2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT) | 978-1-6654-9402-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/WI-IAT55865.2022.00028

LMs go Phishing: Adapting Pre-trained Language Models to Detect Phishing Emails

Kanishka Misra
Department of CIT
Purdue University
West Lafayette, USA
kmisra@purdue.edu

Julia Taylor Rayz
Department of CIT
Purdue University
West Lafayette, USA
jtaylor1@purdue.edu

***Abstract*—Despite decades of research, the problem of Phishing in everyday email communication is ever so prevalent. Traditionally viewed as a text-classification task, the task of phishing detection is an active defense against phishing attempts. Meanwhile, progress in natural language processing has established the universal usefulness of adapting pre-trained language models to perform downstream tasks, in a paradigm known as pre-train-then-fine-tune. In this work, we build on this paradigm, and propose two language models that are adapted on 725k emails containing phishing and legitimate messages. We use these two models in two ways: 1) by performing classification-based fine-tuning, and 2) by developing a simple priming-based approach. Our approaches achieve empirical gains over a good deal of prior work, achieving near perfect performance on in-domain data, and relative improvements on out-of-domain emails.**

***Index Terms*—phishing detection, language models, deep learning, BERT, GPT2**

I. Introduction

Advances in AI have introduced new opportunities and challenges in cybersecurity in general. These advances provide the potential to tackle one of the most difficult problems in cybersecurity: social engineering. Social engineering poses a uniquely difficult problem in cybersecurity because of a combination of factors. First, social engineering is low cost and presents multiple increasingly complex and subtle attack vectors. This combined with the rapid increase in computer users presents a large and growing attack surface. Second, the majority of computer users are not cybersecurity literate, with under 30% judged competent on basic knowledge.[1] Third, it takes advantage of humans being the weakest link in cybersecurity by taking advantage of human vulnerabilities like habit formation and susceptibility to persuasive techniques. These conditions have resulted in increased use of social engineering techniques. It is estimated that 70-90% of cyber-attacks utilize some element of social engineering. Phishing is the most common type of social engineering, and has been increasing every year. Phishing attacks are commonly performed though emails, and are, thus, of immediate concern to Web Intelligence.

This research is partially supported by NSF Award #2039605

[1] https://www.pewresearch.org/internet/2017/03/22/what-the-public-knows-about-cybersecurity/

Phishing detection has been traditionally viewed as a text-classification task. Recent years have established that best models to do text classification is by performing transfer learning over representations of pre-trained language models [1], [2]. This paradigm has also been known as *pre-train and fine-tune*, as language models are first pre-trained on large-scale corpora by learning to predict words in context, and are then fine-tuned to perform specific tasks. While previous work have proposed ML and DL based methods, they have mostly used smaller datasets (ranging from 2,000 emails to 20,000 emails), which are not conducive to the pre-train then fine-tune paradigm.

In this work, we explore the application of this now-ubiquitous paradigm in the task of detecting phishing emails. Our contributions is as follows: First, we collect a dataset of $\approx$ 830,000 emails, spanning both phishing and legitimate messages, and covering multiple types of styles and conversational topics, far exceeding the amount covered in a majority of previous work. Next, we fine-tune two large language models on a subset of these emails and test the extent to which they capture the statistics of email communication. Using our fine-tuned LMs, we propose two approaches to performing phishing detection: (1) a standard fine-tuning approach, and (2) a novel 'priming' approach, which circumvents the need for further training, and allows classification of an email in a manner similar to zero-shot classification—by predicting special label tokens in context. We then perform extensive analyses of our methods, ranging from the effect of pre-training, to the ability of our LMs to sufficiently handle out-of-domain emails, and finally, a comparison of our methods to a subset of prior work. Our preliminary experiments establish the importance of considering pre-trained features from general language corpora in improving models' ability to extrapolate outside of their training experience. For phishing detection, we find that both our approaches outperform baselines that involve statistical learning as well as contemporary deep-learning method, especially on emails that are from the same domain as the training set. On emails that contain topics that the models have never encountered before, the performance of most models substantially declines, suggesting that they have overfit the training domain. However, our priming approach shows most resistance to this change in domain, substantially

978-1-6654-9402-1/22/$31.00 ©2022 IEEE
DOI 10.1109/WI-IAT55865.2022.00028

out performing all baselines as well as the fine-tuned classifiers. Our findings highlight: (1) the importance of using pre-trained features; (2) the benefits of remaining faithful to the LM's original task in detecting phishing attempts for emails outside of training domain; and (3) the tendency of fine-tuned models to become domain-specific, which leads to near-perfect performance on in-domain phishing detection datasets, but relatively weakened performance on emails that are out-of-domain.

II. Background

A. NLP approaches for studying Phishing attempts

NLP for phishing detection is now a well-established field, with at least three surveys in the past three years alone [3]–[5]. A majority of NLP-based phishing detection works in the past have largely used extensive feature engineering, as was the dominant paradigm in much of machine learning based NLP research prior to the popularity and feasibility of deep learning. Popularly used features ranged from information retrieval measures such as word and n-gram tf-idf scores, to meta-features such as number of unique words, presence of an attachment, etc., to topic model distributions, which were primarily extracted semi-automatically using a number of preprocessing techniques [6]–[9]. The recently proposed Phishbench framework [4] provides a total of 63 diverse types of features, along with numerous tools to perform cross validation, up/down-sampling, sophisticated feature selection, and even a wide array of classifiers, to perform phishing detection. The framework almost entirely encompasses phishing detection research prior to pure deep learning approaches, and we use it to compute baseline performance on our datasets. For a more comprehensive list of feature-engineering approaches, see [4], [5]. A slightly different trend from the purely-statistical learning paradigm is the use of knowledge-based features extracted from sentences using an ontology, combined with a semantic parser [10]. Their approach captures rich ontological hierarchies and relationships between concepts evoked by emails, as opposed to shallow surface level features as used in prior approaches. The authors of [10] show nearly equivalent performance on phishing detection against traditional surface level features discussed above.

With the unprecedented progress in Deep learning, more recent works have proposed several deep learning-based systems that automatically detect and optimize dense features to perform phishing detection [11], instead of relying on semi-automatically and manually specified features. Notable works include thr THEMIS model [12], which uses an RCNN architecture and is trained on 6.1k emails and the CatBERT model [13], which fine-tunes the BERT model, and is trained on 4.2m privately stored emails, we use the same BERT model as a baseline in our experiments. Finally, some works have also combined features extracted from models like BERT with tf-idf features and applied statistical learning classifiers [14].

A fundamental drawback of these prior works is that they either experiment with relatively smaller datasets (ranging from around 1,000 to 20,000 emails), or use private internal emails. As a result their training and testing sets are usually from the same fixed domain—i.e., if for instance, a classifier is trained to distinguish between real corporate emails and phishing emails from a particular dataset, with sufficient data/informative features, it is highly expected for it to distinguish between emails that are from the same distribution as its training set. What is left unknown is the performance of the classifier on emails that are phishing but follow completely different styles and use radically different messages to perform social engineering. One exception to this prevalent limitation is the work by [15], who propose an online learning based phishing detection system, that trains on emails from a fixed set of years and is evaluated on emails from a disjoint set of years (usually in the future), to capture its performance on emails that the classifier is yet to see. However, a majority of the emails they use are private, making it difficult to perform level-ground comparisons.

B. Language Modeling

Our methods primarily rely and build on the concept of language modeling, the task of assigning probabilities to word sequences in a given language. More formally, given a vocabulary set $\mathcal{V}$, the goal of a language model (LM) is to map sequences drawn from its Kleene closure $\mathcal{V}^*$—which is the set of finite length sequences formed by elements of $\mathcal{V}$—to $\mathbb{R}$. That is, an LM defines a probability distribution over natural language sequences. Originally constructed using n-gram statistics [16], which were limited in context as well as computational power, contemporary LMs often make use of powerful neural network architectures such as recurrent neural networks [17] and transformers [18] to estimate the mapping $\mathcal{V}^* \rightarrow \mathbb{R}$. These neural network-based LMs are often trained by minimizing the negative log-probability of sequences y) in a large training corpus $\mathcal{Y}$ with respect to their parameters θ:

$$\mathcal{L}(\theta) = -\sum_{y \in \mathcal{Y}} \log p(y) \tag{1}$$

Equation (1) is also known as the negative log-likelihood loss, minimizing which is equivalent to minimizing the cross-entropy of the entire corpus. Contemporary neural network language models are trained using the transformer architecture [18], which utilizes the concept of self-attention, where the vector representation of every word is composed of a weighted representation of the words that occur in its context. Examples of contemporary LMs include the GPT family [19]–[21], which are trained auto-regressively—i.e., left-to-right, as well as BERT [2], RoBERTa [22], etc., which are trained bidirectionally to fill in missing words in context.

LMs such as the ones described above have fundamentally revolutionized the field of NLP, as by training to minimize the language modeling objective—i.e., (1)—on large corpora, they capture general distributional properties of words and phrases in context. These distributional properties are stored in their dense parameters, which can then be fine-tuned on specific tasks, using standard transfer learning practices. This general paradigm is also known as *pre-train-then-fine-tune*,

which has lead to state of the art performance on a number of natural language tasks [23]. While pre-trained LMs such as GPT2 [20] and BERT [2] capture quite a bit about general English, they sometimes lead to modest performance on tasks that deal domain-specific language—such as biomedical data [24]. To improve LMs' performance on domain-specific tasks, [25] recommend an additional round of LM-based training on language data from either the domain of the task, in a process known as Domain Adaptive Pre-training (DAPT), which we build on in our work. We hypothesize that models that have access to the format of an email, the style of language, etc., would perform better than models that do not, and therefore experiment with DAPT-like training on a large collection of emails in our experiments.

III. Methodology

A. Data

1) Collection: We consider a variety of different sources to construct a corpus of phishing and legitimate emails. We collect legitimate emails from three primary sources: (1) the **Enron corpus**, consisting of a total of 517,401 emails sent and received by the employees of Enron corporation; (2) the **Avocado corpus**[2], which consists of around 1.2 million emails exchanged between 279 accounts of employees working at a now-defunct company referred to as "Avocado" for the purposes of anonymity; and (3) the legitimate-email portion of the **IWSPA-AP training data** [26], which contains 4,082 emails. This amounts to a total of around 1.7 million raw legitimate emails.

We consider four different sources for collecting phishing emails: (1) the **Nazario** phishing corpus,[3] from which we collect 5014 emails; (2) the phishing email portion of the **IWSPA-AP training data**, from which we collect 525 emails; (3) the **Fraud corpus**,[4] a corpus of 4,038 emails sent by malicious actors posing to be a "Nigerian prince"; and (4) the **Untroubled corpus**,[5] a collection of emails collected by an individual since 1997 – we use emails from 2011 to 2016 from this collection, averaging at around 500k emails each. Combining these sources, we have 3 million raw phishing emails, prior to any data pre-processing steps.

2) Pre-processing: We apply a number of pre-processing steps to ensure the data we on which we train and evaluate our models is as clean as possible. To this end, we discard all emails without any subject, as well as those that were in non-standard encoding (e.g., base64). When emails are bundled in a multipart format, we treat them as separate emails. When applicable, we parse out all html tags using the python library `BeautifulSoup`, leaving only the textual content of the email. The emails were further cleaned by using standard regex operations—examples include removing noisy characters from non-standard encoding schemes, removing html-tags that our previous step had failed to remove, or removing "bug reports" that contained programming language syntax and error messages. We also replace phone numbers, URLs, and emails with special tags (`<PHONE>`, `<URL>`, and `<EMAIL>`, respectively). We do this not only to significantly reduce the length of the email text, but also to endow models general—as opposed to specific—statistical information about such metadata. To add additional general information to the model, we also replace named entities such as a person's name (`<PERSON>`), date (`<DATE>`), and time (`<TIME>`) with special tokens using an off-the-shelf named entity recognizer model from spacy.[6] Apart from the aforementioned preprocessing steps, we also note that all email sources—especially the Untroubled corpus—are plagued with the issue of duplicate emails. We deduplicate our corpora and include each email only once in our experiments. Finally, we only consider emails that are less than 1024 tokens long, as measured by GPT2's tokenizer. We format our emails by using special tokens to demarcate the subject, separating it from the body of the email, and end each email with an end-of-sequence token (`<eos>`):[7]

`<subject>` Subject `</subject>` Body `<eos>`

Type	Source	Use	Emails
Legitimate	Enron	FT, PD	336,247
	Avocado	FT, PD	265,027
	IWSPA-AP-Legit	PD	2,634
Phishing	Nazario	FT, PD	2,664
	Untroubled	FT, PD	220,343
	IWSPA-AP-Phish	PD	288
	Fraud	PD	1795
Total			828,998

TABLE I: Statistics of the Email sources we consider. The 'Use' column indicates how a given source is used in our experiments. **FT**: Fine-tuning; **PD**: Phishing Detection.

Table I shows the number of clean emails we use in our experiments from each source, after our pre-processing step. It is worth noting that our final dataset has considerably fewer emails than the raw numbers—many emails were discarded due to a combination of deduplication and rigorous text-based pre-processing steps. Nevertheless, we end up with close to 830k emails in our experiments, making this—to our knowledge—one of the largest collection of emails used in academic phishing research.

B. Language Modeling

We draw on recent progress in NLP, driven by the use of language models to perform domain-adaptive pre-training [25] on our email corpora. Specifically, we use the GPT2 architecture [20], which uses a decoder-only transformer and is trained to perform language modeling in an auto-regressive manner. We consider two sizes of the GPT2 model – *small* and *medium*, which comprise of 12 and 24 transformer layers,

[2]https://catalog.ldc.upenn.edu/LDC2015T03
[3]https://monkey.org/~jose/phishing/
[4]https://www.kaggle.com/datasets/rtatman/fraudulent-email-corpus.
[5]http://untroubled.org/spam/

[6]https://spacy.io/; we use the `en_web_trf` model which uses a RoBERTa architecture [22].
[7]in practice, we use the same token as the one used by GPT2 (`<|endoftext|>`), but represent it in shorthand for brevity.

with 12 and 16 self-attention heads, respectively. These models were pre-trained using the language modeling objective on 40GB of text obtained from the web. For our experiments, we add 8 special tokens—as described in our pre-processing step—to the GPT2 vocabulary (original size 50,257 tokens), and initialize them with random values to ensure that the models' tokenizer *always* tokenizes them into distinct tokens.

C. Experimental setup

The goal of this is work is to shed light on the extent to which LMs can be applied in phishing research. To this end, our methodological approach consists of two main stages. First, we perform experiments to train and fine-tune LMs on corpora that reflect the domain of email communication. In the second stage of our approach, we utilize our domain-adapted LMs to perform classification of emails as "phishing" or "legitimate." Our two-stage approach is primarily motivated by recent results in task and domain adaptation using pre-trained LMs [25], where one can demonstrate substantial performance gains on downstream domain-specific tasks (e.g., biomedical text classification [24]) by adapting pre-trained LMs on specialized corpora that approximate the desired domain. We now discuss our two stages in greater detail:

1) Domain adaptation of GPT2 on Emails: In this stage, we adapt GPT2-small and GPT2-medium to the domain of email communication. We use as our training set a mix of legitimate and phishing emails selected from our collection of emails (described in §III-A). Specifically, we combine the Enron and Avocado corpora and leave from this set 10,000 emails for classification experiments. Similarly, we combine the UNTROUBLED and NAZARIO corpora, and leave out 5,000 phishing emails for classification. This results in a total of 805,934 emails, which we split into 90% training (725,340 emails) and 10% validation sets (80,594 emails). We hold out the IWSPA and the Fraud emails in order to use them as evaluation sets that focus on different domains than the ones reflected in the LM-training experiments. Testing our LMs on them will let us quantify their out-of-domain performance—we delve into these analyses in the second stage. To each email in our training and validation sets, we append special tokens that indicate whether the email was phishing (<phish>) or legitimate (<legit>), as these will be used for our priming-based classification method, explained in the next subsection. Note that these special tokens are absent in emails reserved for classification and out-of-domain testing. We then fine-tune the two pre-trained LMs on our training set using the language modeling objective. We use the AdamW optimization algorithm [27] with a learning rate of 5e-05, and a batch size of 16 for the small model and 6 for the large. We train our models for a total of 5 epochs. Hereafter, we name our fine-tuned models PHISHLM$_{small}$ and PHISHLM$_{medium}$, depending on the variant of GPT2 used. This training scheme is also depicted in Figure 1a.

2) Phishing Detection: The goal of the second stage is to use the PHISHLM models from the previous stage to perform phishing detection. To this end, we first split the 15,000 emails left out from our fine-tuning experiments into training, validation, and testing data (using a 80/10/10 split). We then consider two different approaches of performing phishing detection using our PHISHLMs:

a) Fine-tuning based classifier: In our first approach, we fine-tune the entire model, by passing the embedding of the final token (<eos>) to a linear layer, and training the model end-to-end on our training set. Mathematically:

$$email = w_{\texttt{<subject>}}, w_1, \ldots, w_n, w_{\texttt{<eos>}}, \quad (2)$$

$$h_{\texttt{<eos>}} = \text{PHISHLM}(email)[-1], \quad (3)$$

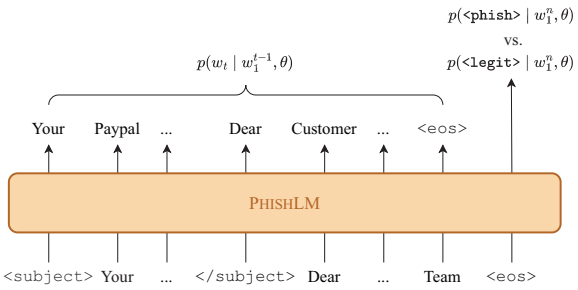
$$\hat{y} = h_{\texttt{<eos>}}\mathbf{W} + \mathbf{b}, \quad (4)$$

where *email* is the sequence of input email tokens, PHISHLM$(email)[-1]$ picks out the final-layer representation of the <eos> token, and $\mathbf{W}$ and $\mathbf{b}$ are the parameters of the linear layer. In our experiments, we fine-tune both variants of our trained PHISHLM using the AdamW optimizer [27]. We use a batch size of 16 for PHISHLM$_{small}$ and 6 for PHISHLM$_{medium}$, and search for learning rates in the set {1e-06, 5e-06, 1e-05, 5e-05}. We train the models for a maximum of 10 epochs and stop training if the loss on the validation set does not drop for 3 consecutive steps. We use the model state with the best performance on the validation set as our final model. Our fine-tuning scheme is shown in Figure 1b.

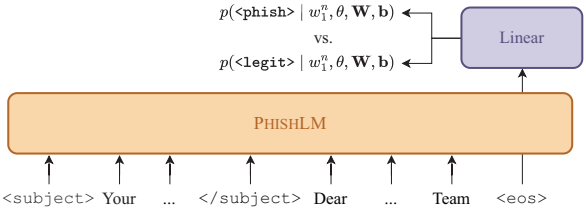
b) Classifier based on In-context Priming: Our second approach is inspired from recent works in zero and few-shot learning with LMs that make use of in-context prompts [21], [28] which prime an LM [29] with a prompt to predict label-specific tokens. For example, one could use a movie review, followed by a prompt such as "This review is" as input to an LM and then compare the probabilities of "good" vs "bad" as the label probabilities for "positive" and "negative." Taking inspiration from this literature, we propose a priming/prompting based approach to phishing detection. Recall that during training of our LMs, we append an input email with special tokens that indicate whether the email was phishing (<phish>) or legitimate (<legit>). We use this feature of our training scheme and devise a classification mechanism where we condition PHISHLM with an email, and then compare the probabilities of <phish> vs <legit>, given that email. That is, we compare: $p($<phish> $\mid$ *email*$)$ vs $p($<phish> $\mid$ *email*$)$. This approach is desirable as it circumvents any need for specialized fine-tuning of the model, which—albeit ubiquitous in NLP research—is prone to overfitting to the task, and adds substantial computational overhead.

Corpus	Phishing	Legitimate	Total
Fine-tuning/Training from scratch			
Train	193,104	532,236	725,340
Test	21,456	59,138	80,594
Phishing Detection			
Training (ID)	4,000	8,000	12,000
Validation (ID)	500	1,000	1,500
Test (ID)	500	1,000	1,500
OOD	2,083	2,634	4,717

TABLE II: Composition of our experimental datasets.



(a) PHISHLM training scheme, along with a mechanism to directly predict if an input email is phishing (<phish>) or legitimate (<legit>) as part of the LM-training objective.



(b) Full model fine-tuning scheme, where PHISHLM is fine-tuned to classify if an input email is phishing (<phish>) or legitimate (<legit>) using a linear layer.

Fig. 1: Modeling Setup for Phishing Detection. w_1^m is shorthand for the sequence $w_1, \ldots, w_m$. The symbols θ and $\mathbf{W}, \mathbf{b}$ represent the parameters of the language model and the linear classification layer, respectively.

We evaluate our models on their capacity to successfully differentiate between phishing and legitimate emails by testing them in two different settings. First, we test models on the 1,500 emails that comprise of the test set from our original classification split. This data is roughly from the same distribution as the model's training data (both, during fine-tuning with the LM objective, and for the classification setting). We therefore refer to this as **in-domain** (ID). Second, we test our models on the held out IWSPA-AP emails (2,634 legitimate and 288 phishing) and the FRAUD emails (1,795 phishing). These emails are substantially different from the ones we used for our models' initial and task-specific fine-tuning. For instance, the fraud emails are entirely composed of texts from a person pretending to be a "Nigerian Prince" in distress—these emails were completely missing from our fine-tuning and classification training sets. Similarly, the IWSPA-AP legitimate emails contain messages from sources like the Democratic National Committee, Sony, etc., which were also missing from the training set, whose legitimate portion comprised of only corporate emails. We refer to the combined corpus of these emails as **out of domain** (OOD). Models that overfit on their training domain would struggle on these emails, likely due to distribution shift. Furthermore, prior work [26] has pointed out the importance of considering *data-difficulty* in phishing detection research. We consider difference in domain as one dimension of difficulty, as we expect language model based classifiers to struggle when their input is sampled from a domain different from the ones they have encountered or memorized during training [30]. We shed on the models' ability to be robust to domain changes by evaluating their performance on OOD set.

We compare our models to a number of baselines: First, we use features that were found to be informative and effective in phishing research prior to the proliferation of deep learning based approaches. We use the `Phishbench` framework [4] to extract and use these features in a number of statistical learning models: logistic regression, SVM, Random Forests, Decision Trees, and XGBoost. We additionally employ the use of three different feature selection method: information gain, gini-ratio, and chi-square. Out of all model and feature selection ablations, the XGBoost model, combined with an information-gain criterion for feature selection achieves the best performance on the validation set. Next, we use the `fasttext` classifier [31], which has proven to be a strong baseline in a number of sequence classification tasks. This model uses the `fasttext` word embeddings [32] for word bigrams,[8] which are summed and passed as input features to a simple feed-forward network which is trained on the training set for 20 epochs. Finally, we use the BERT-base model [2], which we fine-tune on the training set using the same method as we did for fine-tuning PHISHLM. We expect the fine-tuned BERT baseline to achieve performance that is upper-bounded by the fine-tuned PHISHLM$_{small}$ as PHISHLM$_{small}$ undergoes two rounds of fine-tuning – first to learn the generative distribution of emails, and second to learn to classify emails. Furthermore, both of these models have the same number of parameters, and only differ in the *initial* task that they were trained on, which—at a high level—boils down to learning the distribution of words in English.

IV. RESULTS AND ANALYSES

A. Language Modeling Results

We use the perplexity per word metric to evaluate our LMs on their ability to model text sequences used in email communication, following a long precedent in LM literature [20], [33], [34]. This measure is given by:

$$ppl_{word} = \exp\{-\frac{1}{n}\sum_{t=1}^{n} \log p(w_t \mid w_1, \ldots, w_{t-1})\}, \quad (5)$$

where w_t is the t-th token in a sequence, and n is the total number of words in the corpus. Perplexities can be interpreted as a measure of how well a model estimates the probability of a sample—a low perplexity indicates that the model can correctly estimate the sample in a few number of tries. Therefore, models with lower perplexities are better.

[8] we also experimented with other n-grams and found bigrams to perform the best.

Model	Params	Setting	Test	IWSPA	FRAUD	Test-1.5k
PHISHLM$_{small}$	124M	Scratch	12.23	52.56	42.17	17.78
PHISHLM$_{medium}$	355M	Scratch	10.19	27.09	24.56	15.56
PHISHLM$_{small}$	124M	Fine-tuned	8.66	50.28	30.04	12.61
PHISHLM$_{medium}$	355M	Fine-tuned	**7.22**	**22.75**	**16.89**	**10.68**

TABLE III: Perplexities per word obtained by various GPT2 models across training from scratch and fine-tuning settings on three different held-out datasets.

Using this measure, we compare our PHISHLM models in two settings: when they are trained from scratch—i.e., initialized randomly before training on emails, and when they are fine-tuned on the training set—i.e., initialized with the weights of the pre-trained GPT2 model. We report perplexities per word on the 10% of the fine-tuning data that we held out prior to training, as well on IWSPA-AP and FRAUD, which contain emails whose domains are different from the the models' training domain. We additionally also report results on the testing set of the phishing detection task which contains a similar amount of emails (1.5k) as the OOD datasets. We do this to provide roughly level-ground comparisons with the perplexities on our out-of-domain datasets, which are much smaller in sample size as compared to the 80.5k emails in the fine-tuning test set. Our results are shown in Table III.

Table III suggests several important conclusions. First, we see that models that are initialized with parameters that were learned during pre-training (i.e., the fine-tuning setting, rows 2 and 3 in the table) perform substantially better than models that are not, suggesting that the pre-training process has provided our models with the inductive bias necessary to represent general statistical information about English. That is, models that are fine-tuned begin their training on emails from a more desirable initial point compared to models that are trained from scratch. Next, perplexity per word decreases with size: PHISHLM$_{medium}$ consistently obtains lower—i.e., better—perplexity than PHISHLM$_{small}$ on all four datasets, regardless of the training setting. Finally, we see that the perplexities of the models on emails outside of their training domain (IWSPA, FRAUD) are greater than those that are in-domain (Test, Test-1.5k), highlighting the general tendency of models to overfit their training data. However, since the perplexities of models starting with pre-trained parameters is far better than for models that do not, we conclude that pre-training makes models more conducive to capturing domains outside of training experience. Since fine-tuned models perform substantially better than ones trained from scratch, we only use them in our phishing detection experiments.

B. Phishing Detection

In this section, we present results on evaluating the various baselines and the PHISHLMs in performing phishing detection. Since all our evaluation datasets are imbalanced (see Table II), we use matthews correlation coefficient (MCC) [35] as our evaluation metric. It is mathematically defined as:

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}},$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. MCC ranges from -1 (worst) to 1 (best), and is 0 for majority class baselines—where the most frequent label in the training set is used as the prediction for all test items. We compare the MCC scores of all our models on the validation set, the in-domain test set, and the out of domain (OOD) test set. Table IV shows our results.

Model	Val (ID)	Test (ID)	OOD
Baselines			
`PhishBench`	0.86	0.87	0.78
`fasttext`	0.91	0.91	0.77
BERT-base	0.95	0.94	0.72
Fine-tuned Classifiers			
PhishLM$_{small}$	0.98	0.97	0.73
PhishLM$_{medium}$	**0.99**	0.97	0.73
In-context Priming			
PhishLM$_{small}$	0.98	0.98	0.78
PhishLM$_{medium}$	**0.99**	**0.99**	**0.84**

TABLE IV: Performance (MCC) of the tested models on Validation (ID), Test (ID), and OOD evaluation sets. Values in **bold**-face indicate best performance along columns.

Overall, all models perform remarkably well on the in-domain evaluation sets, highlighting their prowess at learning the desired features and transformations that aid in distinguishing phishing emails from legitimate emails. However, their performance declines on the out-of-domain set, indicating a gap between the models' ability to represent and process samples that are similar to the ones they have encountered in training, and their flexibility to show robustness when applied to samples that are different from their training experience. Our fine-tuning and in-context priming approaches substantially outperform all three baselines that we consider in the in-domain evaluation sets, re-affirming the view that LM-based fine-tuning is likely to be strongest at approximating the training distribution. In particular, our priming-based approach achieves the best results in all three evaluation settings—nearly reaching perfect performance in the in-domain setting, and showing substantial robustness on the out-of-domain set relative to the baselines. This is especially the case in the largest model we consider, PHISHLM$_{medium}$, which improves over the best baseline by 4 percent points in the in-domain setting and 6 percent points in the out of domain setting. While our fine-tuning based approach show similar improvements on in-domain sets, it appears to have strongly overfit the training distribution, and therefore achieves modest performance on the OOD set. However, we do see that it slightly improves over the BERT-base baseline (3 points in the in-domain sets and 1 point on the out of domain set), suggesting the benefits of in-domain fine-tuning using the LM task, prior to classification-based fine-tuning [24]. Interestingly, our statistical learning baseline (the XGBoost model trained on features selected using information-gain) outperforms the `fasttext` and the BERT-base baselines, as well as our fine-tuned PHISHLMs on the out-of-domain set, highlighting the importance of

considering semi-automatically engineered features for general phishing detection.

Why does the in-context priming approach substantially out-perform the fine-tuning approach in phishing detection on out-of-domain emails? We conjecture that this is because the priming approach is faithful to PHISHLM's original task—predicting words in context. Recent work has shown that LMs such as the ones we have used here have been found to generate novel text content [36], e.g., generating well-formed novel words, and sentences that have syntactic structures different from the ones encountered in training. In our case, this suggests that they have some capacity to represent novel emails, albeit this could be weaker as compared to their representational priors for emails in their primary training domain. This is also shown by the substantial gap between the perplexities of models that were initialized by random and pre-trained weights (see Table III), suggesting that pre-training on general English domain provides a useful inductive bias to flexibly capture the domains represented in emails, even beyond simply training on a large corpus of emails. In contrast, even though fine-tuning achieves near-perfect classification performance on emails in the training domain, the models still show weakened performance on out-of-domain emails. This suggests that the representations learned as a result of the transformation induced by fine-tuning overfit the domain reflected in the training data. This lack of robustness in out-of-domain data could be due to *catastrophic interference* [37], where the LMs "forget" about the distributional information about general English (which helps in capturing some out-of-domain information, as discussed above) when they are trained to perform the phishing detection task.

While our priming approach shows noteworthy improvements over the baselines, the statistical learning approach of the final `Phishbench` model appears to be far more robust than the other baselines on the out-of-domain dataset. This has important implications. First, it highlights the importance of features that are grounded in prior phishing and social engineering research. Second, and perhaps more importantly, these features are primarily lexical in nature – they operate over words, or at most four n-grams at a time, as opposed to full length email representations that our language models utilize. Despite this lack of message-level—as opposed to word-level—processing, they are able to show robust out of domain performance. This suggests that the datasets we use, most of which are common throughout phishing detection literature, are not entirely well suited for message-level classification. Phishing emails in most corpora often contain deceptive bank account notifications, messages about lotteries, fake offers, etc. Legitimate emails on the other hand contain corporate emails, private messages between family members and friends, etc.—i.e., emails that are vastly different from the ones found in phishing attempts. Therefore it should not come as a shock that all our models achieve near-perfect performance, since it is expected that they are highly sensitive to lexical items, which is exactly what the models in `Phishbench` are trained to do, but with far fewer parameters as compared to the LMs we utilize. It is therefore worthwhile to pursue more difficult phishing detection benchmarks in the future, where legitimate and phishing emails are topically similar and are not easily distinguishable by simply comparing word-level features.

V. CONCLUSION

We have presented an empirical analyses on the application of adapting pre-trained language models (LMs) to perform phishing detection. To this end, we propose two new LMs: PHISHLM$_{small}$, and PHISHLM$_{medium}$, GPT2 models which have been adapted on 725k legitimate and phishing emails. Using these models, we propose two approaches to perform phishing detection, and find both of them to out-perform the baselines, when the evaluation dataset was drawn from the same domain as our models' training data. While all models performed worse on emails that were outside of their training domain, our priming approach proved to be substantially robust compared to other approaches, demonstrating the importance of pre-training and the language modeling task in order to classify out-of-domain emails. Since our models are auto-regressive in nature, they can be used to perform natural language generation. In future work, we hope to analyze our models on their ability to generate well-formed emails, to better understand the extent to which they can perform phishing, thereby aiding in threat-analyses to mitigate social engineering, beyond traditional classification approaches.

REFERENCES

[1] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *ACL 2018*, Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 328–339.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL 2019*, Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186.

[3] A. Das, S. Baki, A. El Aassal, R. Verma, and A. Dunbar, "Sok: A comprehensive reexamination of phishing research from the security perspective," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 671–708, 2019.

[4] A. El Aassal, S. Baki, A. Das, and R. M. Verma, "An in-depth benchmarking and evaluation of phishing detection research for security needs," *IEEE Access*, vol. 8, pp. 22 170–22 192, 2020.

[5] S. Salloum, T. Gaber, S. Vadera, and K. Shaalan, "Phishing email detection using natural language processing techniques: A literature survey," *Procedia Computer Science*, vol. 189, pp. 19–28, 2021.

[6] R. Verma, N. Shashidhar, and N. Hossain, "Detecting phishing emails the natural language way," in *European Symposium on Research in Computer Security*, Springer, 2012, pp. 824–841.

[7] R. Verma and N. Hossain, "Semantic feature selection for text with application to phishing email detection," in *international conference on information security and cryptology*, Springer, 2013, pp. 455–468.

[8] A. Bergholz, J. H. Chang, G. Paass, F. Reichartz, and S. Strobel, "Improved phishing detection using model-based features.," in *CEAS*, 2008.

[9] G. Egozi and R. Verma, "Phishing email detection using robust nlp techniques," in *ICDMW 2018*, IEEE, 2018, pp. 7–12.

[10] G. Park and J. Rayz, "Ontological detection of phishing emails," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2018, pp. 2858–2863.

[11] N. Q. Do, A. Selamat, O. Krejcar, E. Herrera-Viedma, and H. Fujita, "Deep learning for phishing detection: Taxonomy, current challenges and future directions," *IEEE Access*, 2022.

[12] Y. Fang, C. Zhang, C. Huang, L. Liu, and Y. Yang, "Phishing email detection using improved rcnn model with multilevel vectors and attention mechanism," *IEEE Access*, vol. 7, pp. 56 329–56 340, 2019.

[13] Y. Lee, J. Saxe, and R. Harang, "Catbert: Context-aware tiny bert for detecting social engineering emails," *arXiv preprint arXiv:2010.03484*, 2020.

[14] P. Bountakas, K. Koutroumpouchos, and C. Xenakis, "A comparison of natural language processing and machine learning methods for phishing email detection," in *ARES*, 2021, pp. 1–12.

[15] C. N. Gutierrez, T. Kim, R. Della Corte, *et al.*, "Learning from the ones that got away: Detecting new forms of phishing attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 988–1001, 2018.

[16] D. Jurafsky and J. H. Martin, *Speech & Language Processing, 3rd Edition*. 2020.

[17] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[18] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *NeurIPS 2017*, 2017, pp. 5998–6008.

[19] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," Open AI, Tech. Rep., 2018.

[20] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[21] T. B. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[22] Y. Liu, M. Ott, N. Goyal, *et al.*, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv preprint arXiv:1907.11692*, 2019.

[23] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding," in *ICLR*, 2018.

[24] Y. Gu, R. Tinn, H. Cheng, *et al.*, "Domain-specific language model pretraining for biomedical natural language processing," *ACM Transactions on Computing for Healthcare (HEALTH)*, vol. 3, no. 1, pp. 1–23, 2021.

[25] S. Gururangan, A. Marasović, S. Swayamdipta, *et al.*, "Don't stop pretraining: Adapt language models to domains and tasks," in *ACL 2020*, Online: Association for Computational Linguistics, Jul. 2020, pp. 8342–8360.

[26] R. M. Verma, V. Zeng, and H. Faridi, "Data quality for security challenges: Case studies of phishing, malware and intrusion detection datasets," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2605–2607.

[27] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *International Conference on Learning Representations*, 2018.

[28] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *arXiv preprint arXiv:2107.13586*, 2021.

[29] K. Misra, A. Ettinger, and J. Rayz, "Exploring BERT's sensitivity to lexical cues using tests from semantic priming," in *Findings of EMNLP 2020*, 2020, pp. 4625–4635.

[30] T. McCoy, E. Pavlick, and T. Linzen, "Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference," in *ACL 2019*, Florence, Italy: Association for Computational Linguistics, 2019, pp. 3428–3448.

[31] A. Joulin, É. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *EACL*, 2017, pp. 427–431.

[32] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *TACL*, vol. 5, pp. 135–146, 2017.

[33] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[34] C. Chelba, T. Mikolov, M. Schuster, *et al.*, "One billion word benchmark for measuring progress in statistical language modeling," *arXiv preprint arXiv:1312.3005*, 2013.

[35] B. W. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et Biophysica Acta (BBA)-Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.

[36] R. T. McCoy, P. Smolensky, T. Linzen, J. Gao, and A. Celikyilmaz, "How much do language models copy from their training data? evaluating linguistic novelty in text generation using RAVEN," *arXiv preprint arXiv:2111.09509*, 2021.

[37] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, vol. 24, Elsevier, 1989, pp. 109–165.