A Deep Learning Model with Hierarchical LSTMs and Supervised Attention for Anti-Phishing

Minh Nguyen
Hanoi University of Science and Technology
Hanoi, Vietnam
minh.nv142950@sis.hust.edu.vn

Toan Nguyen
New York University
Brooklyn, New York, USA
toan.v.nguyen@nyu.edu

Thien Huu Nguyen
University of Oregon
Eugene, Oregon, USA
thien@cs.uoregon.edu

Abstract

Anti-phishing aims to detect phishing content/documents in a pool of textual data. This is an important problem in cybersecurity that can help to guard users from fraudulent information. Natural language processing (NLP) offers a natural solution for this problem as it is capable of analyzing the textual content to perform intelligent recognition. In this work, we investigate the state-of-the-art techniques for text categorization in NLP to address the problem of anti-phishing for emails (i.e, predicting if an email is phishing or not). These techniques are based on deep learning models that have attracted much attention from the community recently. In particular, we present a framework with hierarchical long short-term memory networks (H-LSTMs) and attention mechanisms to model the emails simultaneously at the word and the sentence level. Our expectation is to produce an effective model for anti-phishing and demonstrate the effectiveness of deep learning for problems in cybersecurity.

1 Introduction

Despite being one of the oldest tactics, email phishing remains the most common attack used by cybercriminals [phi17a] due to its effectiveness. Phishing attacks exploit users' inability to distinguish between legitimate information from fake ones sent to them [DTH06, SNM15, SNM17, SNG$^+$17]. In an email phishing campaign, attackers send emails appearing to be from well-known enterprises or organizations directly to their victims or by spoofed emails [Sin05]. These emails try to lure victims to divulge their private information [JJJM07, SNM15, SNG$^+$17] or to visit an impersonated site (i.e., a fake banking website), on which they will be asked for passwords, credit card numbers or other sensitive information. The recent hack of a high profile US politician (usually referred as "John Podesta's hack") is a famous example of this type of attack. It was all started by a spoofed email sent to the victim asking him to reset his Gmail password by clicking on a link in the email [pod16]. The technique of email phishing may seem simple, yet the damage it makes is huge. In the US alone, the estimated cost of phishing emails to business is half a billion dollars per year [phi17b].

Numerous methods have been proposed to automatically detect phishing emails [BCP$^+$08, FST07, ANNWN07, GTJA17]. Chandrasekaran et. al proposed to use structural properties of emails and Support Vector Machines (SVM) to classify phishing emails [CNU06]. In [ANNWN07], Abu-Nimeh et. al evaluated six machine learning classifiers on a public phishing email dataset using proposed 43 features. Gupta et. al [GTJA17] presented a nice survey on recent state-of-the-art research on phishing detection.

In: R. Verma, A. Das. (eds.): Proceedings of the 1st Anti-Phishing Shared Pilot at 4th ACM International Workshop on Security and Privacy Analytics (IWSPA 2018), Tempe, Arizona, USA, 21-03-2018, published at http://ceur-ws.org

However, these methods mainly rely on feature engineering efforts to generate characteristics (features) to represent emails, over which machine learning methods can be applied to perform the task. Such feature engineering is often done manually and still requires much labor and domain expertise. This has hindered the portability of the systems to new domains and limited the performance of the current systems.

In order to overcome this problem, our work focuses on deep learning techniques to solve the problem of phishing email detection. The major benefit of deep learning is its ability to automatically induce effective and task-specific representations from data that can be used as features to recognize phishing emails. As deep learning has been shown to achieve state-of-the-art performance for many natural language processing tasks, including text categorization [GBB11, LXLZ15], information extraction [NG15b, NG15a, NG16], machine translation [BCB14], among others, we expect that it would also help to build effective systems for phishing email detection.

We present a new deep learning model to solve the problem of email phishing prediction using hierarchical long short-term memory networks (H-LSTMs) augmented with supervised attention technique. In the hierarchical LSTM model [YYD$^+$16], emails are considered as hierarchical architectures with words in the lower level (the word level) and sentences in the upper level (the sentence level). LSTM models are first implemented in the word level whose results are passed to LSTM models in the sentence level to generate a representation vector for the entire email. The outputs of the LSTM models in the two levels are combined using the attention mechanism [BCB14] that assigns contribution weights to the words and sentences in the emails. A header network is also integrated to model the headers of the emails if they are available. In addition, we propose a novel technique to supervise the attention mechanism [MWI16, LUF$^+$16, LCLZ17] at the word level of the hierarchical LSTMs based on the appearance rank of the words in the vocabulary. Experiments on the datasets for phishing email detection in the First Security and Privacy Analytics Anti-Phishing Shared Task (IWSPA-AP 2018) [EDB$^+$18] demonstrate the benefits of the proposed models, being ranked among the top positions among the participating systems of the shared task (in term of the performance on the unseen test data).

2 Related Work

Phishing email detection is a classic problem; however, research on this topic often has the same limitation: there is no official and big data set for it. Most previous works typically used a public set consists of legitimate or "ham" emails[1] and another public set of phishing emails[2] for their classification evaluation[FST07, BCP$^+$08, BMS08, HA11, ZY12]. Other works used private but small data sets[CNU06, ANNWN07]. In addition, the ratio between phishing and legitimate emails in these data sets was typically balanced. This is not the case in the real-world scenario where the number of legitimate emails is much larger than that of phishing emails. Our current work relies on larger data sets with unbalanced distributions of phishing and legitimate emails collected for the the First Security and Privacy Analytics Anti-Phishing Shared Task (IWSPA-AP 2018) [EDB$^+$18].

Besides the limitation of small data sets, the previous work has extensively relied on feature engineering to manually find representative features for the problem. Apart from features extracted from emails, [LV12] also uses a blacklist of phishing webistes to get an additional feature for urls appearing in emails. Some neural network systems are also introduced to detect such blacklists [MTM14, MAS$^+$11]. This is undesirable because these engineered features need to be updated once new types of phishing emails with new content are presented. Our work differs from the previous work in this area in that we automate the feature engineering process using a deep learning model. This allows us to automatically learn effective features for phishing email detection from data. Deep learning has recently been employed for feature extraction with success on many natural language processing problems [NG15b, NG15a].

3 Proposed Model

Phishing email detection is a binary classification problem that can be formalized as follow.

Let $e = \{b, s\}$ be an email in which b and s are the body content and header of the email respectively. Let y the binary variable to indicate whether e is a phishing email or not ($y = 1$ if e is a phishing email and $y = 0$ otherwise). In order to predict the legitimacy of the email, our goal is to estimate the probability $P(y = 1|e) = P(y = 1|b, s)$. In the following, we will describe our methods to model the body b and header s with the body network and header network respectively to achieve this goal.

3.1 Body Network with Hierarchical LSTMs

For the body b, we view it as a sequence of sentences $b = (u_1, u_2, \ldots, u_L)$ where u_i is the i-th sentence and L is the number of sentences in the email body b. Each sentence u_i is in turn a sequence of words/tokens $u_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,K})$ with $v_{i,j}$ as the j-th token in u_i and

[1] `https://spamassassin.apache.org/old/publiccorpus`

[2] `https://monkey.org/~jose/phishing`

K as the length of the sentence. Note that we set L and K to the fixed values by padding the sentences u_i and the body b with dummy symbols.

As there are two levels of information in b (i.e, the word level with the words $v_{i,j}$ and the sentence level with the sentence u_i), we consider a hierarchical network that involves two layers of bidirectional long short-term memory networks (LSTMs) to model such information. In particular, the first layer consumes the words in the sentences via the embedding module, the bidirectional LSTM module and the attention module to obtain representation vectors for every sentence u_i in b (the word level layer). Afterward, the second network layer combines the representation vectors from the first layer with another bidirectional LSTM and attention module, leading to a representation vector for the whole body email b (the sentence level layer). This body representation vector would then be used as features to estimate $P(y|b, s)$ and make the prediction for the initial email e.

3.1.1 The Word Level Layer

Embedding

In the word level layer, every word $v_{i,j}$ in each sentence u_i in b is first transformed into its embedding vector $w_{i,j}$. In this paper, $w_{i,j}$ is retrieved by taking the corresponding column vector in the word embedding matrix W_e [MSC+13] that has been pre-trained from a large corpus: $w_{i,j} = W_e[v_{i,j}]$ (each column in the matrix W_e corresponds to a word in the vocabulary). As the result of this embedding step, every sentence $u_i = (v_{i,1}, v_{i,2}, \dots, v_{i,K})$ in b would be converted into a sequence of vectors $(w_{i,1}, w_{i,2}, \dots, w_{i,K})$, constituting the inputs for the bidirectional LSTM model in the next step.

Bidirectional LSTMs for the word level

This module employs two LSTMs [HS97, GS05] that run over each input vector sequence $(w_{i,1}, w_{i,2}, \dots, w_{i,K})$ via two different directions, i.e, forward (from $w_{i,1}$ to $w_{i,K}$) and backward (from $w_{i,K}$ to $w_{i,1}$). Along with their operations, the forward LSTM generates the forward hidden vector sequence $(\overrightarrow{h_{i,1}}, \overrightarrow{h_{i,2}}, \dots, \overrightarrow{h_{i,K}})$ while the backward LSTM produce the backward hidden vector sequence $(\overleftarrow{h_{i,1}}, \overleftarrow{h_{i,2}}, \dots, \overleftarrow{h_{i,K}})$. These two hidden vector sequences are then concatenated at each position, resulting in the new hidden vector sequence $(h_{i,1}, h_{i,2}, \dots, h_{i,K})$ for the sentence u_i in b where $h_{i,j} = [\overrightarrow{h_{i,j}}, \overleftarrow{h_{i,j}}]$. The notable characteristics of the hidden vector $h_{i,j}$ is that it encodes the context information over the whole sentence u_i due to the recurrent property of the forward and backward LSTMs although a greater focus is put at the current word $v_{i,j}$.

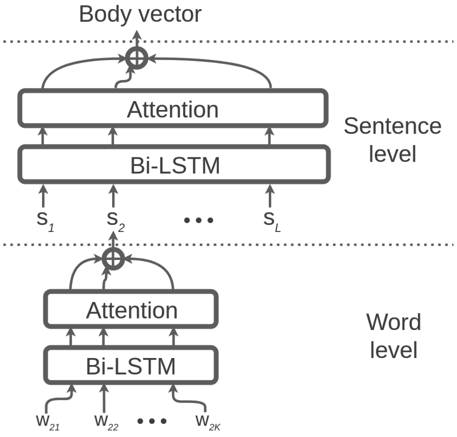


Figure 1: Hierarchical LSTMs.

Attention

In this module, the vectors in the hidden vector sequence $(h_{i,1}, h_{i,j,2}, \dots, h_{i,K})$ are combined to generate a single representation vector for the initial sentence u_i. The attention mechanism [YYD+16] seeks to do this by computing a weighted sum of the vectors in the sequence. Each hidden vector $h_{i,j}$ would be assigned to a weight $\alpha_{i,j}$ to estimate its importance/contribution in the representation vector for u_i for the phishing prediction of the email e. In this work, the weight $\alpha_{i,j}$ for $h_{i,j}$ is computed by:

$$\alpha_{i,j} = \frac{\exp(a_{i,j}^{\top} w_a)}{\sum_{j'} \exp(a_{i,j'}^{\top} w_a)} \tag{1}$$

in which

$$a_{i,j} = \tanh(W_{att} h_{i,j} + b_{att}) \tag{2}$$

Here, W_{att}, b_{att} and w_a are the model parameters that would be learnt during the training process. Consequently, the representation vector $\hat{u}_i$ for the sentence u_i in b would be:

$$\hat{u}_i = \sum_j \alpha_{i,j} h_{i,j} \tag{3}$$

After the word level layer completes its operation on every sentence of $b = (u_1, u_2, \dots, u_L)$, we obtain a corresponding sequence of sentence representation vectors $(\hat{u}_1, \hat{u}_2, \dots, \hat{u}_L)$. This vector sequence would be combined in the next sentence level layer to generate a single vector to represent b for phishing prediction.

3.1.2 The Sentence Level Layer

The sentence level layer processes the vector sequence $(\hat{u}_1, \hat{u}_2, \dots, \hat{u}_L)$ in the same way that the word level layer has employed for the vector sequence $(w_{i,1}, w_{i,2}, \ldots, w_{i,K})$ for each sentence u_i. Specifically, $(\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_L)$ is also first fed into a bidirectional LSTM module (i.e, with a forward and backward LSTM) whose results are concatenated at each position to produce the corresponding hidden vector sequence $(\hat{h}_1, \hat{h}_2, \ldots, \hat{h}_L)$. In the next step with the attention module, the vectors in $(\hat{h}_1, \hat{h}_2, \ldots, \hat{h}_L)$ are weighted and summed to finally generate the representation vector r^b for the email body b of e. Assuming the attention weights for $(\hat{h}_1, \hat{h}_2, \ldots, \hat{h}_L)$ are $(\beta_1, \beta_2, \ldots, \beta_L)$ respectively. the body vector r_b is then computed by:

$$r^b = \sum_i \beta_i \hat{h}_i \tag{4}$$

Note that the model parameters of the bidirectional LSTM modules (and the attention modules) in the word level layer and the sentence level layer are separate and they are both learnt in a single training process. Figure 1 shows the overview of the body network with hierarchical LSTMs and attention.

Once the body vector r^b has been computed, we can use it as features to estimate the phishing probability via:

$$P(y = 1|b, s) = \sigma(W_{out} r^b + b_{out}) \tag{5}$$

where W_{out} and b_{out} are the model parameters and σ is the logistic function.

3.2 Header Network

The probability estimation in Equation 5 does not consider the headers of the emails. For the email datasets with headers available, we can model the headers with a separate network and use the resulting representation as additional features to estimate the phishing probability. In this work, we consider the header s of the initial email e as a sequence of words/tokens: $(x_i, x_2, \ldots, x_H)$ where x_i is the i-th word in the header and H is the length of the header. In order to compute the representation vector r^s for s, we also employ the same network architecture as the word level layer in the body network using separate modules for embedding module, bidirectional LSTM, and attention (i.e, Section 3.1.1). An overview of this header network is presented in Figure 2.

Once the header representation vector r^s is generated, we concatenate it with the body representation vector r^b obtained from the body network, leading to the final representation vector $r = [r^b, r^s]$ to compute the probability $P(y = 1|b, s) = \sigma(W_{sub} r + b_{sub})$ (W_{sub} and b_{sub} are model parameters).

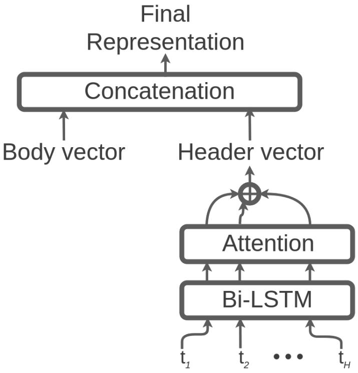


Figure 2: Hierarchical LSTMs with header network.

In order to train the models in this work, we minimize the negative log-likelihood of the models on a training dataset in which the negative log-likelihood for the email e is computed by:

$$L_c = -\log(P(y = 1|e)) \tag{6}$$

The model we have described so far is called H-LSTMs for convenience.

3.3 Supervised Attention

The attention mechanism in the body and header networks is expected to assign high weights for the informative words/sentences and downgrade the irrelevant words/sentences for phishing detection in the emails. However, this ideal operation can only be achieved when an enormous training dataset is provided to train the models. In our case of phishing email detection, the size of the training dataset is not large enough and we might not be able to exploit the full advantages of the attention. In this work, we seek for useful heuristics for the problem and inject them into the models to facilitate the operation of the attention mechanism. In particular, we would first heuristically decide a score for every word in the sentences so that the words with higher scores are considered as being more important for phishing detection than those with lower scores. Afterward, the models would be encouraged to produce attention weights for words that are close to their heuristic importance scores. The expectation is that this mechanism would help to introduce our intuition into the attention weights to compensate for the small scale of the training dataset, potentially leading to a better performance of the models. Assuming the importance scores for the words in the sentence $(v_{i,1}, v_{i,2}, \ldots, v_{i,K})$ be $(g_{i,1}, g_{i,2}, \ldots, g_{i,K})$ respectively, we force the attention weights $(\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,K})$ (Equation 1) to be close to the importance scores by penalizing the models that render large square difference between the attention weights and the importance scores. This amounts to adding the square difference into the objective function in Equation 6:

$$L_e = L_c + \lambda \sum_{i,j} (g_{i,j} - \alpha_{i,j})^2 \tag{7}$$

where λ is a trade-off constant.

Importance Score Computation

In order to compute the importance scores, our intuition is that a word is important for phishing detection if it appears frequently in phishing emails and less frequently in legitimate emails. The fact that an important word does not appear in many legitimate emails helps to eliminate the common words that are used in most documents. Consequently, the frequent words that are specific to the phishing emails would receive higher importance scores in our method. Note that our method to find the important words for phishing emails is different from the prior work that has only considered the most frequent words in the phishing emails and ignored their appearance in the legitimate emails.

We compute the importance scores as follow. For every word v in the vocabulary, we count the number of the phishing and legitimate emails in a training dataset that contain the word. We call the results as the phishing email frequency and the legitimate email frequency respectively for v. In the next step, we sort the words in the vocabulary based on its phishing and legitimate email frequencies in the descending order. After that, a word v would have a phishing rank ($phishingRank(v)$) and a legitimate rank ($legitimateRank(v)$) in the sorted word sequences based on the phishing and legitimate frequencies (the higher the rank is, the less the frequency is). Given these ranks, the unnormalized importance score for v is computed by:[3]

$$score[v] = \frac{legitimateRank[v]}{phishingRank[v]} \tag{8}$$

The rationale for this formula is that a word would have a high importance score for phishing prediction if its legitimate rank is high and its phishing rank is low. Note that we use the ranks of the words instead of the frequencies because the frequencies are affected by the size of the training dataset, potentially making the scores unstable. The ranks are less affected by the dataset size and provide a more stable measure. Table 1 demonstrates the top 20 words with the highest unnormalized importance scores in our vocabulary.

The H-LSTMs model augmented with the supervised attention mechanism above is called H-LSTM+supervised in the experiments.

account	21.45
your	15.00
click	14.11
mailbox	9.59
cornell	9.58
link	9.37
verify	8.83
customer	8.63
access	8.50
reserved	8.03
dear	7.85
log	7.70
accounts	7.61
paypal	7.52
complete	7.37
service	7.15
protect	6.95
secure	6.94
mail	6.70
clicking	6.63

Table 1: Top 20 words with the highest scores.

3.3.1 Training

We train the models in this work with stochastic gradient descent, shuffled mini-batches, Adam update rules [KB14]. The gradients are computed via back-propagation while dropout is used for regularization [SHK+14]. We also implement gradient clipping to rescale the Frobenius norms of the non-embedding weights if they exceed a predefined threshold.

4 Evaluation

4.1 Datasets and Preprocessing

The models in this work are developed to participate in the First Security and Privacy Analytics Anti-Phishing Shared Task (IWSPA-AP 2018) [EDB+18]. The organizers provide two datasets to train the models for email phishing recognition. The first dataset involves emails that only have the body part (called *data-no-header*) while the second dataset contains emails with both bodies and headers (called *data-full-header*. These two datasets translate into two shared tasks to be solved by the participants. The statistics of the training data for these two datasets are shown in Table 2.

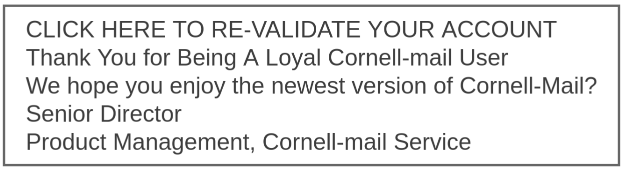
Datasets	**#legit**	**#phish**
data-no-header	5092	629
data-full-header	4082	503

Table 2: Statistics of the *data-no-header* and *data-full-header* datasets. **#legit** and **#phish** are the numbers of legitimate and phishing emails respectively.

[3] The actual important scores of the words we use in Equation 7 are normalized for each sentence.

The raw test data (i.e, without labels) for these datasets are released to the participants at a specified time. The participants would have one week to run their systems on such raw test data and submit the results to the organizers for evaluation.

Regarding the preprocessing procedure for the datasets, we notice that a large part of the text in the email bodies is quite unstructured. The sentences are often short and/or not clearly separated by the ending-sentence symbols (i.e, {. ! ?}). In order to split the bodies of the emails into sentences for our models, we develope an in-house sentence splitter specially designed for the datasets. In particular, we determine the beginning of a sentence by considering if the first word of a new line is capitalized or not, or if a capitalized word is immediately followed by an ending-sentence symbol. The sentences whose lengths (numbers of words) are less than 3 are combined to create a longer sentence. This reduces the number of sentences significantly and expands the context for the words in the sentences as they are processed by the models. Figure 3 shows a phishing email from the datasets.



CLICK HERE TO RE-VALIDATE YOUR ACCOUNT
Thank You for Being A Loyal Cornell-mail User
We hope you enjoy the newest version of Cornell-Mail?
Senior Director
Product Management, Cornell-mail Service

Figure 3: A case in which splitting body into sentences cannot be done as usual. (Phishing email: 28.txt in data-no-header).

4.2 Baselines

In order to see how well the proposed deep learning models (i.e, H-LSTMs and H-LSTMs+supervised) perform with respect to the traditional methods for email phishing detection, we compare the proposed models with a baseline model based on Support Vector Machines (SVM) [CNU06]. We use the `tf-idf` scores of the words in the vocabulary as the features for this baseline [CNU06]. Note that since the email addresses and urls in the provided datasets have been mostly hidden to protect personal information, we cannot use them as features in our SVM baselines as do the previous systems. In addition, we examine the performance of this baseline when the pre-trained word embeddings are included in its feature set. This allows a fairer comparison of SVM with the deep learning models in this work that take pre-trained word embeddings as the input.

We employ the implementation of linear and nonlinear (kernel) SVM from the `sklearn` library [PVG+11] for which the `tf-idf` representations of the emails are obtained via the `gensim` toolkit [ŘS10]. The word embedding features are computed by taking the mean vector of the pre-trained embeddings of the words in the emails [NPG15].

4.3 Hyper-parameter Selection

As the size of the provided datasets is small and no development data is included, we use a 5-fold stratified cross validation on the training data of the provided datasets to search for the best hyper-parameters for the models. The hyper-parameters we found are as follows.

The size of word embedding vector is 300 while the cell sizes are set to 60 for all the LSTMs in the body and header networks. The size of attention vectors at the attention modules for the body and header networks are also set to 60. The λ coefficient for supervised attention is set to 0.1, the threshold for gradient clipping is 0.3 and the drop rate for drop-out is 0.5. For the Adam update rule, we use the learning rate of 0.0025. Finally, we set $C = 10.0$ for the linear SVM baseline. The nonlinear version of SVM we use is C-SVC with radial basis function kernel and $(C, \gamma) = (50.0, 0.1)$.

4.4 Results

In the experiments below, we employ the precision, recall and F1-score to evaluate the performance of the models for detecting phishing emails. In addition, the proposed models H-LSTMs and H-LSTMs+supervised only utilize the header network in the evaluation on *data-full-header*.

Data without header

In the first experiment, we focus on the first shared task where email headers are not considered. We compare the proposed deep learning models with the SVM baselines. In particular, in the first setting, we use *data-no-header* as the training data and perform a 5-fold stratified cross-validation to evaluate the models. In the second setting, *data-no-header* is also utilized as the training data, but the bodies extracted from *data-full-header* (along with the corresponding labels) are employed as the test data. The results of the first setting are shown in Table 3 while the results of the second setting are presented in Table 4. Note that we report the performance of the SVM baselines when different combinations of the two types of features (i.e, `tf-idf` and word embeddings) are employed in these tables.

The first observation from the tables is that the effect of the word embedding features for the SVM

Models	Precision	Recall	F1
H-LSTMs+supervised	0.9784	0.9466	**0.9621**
H-LSTMs	0.9638	0.9448	0.9542
Linear SVM			
+tfidf	0.9824	0.8856	0.9313
+emb	0.9529	0.9206	0.9364
+tfidf+emb	0.9837	0.9253	0.9536
Kernel SVM			
+tfidf	0.9684	0.8730	0.9180
+emb	0.9408	0.9141	0.9273
+tfidf+emb	0.9714	0.9174	0.9436

Table 3: Performance comparison between the proposed models H-LSTMs and H-LSTMs+supervised with the baseline models Linear and Kernel SVM. *tfidf* indicates `tf-idf` features while *emb* denotes features obtained from the pre-trained word embeddings.

Models	Precision	Recall	F1
H-LSTMs+supervised	0.8892	0.7395	**0.8075**
H-LSTMs	0.8934	0.7054	0.7883
Linear SVM			
+tfidf	0.8864	0.6978	0.7809
+emb	0.8112	0.6918	0.7468
+tfidf+emb	0.8695	0.7018	0.7767
Kernel SVM			
+tfidf	0.8698	0.7038	0.7780
+emb	0.8216	0.6501	0.7259
+tfidf+emb	0.8564	0.6937	0.7665

Table 4: Performance of all models on the test data (data-full-headers).

models are quite mixed. It improves the SVM models with just `tf-idf` features significantly in the first experiment setting while the effectiveness is somewhat negative in the second experiment setting. Second, we see that the two versions of hierarchical LSTMs (i.e, H-LSTMs and H-LSTMs+supervised) outperform the baseline SVM models in both experiment settings. The performance improvement is significant with large margins (up to 2.7% improvement on the absolute F1 score) in the second experiment setting (i.e, Table 4). The main gain is due to the recall, demonstrating the generalization advantages of the proposed deep learning models over the traditional methods for phishing detection with SVM. Comparing H-LSTMs+supervised and H-LSTMs, we see that H-LSTMs+supervised is consistently better than H-LSTMs with significant improvement in the second setting. This shows the benefits of supervised attention for hierarchical LSTM models for email phishing detection. Finally, we see that the performance in the first setting is in general much better than those in the second setting. We attribute this to the fact that text data in *data-no-header* and *data-full-header* is quite different, leading to the mismatch between data distributions of the training data and test data in the second experiment setting.

In the final submission for the first shared task (i.e, without email headers), we combine the training data from *data-no-header* with the extracted bodies (along with the corresponding labels) from the training data of *data-full-header* to generate a new training set. As H-LSTM+supervised is the best model in this development experiment, we train it on the new training set and use the trained model to make predictions for the actual test set of the first shared task.

Data with full header

In this experiment, we aim to evaluate if the header network can help to improve the performance of H-LSTMs. We take the training dataset from *data-full-header* to perform a 5-fold cross-validation evaluation. The performance of H-LSTMs when the header network is included or excluded is shown in Table 5.

Models	Precision	Recall	F1
H-LSTMs (only body)	0.9732	0.9534	0.9631
H-LSTMs + headers	0.9816	0.9596	0.9705

Table 5: Cross-validation performance of H-LSTMs with using headers compared to the original version.

From the table, we see that the header network is also helpful for H-LSTMs as it helps to improve the performance of H-LSTMs for the dataset with email headers (an 0.7% improvement on the F1 score).

In the final submission for the second shared task (i.e, with email headers), we simply train our best model in this setting (i.e, H-LSTM+supervised) on the training dataset of *data-full-header*.

The time for the training and test process of the proposed (and submitted) models is shown in Table 6. Note that the training time of *H-LSTMs+supervised* (for the first shared task) is longer than that of *H-LSTMs+headers+supervised* (for the second shared task) since the training data of the former model includes both the original training data of the first task and the extracted bodies from the training data of the second task. The test data of the first shared task with *H-LSTMs+supervised* is also larger than that of the second shared task with *H-LSTMs+headers+supervised*.

Models	Training	Test
	Time	Time
H-LSTMs+supervised	3.7 hours	4 minutes
H-LSTMs+headers+supervised	1.5 hours	1 minute

Table 6: Training and test times of the submitted models. The experiments are run on a single NVIDIA Tesla K80 GPU.

Comparision with the participating systems on the actual test sets

Tables 7 and 8 show the best performance on the actual test data of all the teams that participate in the shared tasks. Table 7 reports the performance for the first shared task (i.e, without email headers) while Table 8 presents the performance for the second shared task (i.e, with email headers). These performance is measured and released by the organizers. The performance of the systems we submitted is shown in the rows with our team name (i.e, *TripleN*).

Teams	Precision	Recall	F1
TripleN (our team)	**0.981**	**0.978**	**0.979**
Security-CEN@Amrita	0.962	0.989	0.975
Amrita-NLP	0.972	0.974	0.973
CEN-DeepSpam	0.951	0.964	0.958
CENSec@Amrita	0.914	0.998	0.954
CEN-SecureNLP	0.890	1.000	0.942
CEN-AISecurity	0.936	0.910	0.923
Crypt Coyotes	0.936	0.910	0.923

Table 7: The best performance of all the participating teams in the first shared task with no email headers.

Teams	Precision	Recall	F1
Amrita-NLP	0.998	0.994	0.996
TripleN (our team)	**0.990**	**0.992**	**0.991**
CEN-DeepSpam	1.000	0.978	0.989
Security-CEN@Amrita	0.998	0.976	0.987
CENSec@Amrita	0.882	1.000	0.937
CEN-AISecurity	0.957	0.900	0.928
CEN-SecureNLP	0.880	0.971	0.924
Crypt Coyotes	0.960	0.863	0.909

Table 8: The best performance of all the participating teams in the second shared task with email headers.

As we can see from the tables, our systems achieve the best performance for the first shared task and the second best performance for the second shared task. These results are very promising and demonstrate the advantages of the proposed methods in particular and deep learning in general for the problem of email phishing recognition.

5 Conclusions

We present a deep learning model to detect phishing emails. Our model employs hierarchical attentive LSTMs to model the email bodies at both the word level and the sentence level. A header network with attentive LSTMs is also incorporated to model the headers of the emails. In the models, we propose a novel supervised attention technique to improve the performance using the email frequency ranking of the words in the vocabulary. Several experiments are conducted to demonstrate the benefits of the proposed deep learning models.

References

[ANNWN07] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. A comparison of machine learning techniques for phishing detection. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, pages 60–69. ACM, 2007.

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *arXiv preprint arXiv:1409.0473*, 2014.

[BCP+08] Andre Bergholz, Jeong Ho Chang, Gerhard Paass, Frank Reichartz, and Siehyun Strobel. Improved phishing detection using model-based features. In *CEAS*, 2008.

[BMS08] Ram Basnet, Srinivas Mukkamala, and Andrew H Sung. Detection of phishing attacks: A machine learning approach. In *Soft Computing Applications in Industry*, pages 373–383. Springer, 2008.

[CNU06] Madhusudhanan Chandrasekaran, Krishnan Narayanan, and Shambhu Upadhyaya. Phishing email detection based on structural properties. In *NYS Cyber Security Conference*, volume 3, 2006.

[DTH06] Rachna Dhamija, J Doug Tygar, and Marti Hearst. Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 581–590. ACM, 2006.

[EDB+18] Ayman Elaassal, Avisha Das, Shahryar Baki, Luis De Moraes, and Rakesh Verma. Iwspa-ap: Anti-phising shared task at acm international workshop on security and privacy analytics. In *Proceedings of the 1st IWSPA Anti-Phishing Shared Task*. CEUR, 2018.

[FST07] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to detect phishing emails. pages 649–656. ACM, 2007.

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520, 2011.

[GS05] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[GTJA17] BB Gupta, Aakanksha Tewari, Ankit Kumar Jain, and Dharma P Agrawal. Fighting against phishing attacks: state of the art and future challenges. *Neural Computing and Applications*, 28(12):3629–3654, 2017.

[HA11] Isredza Rahmi A Hamid and Jemal Abawajy. Hybrid feature selection for phishing email detection. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 266–275. Springer, 2011.

[HS97] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. In *Neural Computation*, 1997.

[JJJM07] Tom N Jagatic, Nathaniel A Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, 2007.

[KB14] Diederik P. Kingma and Jimmy. Ba. Adam: A method for stochastic optimization. In *arXiv: 1412.6980*, 2014.

[LCLZ17] Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1789–1798, 2017.

[LUF$^+$16] Lemao Liu, LemLiu, Masao Utiyama, Andrew Finch, ao Sumita, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. Neural machine translation with supervised attention. *arXiv preprint arXiv:1609.04186*, 2016.

[LV12] V Santhana Lakshmi and MS Vijaya. Efficient prediction of phishing websites using supervised learning algorithms. *Procedia Engineering*, 30:798–805, 2012.

[LXLZ15] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273, 2015.

[MAS$^+$11] Anutthamaa Martin, Na Anutthamaa, M Sathyavathy, Marie Manjari Saint Francois, Dr V Prasanna Venkatesan, et al. A framework for predicting phishing websites using neural networks. *arXiv preprint arXiv:1109.1074*, 2011.

[MSC$^+$13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[MTM14] Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications*, 25(2):443–458, 2014.

[MWI16] Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. Supervised attentions for neural machine translation. *arXiv preprint arXiv:1608.00112*, 2016.

[NG15a] Thien Huu Nguyen and Ralph Grishman. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 365–371, 2015.

[NG15b] Thien Huu Nguyen and Ralph Grishman. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48, 2015.

[NG16] Thien Huu Nguyen and Ralph Grishman. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[NPG15] Thien Huu Nguyen, Barbara Plank, and Ralph Grishman. Semantic representations for domain adaptation: A case study on the tree kernel-based method for relation extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2015.

[phi17a] 2017 data breach report finds phishing, email attacks still potent. In *https://digitalguardian.com/blog/2017-data-breach-report-finds-phishing-email-attacks-still-potent*, 2017.

[phi17b] Phishing scams cost american businesses half a billion dollars a year. In *Forbes: Phishing Scams Cost American Businesses Half a Billion Dollars a Year*, 2017.

[pod16] How john podesta's emails were hacked. In *Forbes: How John Podestas Emails Were Hacked and How to Prevent it from Happening to You*, 2016.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[ŘS10] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, and Yoshua Bengio. Dropout: A simple way to prevent neural networks from overfitting. In *The Journal of Machine Learning Research*, 2014.

[Sin05] David Singer. Identification of spoofed email. Google Patents, August 25 2005. US Patent App. 10/754,220.

[SNG+17] Hossein Siadati, Toan Nguyen, Payas Gupta, Markus Jakobsson, and Nasir Memon. Mind your smses: Mitigating social engineering in second factor authentication. *Computers & Security*, 65:14–28, 2017.

[SNM15] Hossein Siadati, Toan Nguyen, and Nasir Memon. Verification code forwarding attack (short paper). In *International Conference on Passwords*, pages 65–71. Springer, 2015.

[SNM17] Hossein Siadati, Toan Nguyen, and Nasir Memon. X-platform phishing: Abusing trust for targeted attacks short paper. In *International Conference on Financial Cryptography and Data Security*, pages 587–596. Springer, 2017.

[YYD+16] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

[ZY12] Ningxia Zhang and Yongqing Yuan. Phishing detection using neural network. *CS229 lecture notes*, 2012.