of emails	30776	NA
BC3	Used for word embeddings	✓	
(Ulrich et al., 2008)	Collection type	Public available	
Benign emails	Number of emails	259	NA
Phishing[4]/non-phishing	Used for word embeddings	✓	
	Collection type	Non-public available	
Non-benign emails	Number of emails	5338	NA
Malware[5]/non-malware	Used for word embeddings	✓	
	Collection type	Non-public available	
Non-benign emails	Number of emails	2914	NA
Propaganda[6]	Used for word embeddings	✓	
/non-propaganda	Collection type	Non-public available	
Non-benign emails	Number of emails	261	NA
Spam[7]/non-spam	Used for word embeddings	✓	
	Collection type	Non-public available	
Non-benign emails	Number of emails	1294	NA
social engineering[8]	Used for word embeddings	✓	
/non-social engineering	Collection type	Non-public available	
Non-benign emails	Number of emails	1059	NA
Reconnaissance[9]	Used for word embeddings	✓	
/non-reconnaissance	Collection type	Non-public available	
Non-benign emails	Number of emails	173	NA
Dry-run 1	Used for word embeddings	✗	
Benign and	Collection type	Non-public available	
Non-benign emails	Number of emails	NA	1025
Dry-run 2	Used for word embeddings	✗	
Benign and	Collection type	Non-public available	
Non-benign emails	Number of emails	NA	3023

Table 1: Datasets used in this study.

From above table, it is important to highlight that dry-run datasets comprise also email samples of day-to-day interactions in a work environment. This collections are non public available considering that there are utilize for evaluating an active social engineering program of the USA government. Despite that, it can be mentioned that this datasets have an unbalanced nature with a proportion of 80% benign samples and 20% non-benign ones which is consistent with a real world scenario.

[3] Header information included in the emails is not always complete due to privacy considerations.

[4] Email messages often used to steal users data.

[5] Emails embedded code designed to cause extensive damage to users data/systems.

[6] Email sent to disseminate facts, arguments, rumours related to a specific topic.

[7] Unsolicited, undesired or annoying email messages.

[8] Email message used for manipulate users, so they give up confidential information voluntarily.

[9] Email sent to gain preliminary information about a potential victim.

5. Word Embeddings

Accurate detection of suspicious emails in the stream of daily messages, based on email content alone, requires attention to subtle differences in word use, sequencing, and the "tone" of the message. Unlike most ordinary communication, malicious messages attempt to produce a reaction from the recipient in a manner that tends to violate communication norms – the subtleties that we are attempting to tease out.

Word embeddings (Mikolov et al., 2013; Bengio et al., 2006) which capture contextual meaning of words in texts by creating vector representations, are particularly suitable for this task. We derive word embeddings from a corpus of emails, thus capturing what we believe are the contextual meanings of words use in email genre.

In this paper, a continuous bag-of-word model based on Gensim-word2vec (Řehůřek and Sojka, 2010) is utilized for obtaining numerical vectors of words. We use a window of 10 words for analyzing the neighborhood of texts and vectors of different size are created for testing different neural networks architectures (see Section 7.1.).

In the next section we explain the role of word embedding vectors as inputs to a supervised learning algorithm implemented with different neural network architectures.

6. Neural Networks Architectures

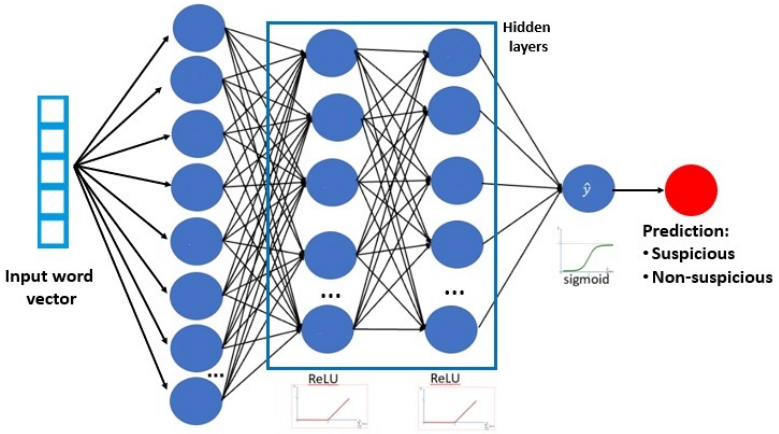
Neural networks (Goodfellow et al., 2016) are a special type of classifiers which are strongly tied to supervised machine learning suitable for modeling of non-linear problems.

Figure 2 shows the three neural network architectures proposed for training classifiers for suspicious/non-suspicious emails. All variants are implemented in Keras (Chollet, 2017) as follows:

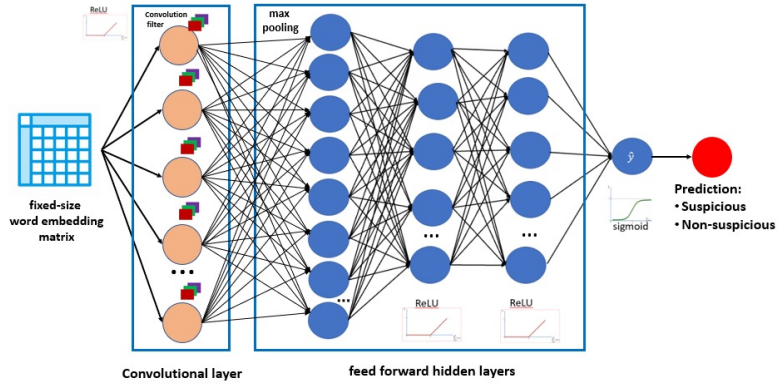
1. **Back-propagation network**: A classic feed-forward network which creates multiple hidden layers between input and output elements. This architecture adjusts model efficiency according to a gradient descent technique (Ruder, 2016) which minimizes the error rate after multiple back and forth iterations over the network on training samples.

   Figure 3a highlights how the word embeddings are utilized as input in the back-propagation process. For each training and test emails, content word embedding vectors are combined into a single vector by computing averages across corresponding dimensions. The objective is to obtain a vector representing the meaning of each email.

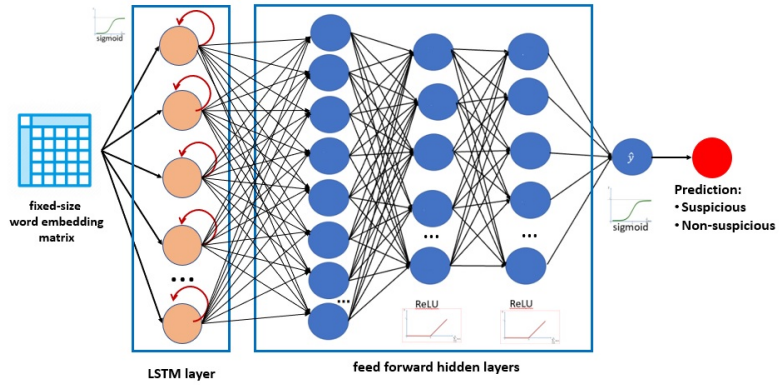
2. **Convolutional network**: A specialization of the back-propagation model (Indolia et al., 2018) that employs mathematical transformations (convolutions)



(a) Back-propagation network.



(b) Convolutional network.



(c) Recurrent network (LSTM).

Figure 2: Proposed neural network architectures.

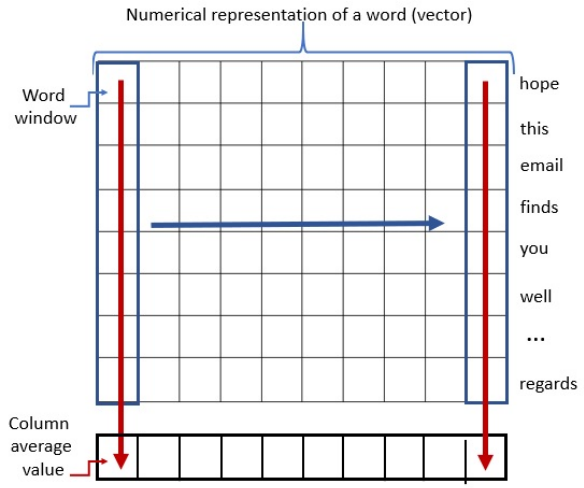
over specific hidden layers for detecting fine grained features. The combined new features (max pooling), help capturing patterns related to order and proximity over the words, increasing the detection of spatio-temporal aspects of original texts.

Figure 3b shows how the embeddings are used in this architecture. A fixed-size matrix is created for each training and test email taking as input tokens from text. In this matrix, columns represent features of an embedding vector and rows represent tokens associated with email samples.

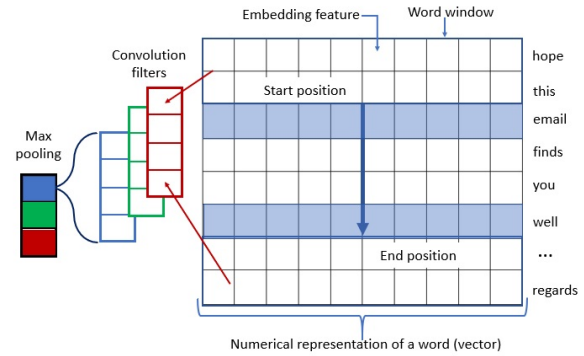
It is important to note that this type of architecture requires matrices of the same size. Accordingly, we take the first N words from each email as input to the process[10].

3. **Recurrent neural network**: Another specialization of back-propagation model (Hochreiter and Schmidhuber, 1997; Soutner and Müller, 2013) where data sequences are analyzed in order to predict new ones based on prior knowledge. In this architecture, specific hidden layers implement neuron loops allowing a small memory state where previous words are used as input to current word analysis, this help to relate token patterns that are syntactically separate in the word

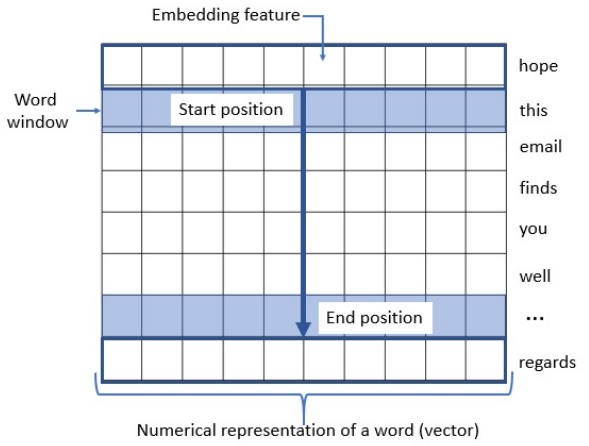
[10] If an email is shorter than N, all its words are used as rows in a matrix and the remaining positions are padded with zeros.



(a) Back-propagation network.



(b) Convolutional network.



(c) LSTM network.

Figure 3: Word embedding matrix representations.

sequence.

In this paper, we used a specific type of recurrent network called Long Short-Term Memory network (LSTM). This kind of network expands the idea of a memory state by creating a complex architecture of nodes that remember information of correlated elements that are far away (key difference from a classical recurrent network). LSTM networks analyze and predict information considering past knowledge, which most of the time is omitted or managed independently introducing some bias to the learning algorithms.

Figure 3c show how the LSTM network takes consecutive word windows in order to analyze past, present and future words in the email text. As with the convolutional network, LSTMs require inputs of the same size, therefore only the first N email words are used as input in the network.

7. Experimental Results

Results obtained using the proposed neural network architectures are discussed in this section. First, the experiments are described, then the results are shown, and followed by a discussion of the findings.

7.1. Experiments Performed

A series of experiments were performed in order to test the accuracy of the proposed variants. In total, 60480 experimental runs were performed using multiple combinations of neural network parameters for each architecture type. Table 2 summarizes the different configurations that were tested in a **supervised learning** fashion.

Parameter	Possible values
Programming language	Python https://www.python.org/
Neural network package	Keras https://keras.io/
Word embedding package	Gensim https://radimrehurek.com/gensim/
Email words/tokens (Matrix rows)	$10, 20, 30, 40$
Word embedding features (Matrix columns)	$30, 40, 50, 60$
Convolutional layer (number of neurons)	$20, 30, 40, 50$
LSTM layer (number of neurons)	$30, 40, 50, 60$
Back-propagation neurons (Multiple hidden layers)	50-50,32-16-8-3,8-8,4-4,3-3-3
Convolutional word window	3
LSTM word window	2
Convolutional network filter number	100
Convolutional activation function	Relu
LSTM activation function	Sigmoid
Back-propagation activation function	Relu and Sigmoid
Batches	$50, 60, 70, 80, 90, 100, 110$
Epochs	$2, 3, 4, 5, 6, 7$

Table 2: Experimental parameters.

It is worth noting that parameters were selected according to a preliminary experimentation and the recommendations from relevant literature (Lane et al., 2019).

7.2. Experimental Results

Table 3 summarizes the results over dry-run 1 and dry-run 2 test collections. Experimental results obtained from variants of NN architecture, as explained above, are compared against traditional classifiers (SVM, NB, and LR) that use

Dataset type	Approach	Features (Matrix colums)	Email length (Matrix rows)	LSTM Neurons	Convolutional Neurons	Back-Propagation Neurons	Batches	Epochs	Accuracy
	BP	60	40	–	–	8-8	70	6	**0.9568***
	LSTM	50	40	50	–	32-16-8-3	90	7	0.9317
	BP	50	50	–	–	4-4	100	6	0.9127
	CN	20	40	–	40	8-8	50	4	0.9175
	LSTM	30	50	60	–	4-4	110	6	0.9114
	BP	40	40	–	–	8-8	90	5	0.9031
Dry-Run 1	BP	30	60	–	–	8-8	50	4	0.9012
	LSTM	40	40	50	–	32-16-8-3	60	6	0.8855
	LSTM	40	50	60	–	50-50	90	7	0.8821
	CN	20	30	–	40	3-3-3	60	4	0.8793
	SVM	–	–	–	–	–	–	–	0.8137
	NB	–	–	–	–	–	–	–	0.7915
	LR	–	–	–	–	–	–	–	0.7824
	LSTM	30	40	60	–	3-3-3	60	5	**0.9185***
	BP	40	60	–	–	8-8	70	7	0.9136
	BP	30	60	–	–	3-3-3	80	5	0.9023
	BP	40	40	–	–	4-4	70	6	0.9012
	CN	20	30	–	40	50-50	50	3	0.8983
	BP	30	40	–	–	32-16-8-3	70	4	0.8839
Dry-Run 2	CN	30	30	–	40	32-16-8-3	50	3	0.8748
	LSTM	40	50	60	–	8-8	100	6	0.8612
	SVM	–	–	–	–	–	–	–	0.8529
	BP	40	40	–	–	50-50	110	6	0.8512
	BP	30	30	–	–	3-3-3	80	5	0.8507
	LR	–	–	–	–	–	–	–	0.8045
	NB	–	–	–	–	–	–	–	0.7749

BP: Back-propagation
CN: Convolutional Network
LSTM: Long Short-Term Memory Network (recurrent network)
SVM: Support Vector Machine
NB: Naïve Bayes
LR: Logistic Regression

Table 3: Summary of best experimental results.

standard bag of words approach[11] (Manning et al., 2008; Sarah Guido, 2016).

Experimental results demonstrate that the performance of all neural network variants surpasses baseline techniques using three main options with customized word embeddings over dry-run 1 and 2: back-propagation accuracy (95%, 91%), convolutional network accuracy (91%, 89%) and LSTM network accuracy (93%, 91%).

The best results (independently of the test collection) were obtained using the first 30 to 40 words of each email, which indicates that the core threat information is included in this range. For the word embeddings size, relatively small vectors of 40 to 60 features appear to pack the semantic content of emails to capture the distinctive indicators associated with email intent (malicious or benign).

The major advantage of the LSTM and convolutional networks is the fine grained analysis of word sequences, which helps to identify subtle textual patterns that are lost when each word is considered independently. On the other hand, the major disadvantage is the extra amount of time and resources required for training for relatively modest performance gain.

Finally, the obtained results demonstrate that the proposed neural network configurations can be used to effectively train accurate classifiers for detecting suspicious emails independently of their topic and subject domain. This fact highlights the relevance of the neural networks created and the features used as an effective method of capturing the intent of emails.

[11] Baseline experiments were implemented using the whole set of words on the training collections as features and the default parameters for the scikit-learn package for classifiers.

8. Conclusions and Future Work

Several neural network architectures were tested in order to train effective classifiers for identification of malicious content in email messages, independently of their subject matter. The results demonstrate viability of the proposed methods for capturing malicious intent in messages. Our key findings are summarized below:

1. *Datasets* selected for this project are shown to be relevant for training and testing the email classifiers. Taken together, they provide sufficient lexical and syntactic resources for effective learning of textual patterns related to malicious messages.

2. *Word embeddings* proved to be an adequate option for representing the writing style of malicious emails. This technique helped to capture the context of words in email messages as well as their relationship with other words which ultimately lead to an accurate representation.

3. The *back-propagation network* obtained the best results compared with other approaches. This highlights the ability of the model to learn non-linear and complex relationships between inputs and outputs. Results

obtained also demonstrate that the analysis of dense feed forward networks helps to generalize textual patterns across multiple email threat types.

4. The *convolutional networks* achieved a performance slightly below baseline results. This reveals that convolutions over word embeddings windows of size up to three are not enough to fully capture the writing style of malicious email messages. A higher accuracy could be obtained if complete word embedding windows are analyzed although it would consume a higher number of computational resources due to the complexity of the network.

5. The *recurrent neural networks* obtained similar results than baseline techniques. The results demonstrate that the analysis of words that are dependent of previous ones is not crucial for detecting suspicious email messages independently of the threat type.

6. Overall, our results surpass baseline techniques showing the relevance of the neural networks approach combined with word embeddings for detecting distinctive elements on email exchanges. Results also show the effectiveness of the networks in a real world application (Non-public datasets) where emails could not be related to the topics and domains used for training.

Future work includes the following actions:

- Run related work methods (see Section 2.) against test collections used in this paper (dry-run 1 and 2). The idea is to compare state of the art accuracy associated to a specific threat type (spam, or phishing) with the results obtained in this paper where classifier created can deal with different types of threats.

- Add more public email samples that deal with the problem in order to enrich the training process over the neural network architectures that show a better performance.

- Refine and enrich existing training datasets by careful manual labelling by multiple annotators.

- Apply additional neural networks techniques (Chollet, 2017; Lane et al., 2019) for improving the approach accuracy.

- Evaluate the proposed approach on different genre of messaging, e.g. social media channels (Inuwa-Dutse et al., 2018).

- Apply this approach in other languages (e.g., Spanish) keeping the same neural network parameters but creating a new word embedding model according to the language vocabulary.

Acknowledgments

Supported by DARPA FA8650-18-C-7881 and by DARPA through Army Contract W31P4Q-17-C-0066. All statements are those of the authors, not AFRL, DARPA, Army, or USG.

9. Bibliographical References

Abiodun, O., Sodiya, A., and Akinwale, A. T. (2019). A predictive model for phishing detection. *Journal of King Saud University - Computer and Information Sciences*, 1:1–16.

Abu-Nimeh, S., Nappa, D., Wang, X., and Nair, S. (2007). A comparison of machine learning techniques for phishing detection. In *Proceedings of the Anti-Phishing Working Groups 2nd Annual ECrime Researchers Summit*, page 60–69. Association for Computing Machinery.

Bahgat, E. M., Rady, S., Gad, W., and Moawa, I. F. (2018). Efficient email classification approach based on semantic methods. *Ain Shams Engineering Journal*, 9(4):3259 – 3269.

Bengio, Y., Schwenk, H., Senécal, J.-S., Morin, F., and Gauvain, J.-L., (2006). *Neural Probabilistic Language Models*, pages 137–186. Springer Berlin Heidelberg, Berlin, Heidelberg.

Chollet, F., (2017). *Deep Learning with Python*, pages 178–232. Manning Publications Co., Shelter Island, NY 11964.

Dada, E. G., Bassi, J. S., Chiroma, H., Abdulhamid, S. M., Adetunmbi, A. O., and Ajibuwa, O. E. (2019). Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, 5(6):1–23.

Diale, M., Celik, T., and Walt, C. V. D. (2019). Unsupervised feature learning for spam email filtering. *Computers & Electrical Engineering*, 74:89–104.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, Massachusetts.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Indolia, S., Goswami, A. K., Mishra, S. P., and Asopa, P. (2018). Conceptual understanding of convolutional neural network- a deep learning approach. *Procedia Computer Science*, 132:679–688.

Inuwa-Dutse, I., Liptrott, M., and Korkontzelos, I. (2018). Detection of spam-posting accounts on twitter. *Neurocomputing*, 315:496–511.

Klimt, B. and Yang, Y. (2004). The enron corpus: A new dataset for email classification research. In *Machine Learning: ECML 2004*, pages 217–226. Springer Berlin Heidelberg.

Lane, H., Howard, C., and Hapke, H., (2019). *Natural Language Processing in Action*, pages 247–273. Manning Publications Co., Shelter Island, NY 11964.

M, H., Unnithan, N. A., R, V., and Kp, S. (2018). Deep learning based phishing e-mail detection cen-deepspam. In *1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal.(IWSPA)*, pages 1–5.

Manning, C. D., Raghavan, P., and Schütze, H., (2008). *Introduction to Information Retrieval*, pages 18–43. Cambridge University Press.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, pages 3111–3119. Curran Associates Inc.

Mujtaba, G., Shuib, L., Raj, R. G., Majeed, N., and Al-Garadi, M. A. (2017). Email classification research trends: Review and open issues. *IEEE Access*, 5:9044–9064.

Oest, A., Safei, Y., Doupe, A., Ahn, G.-J., Wardman, B., and Warner, G. (2018). Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis. In *Proceedings of the 2018 APWG Symposium on Electronic Crime Research, eCrime 2018*, pages 1–12. IEEE Computer Society.

Řehůřek, R. and Sojka, P. (2010). Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworksg*, pages 45–50. ELRA.

Roy, P. K., Singh, J. P., and Banerjee, S. (2020). Deep learning to filter sms spam. *Future Generation Computer Systems*, 102:524–533.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747:1–14.

Sarah Guido, A. M., (2016). *Introduction to Machine Learning with Python*, pages 27–129. O'Reilly Media.

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

Smadi, S., Aslam, N., and Zhang, L. (2018). Detection of online phishing email using dynamic evolving neural network based on reinforcement learning. *Decision Support Systems*, 107:88–102.

Soutner, D. and Müller, L. (2013). Application of lstm neural networks in language modelling. In *Text, Speech, and Dialogue*, pages 105–112. Springer Berlin Heidelberg.

Ulrich, J., Murray, G., , and Carenini, G. (2008). A publicly available annotated corpus for supervised email summarization. In *AAAI08 EMAIL Workshop*, pages 1–6. AAAI.

Varol, C. and Abdulhadi, H. M. T. (2018). Comparision of string matching algorithms on spam email detection. In *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, pages 6–11.