PhishCoder: Efficient Extraction of Contextual Information from Phishing Emails

Tarini Saka[1][0000−0003−3936−6766], Kami Vaniea[2][0000−0001−8042−3342], and Nadin Kökciyan[1][0000−0002−2653−6669]

[1] School of Informatics, University of Edinburgh, Edinburgh, UK
tarini.saka@ed.ac.uk, nadin.kokciyan@ed.ac.uk
[2] University of Waterloo, Waterloo, Canada
kami.vaniea@uwaterloo.ca

Abstract. Phishing has emerged as one of the most widespread and expensive types of cyber threats, posing significant challenges to individuals and organizations worldwide. In response to the evolving threat, security workers are integrating AI and ML algorithms to mitigate phishing attacks automatically. Although email text is a crucial element in identifying phishing attacks, traditional text-embedding techniques face challenges due to variations in text length, structure, and the inability to capture context effectively. In this paper, we introduce *PhishCoder*, a novel framework designed to extract contextual information from phishing emails. Our focus is on human-centric features, which are often overlooked in traditional approaches, as they are the features that users notice when evaluating a potentially suspicious email. By fine-tuning four transformer-based models, we accurately extract seven descriptive features from phishing email texts. Our findings indicate that language models provide a promising method for extracting contextual information from phishing emails. This approach offers researchers and security workers a whole new set of features that would be valuable in combating phishing attacks and developing effective mitigation tools.

Keywords: Phishing, Context, Language Models, Email Security, Human-Centered Artificial Intelligence

1 Introduction

Phishing is a cyber-attack where an attacker sends emails impersonating reputable sources to deceive recipients into providing sensitive information such as usernames, passwords, or credit card numbers. Phishing is one of the most pervasive and expensive types of cyber threats, posing significant challenges to individuals and organizations worldwide. Statistics highlight the magnitude of the phishing epidemic, with reports indicating a relentless surge in both the frequency and complexity of such attacks. The Anti-Phishing Working Group (APWG) observed almost five million phishing attacks in 2023, the worst year for phishing on record [4]. IBM's Cost of a Data Breach 2022 report [12] rates phishing as

the second-most common cause of data breaches (up from fourth the previous year) and the most expensive, costing victims $4.91 million on average.

In response to this escalating threat, security workers across the world are trying to develop effective cybersecurity solutions, including integrating Artificial Intelligence (AI) and Machine Learning (ML) algorithms. An approach that shows promise. According to an IBM research survey, organizations that used AI and automation in their approach experienced, on average, a 108-day shorter time to identify and contain a breach and also reported $1.76 million lower data breach costs compared to organizations that did not use such capabilities [12]. Khonji *et al.* [15] provide a high-level overview of various categories of phishing mitigation techniques. They reviewed four types of software detection solutions for phishing: blacklists, rule-based heuristics, visual similarity, and machine-learning-based classifiers. They found that machine-learning-based detection techniques achieve high accuracy in analyzing similar data parts compared to rule-based heuristic techniques.

Despite their advantages ML algorithms face specific challenges, notably in selecting the appropriate feature set and feature representation. In the case of phishing emails, the appropriate feature set requires consideration of all the email parts such as the *header*, *body*, *attachments*, and *URLs* included in the email body. The *body text* is especially crucial for phishing mitigation as it provides the context for the email. Several studies have demonstrated the efficiency of using email body text to enhance the performance of phishing classifiers [1,31,26,2,30]. Recently there has been a shift towards using advanced text embedding techniques to extract contextual and semantic information from email text for machine learning tasks like classification and clustering [26,27,6,29]. However, the unstructured nature of email body text and the variation in language, grammar, length, and layout pose significant challenges for machine interpretation. Efficiently representing the text can be difficult due to this variation. Text-embedding techniques face common problems such as the absence of sufficient context, the presence of generic text, and implied threats or actions. Furthermore, while text embeddings can effectively capture semantic relationships, they operate as "black boxes", making it difficult to understand how they arrive at specific decisions. This opacity can be problematic when justifying the reasons behind flagging an email as potential phishing and limits error diagnoses.

To address the range of challenges, it is necessary to establish a more concise and standardized representation of phishing email text that prioritizes human-oriented features to enhance explainability. We propose a hybrid approach that utilizes language models to extract and summarize contextual information into a concise, interpretable format, which then serves as input for algorithms. This method not only improves the representation of the email text but also introduces a layer of explainability to the system. The final classification or clustering decisions are based on a set of human-interpretable features, making it easier to understand and justify the reasons behind flagging an email as malicious or benign.

In this paper, we investigate the use of pre-trained language models to extract categorical and descriptive features from phishing emails. To achieve this, we propose the *PhishCoder* framework, which combines text classification and text extraction tasks to obtain the categorical features. Based on our previous research [25], we included seven categorical features in PhishCoder that effectively define the context of an email. These features were identified based on the factors that influence human decision-making when assessing suspicious emails. We use a small set of labeled data (∼500 emails) to fine-tune four transformer-based language models to achieve these tasks. Our findings indicate that language models offer a promising method for extracting contextual information from phishing emails, and could be valuable in combating phishing attacks.

In summary, we make the following contributions:

1. We introduce *PhishCoder* to capture the contextual nature of phishing emails by considering human-centric features.
2. By using real-world datasets, we conducted experiments with four pre-trained language models to demonstrate their effectiveness in extracting categorical features that represent phishing emails in an interpretable way.
3. We provide a comparative analysis of the four models for information extraction. Additionally, we present a fine-tuned multi-task classifier to simultaneously perform the four text classification tasks.

The outline of the paper is as follows. Section 2 presents a background and the related work on the topic. Section 3 describes in detail our motivation and use-cases for developing PhishCoder. Section 4 presents the sevev categorical and descriptive features extracted by PhishCoder. Section 5 outlines the proposed approach and explains each step in the fine-tuning process. We discuss the results of the experiments in Section 6. Furthermore, in Section 7 we present a multi-headed classification model we trained to perform all four classification tasks simultaneously. We conclude with a conclusion, and future work 8.

2 Background and Related Work

In this section, we introduce the relevant concepts about language models and tools for information extraction and discuss related work on the topic.

2.1 Phishing Email Text

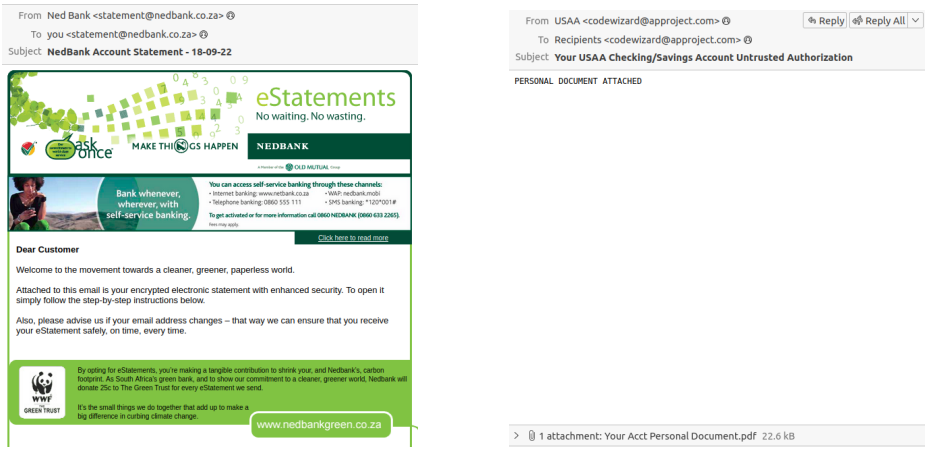
Phishing email text plays a crucial role in detecting and mitigating phishing attacks as it contains information or context about the underlying scam. Security systems often analyze email content to recognize keywords and phrases commonly associated with phishing, such as urgent calls to action or requests for sensitive information. Many *phishing email detection* techniques use text-based features and machine learning algorithms to distinguish phishing from legitimate emails [6,14,33,34]. Salloum *et al.* [28] conducted a systematic literature review of 100 research articles published between 2006 and 2022 on phishing

email detection using NLP techniques and found that TF-IDF (term frequency-inverse document frequency) and word embeddings are the most frequently used text representation techniques. Email text also plays a crucial role in identifying *phishing email campaigns*, groups of emails sent from a common source as part of a mass attack with similar tactics and objectives [26,3,11]. Analyzing the language, content, and structure of emails can help detect recurring patterns and consistent themes. Furthermore, email text is the most commonly used input feature for *profiling phishing* attacks [10,36]. Profiling involves the examination of phishing emails to understand the modus operandi of attackers, allowing for the creation of profiles to better detect future phishing attempts.

Although phishing email text is a crucial feature, it does pose certain challenges to security researchers. The inherently unstructured nature of email body text, coupled with significant variations in language, grammar, and layout, presents significant challenges for machine interpretation. Traditional text-embedding techniques like TF-IDF face problems such as the absence of sufficient context 1b, the presence of generic text 1a, implied threats or actions rather than explicit text, and variation in the length and structure of the text 1. Automated systems tasked with analyzing email content must contend with these complexities. For instance, Figure 1 shows two phishing emails, taken from the Nazario Phishing Corpus [21]. Both emails claim to be from financial organizations about a statement or problem, and both ask the user to download an attachment. However, the first figure 1a is a sophisticated phishing attempt claiming to be from Nedbank, a South Africa-based bank. It has a well-designed layout with the right logos and references to legitimate organizations like the World Wildlife Fund. The wording and URLs appear to be legitimate and contain no obvious signs of malicious intent. In contrast, the second email 1b claims to be from USAA, a United States-based financial organization, but it has minimal text and no useful information or context, again posing a challenge for the machines. With advancements in NLP, there has been a recent shift towards using word embedding techniques to extract contextual and semantic information from email text for machine learning tasks like classification and clustering [34,27,6,29].

2.2 Information Extraction

Information Extraction (IE) in NLP involves automatically extracting structured information, such as entities, relationships, and events, from unstructured text [7]. IE techniques are crucial for transforming raw text into usable data for various applications, such as knowledge base construction and semantic search. Such methods have been previously used for other email related tasks. Laclavík *et al.* explore a lightweight approach to enterprise email communication analysis and information extraction [16]. They tested their approach on several small and medium enterprises (SMEs) and claimed that it was promising for enterprise interoperability and collaboration in SMEs that depend on emails to accomplish their daily business tasks. Mahlawi *et al.* proposed a novel process to extract structured data from emails pertaining to a domain, involving three phases: data cleaning, extraction (including keyword, sentiment, and entity extraction),



(a) Longest Email (b) Shortest Email

Fig. 1: An example of the variation in email structure observed in our dataset.

and consolidation, to facilitate easier management and analysis of knowledge for better decision-making in large industries [20]. Listík et al. [18] developed an algorithm that utilizes named entity recognition (NER) to detect company names and compare URLs in the email content against a company URL profile built from historical legitimate traffic. Their method achieves high accuracy on live email traffic.

2.3 Role of language models

The recent advancements in Natural Language Processing (NLP) and Language Models have created a significant evolution in the field of information extraction. Language models (LMs) are computational models designed to understand, generate, and manipulate human language. They learn the statistical patterns and structures in a corpus of text, enabling them to predict the likelihood of a sequence of words [5]. *Transformers*, a breakthrough architecture in language models, leverages self-attention mechanisms to process words in parallel, leading to significant improvements in tasks such as translation, summarization, and text generation [32]. Transformer-based architectures such as BERT (Bidirectional Encoder Representations from Transformers) [9] and GPT (Generative Pre-trained Transformer) [22] have achieved unprecedented levels of accuracy and contextual understanding. The fine-tuning capabilities of these models allow for customization to specific tasks and domains, further enhancing their applicability in information extraction tasks across various industries and domains. For instance, Dagdelen *et al.* demonstrate how pretrained large language models (GPT-3, Llama-2) can be fine-tuned to extract structured knowledge from records of complex scientific text [8]. Sage *et al.* demonstrate that language models like LayoutLM, a pre-trained model recently proposed for encoding 2D

documents, are effective for extracting information from business documents. Their model achieves high sample efficiency in data-constrained settings, reaching over 80% performance with as few as 32 documents for fine-tuning [24].

3 Motivation

When given a phishing email, PhishCoder will automatically analyze the email to provide a human-readable summary that includes contextual information. This structured summary can be used by security personnel for various purposes such as automated email filtering, real-time threat detection, and improved incident response procedures. Furthermore, the features extracted by PhishCoder can be used to support advanced analytics and machine learning models, allowing for the prediction and prevention of future phishing attempts. Some potential applications of PhishCoder include:

1. Phishing Email Profiling: Phishing profiling involves analyzing and categorizing phishing attempts based on their characteristics and patterns to better detect and prevent future attacks [36]. The outputs of PhishCoder can be used to create profiles for a single phishing email or a cluster of emails. For instance, the From-Sector helps identify the industry targeted by the phishing attempt, while Action-Generic categorizes the expected recipient response and Urgency Cues flag time-sensitive elements. This enables better detection of future phishing attempts by recognizing patterns and anomalies in email text. Such profiles with a contextual summary can be very helpful for quickly processing emails by support staff. Instead of manually analyzing every word in the email, the staff can simply review the profile to handle the email faster.

2. Phishing Campaign Identification: Phishing email campaign detection involves identifying and grouping emails based on shared characteristics and patterns indicative of coordinated malicious activities [26,3,11]. Security experts believe that phishing emails from a single campaign share certain similarities, including impersonated organizations, solicited actions, and URL domains. Outputs of PhishCoder contain crucial information about the underlying scam and hence can be used to detect similarities and recognize phishing campaigns based on shared tactics and objectives.

3. Phishing Guidance or Advice Tool: A crucial step in organizational phishing management is to provide timely and suitable guidance to users to keep them safe from phishing attacks, which often use urgent or threatening language to trick users into taking dangerous actions [35,17,13]. PhishCoder outputs can be used to create well-tailored advice for users who report suspicious emails. For instance, the Action-Generic category helps users identify the types of actions requested, allowing advice to be tailored accordingly, such as cautioning against clicking on suspicious links or downloading attachments from unknown sources. Integrating these contextual features into tools like PhishEd [13] can provide users with quick and accurate support, helping them to recognize and respond effectively to phishing attempts.

4 Contextual Representation of Emails

Phishing emails typically contain various elements of information about the underlying scam spread across different parts of the email, including the body text, subject line, and other components. The body text often contains deceptive cues to prompt actions like clicking on malicious links or providing sensitive information. Urgency, fear, or rewards are used to persuade recipients to react quickly. Additionally, the subject lines are crafted to grab the recipient's attention and entice them to open the email, often using attention-grabbing phrases or alarming statements. Other parts of the email, such as sender information, logos, and formatting, may also be manipulated to appear legitimate and deceive users further. In this section, we describe the various contextual information types extracted by PhishCoder. It is important to note that the feature set used in the study was originally introduced as part of the Phishing Codebook in previous work [25]. We utilized a subset of the original codebook and modified some aspects, but the full Phishing Codebook is provided in the Appendix. Table 1 provides a summary of our feature set, a brief description of each feature, and pre-defined labels for the classification tasks. Table 2 provides sample outputs for the two emails shown in Figure 1.

Table 1: The annotation guide for PhishCoder. Comprises seven information types (codes), a brief explanation of each, and predefined labels (if any).

Task	Information Type	Explanation/Question	Labels
		Text Classification Tasks	
TC1	From – Sector	Type of sector the email claims to be from	financial (28.2%), email (42.8%), document share (5.4%), logistics (4.8%), shopping (4.6%), service provider (4.4%), government (4.8%), unknown (5.0%)
TC2	Action – Generic	The action being prompted in the email	click (86%), download (12.3%), other (1.7%)
TC3	Urgency Cues	Presence of time pressure or urgency cues	urgent (34.9%), none (65.1%)
TC4	Threatening language	Presence of threatening language, tone	threat (42.4%), none (57.6%)
		Text Extraction Tasks	
TE1	From – Company Name	Name of the organization being impersonated	N/A
TE2	Action – Specific	The reason provided to perform an action	N/A
TE3	Main Topic	Main purpose of the email	N/A

1. **From-Sector** is the type of organization the phishing email claims to be from, refers to the affiliation with a specific industry or sector. The codebook defines eight sectors in this category: financial, email, document share, logistics, shopping, service provider, government, and unknown.
2. **Action-Generic** refers to the action prompted from the recipient within the email. There are three classes, representing the types of responses or behav-

Table 2: Outputs of PhishCoder for the two emails shown in Figure 1.

Information Type	**Email 1**	**Email 2**
From-Sector	financial	financial
Action-Generic	download	download
Threat	none	none
Urgency Cues	none	none
From-Company	nedbank	usaa
Main Topic	encrypted electronic statement	personal document
Action Specific	attached to this email	document attached

iors expected from the recipient: 'click' on a link, 'download' an attachment, and 'other'.

3. **Urgency Cues** indicate whether the email contains elements of time pressure or urgency. It is classified into two categories based on the presence or absence of such cues.
4. **Threatening Language** identifies the presence of threatening language/tone within the email, indicating potential negative consequences. It is categorized into two classes based on whether such language is present or not.
5. **From-Company Name** refers to the name of the organization that the phishing email is impersonating. This information is extracted directly from the text of the email.
6. **Main Topic** signifies the primary purpose or theme of the phishing email. This information is extracted directly from the text of the email.
7. **Action-Specific** denotes the justification provided to the recipient for carrying out the specified action requested in the email. This information is extracted directly from the text of the email.

5 Methodology

Our goal is to create a pipeline for structured information extraction from phishing emails to create a standardized representation of the email context. Our approach consists of four key stages: (1) processing phishing emails, (2) creating the training dataset by applying text classification and text extraction methods, (3) fine-tuning pre-trained models for these tasks and (4) evaluating the models by using a real-world dataset, and providing a comparative analysis. Our code implementation is available in a repository[3].

5.1 Processing Phishing Emails

In the first step, we pre-process the phishing emails, a critical step to ensure the data is formatted suitably for subsequent analysis.

[3] `https://git.ecdf.ed.ac.uk/s2138664/secai_phishcoder`

Training Dataset We use the annotated dataset from previous work [25]. This dataset is a subset of the *Nazario Phishing Dataset*[1], which is a publicly available collection of hand-screened phishing messages collected by Jose Nazario [21]. It is the most well-known *publicly-available* phishing email dataset and has been previously used in multiple works for phishing detection and classification [6,26,37]. The author published a collection of 1916 phishing emails collected between 2015 and 2021, where all emails collected in a single year were published together in a single plain text file (i.e., mbox). The mbox file for each year (2015-2021) was downloaded from the original site. Any emails with empty content (201) and non-English content (114) were removed, resulting in a dataset of 1,688 processed emails. They then randomly sampled around 70 emails from each year (2015-2021) to create a dataset of 503 emails.

In this study, we used a subset of the original annotated dataset comprising 490 emails (D1) and added 31 emails from a new dataset collected from one department in a university in the United Kingdom (D2). Students and staff were invited to "donate" phishing emails they received by forwarding them to a research inbox. We obtained ethics approval from the institution for this data collection. The additional emails were added because the original dataset (D1) had very few emails in some sectors ('shopping', 'government', and 'service provider') under the 'From-Sector' category, which would have caused issues when training models. We manually sampled D2 for more emails in these sectors, resulting in a final dataset of 521 emails (D). Since D2 is a private dataset, it cannot be shared with the public in the repository.

5.2 Creating Labelled Dataset

Following the pre-processing phase, we create a labeled training dataset. This dataset serves as the input for fine-tuning our language models. To implement PhishCoder, we designed four text classification tasks for the features *From-Sector, Action-Generic, Threatening Language, and Urgency Cues*. Additionally, we formulated three text extraction tasks aimed at identifying *From-Company Name, Action-Specific, and the Main Topic* of the emails. Table 1 provides a summary of the information types, a brief explanation of each, and available labels for the text classification tasks.

The inputs for fine-tuning language models need to be in a specific format. We utilized *Label Studio*[4] for the annotation process to obtain labels in the required format. Label Studio is an open-source data labeling tool known for its versatility and user-friendly interface. Specifically, we used it for both text classification and text extraction. To perform the labeling, we combined the subject line with the email body text.

Text classification tasks. Text classification is a process in natural language processing that involves categorizing text into predefined labels or categories

[1] `https://monkey.org/~jose/phishing/`
[4] `https://labelstud.io/`

based on its content. It was the suitable choice for these four tasks as they have a predefined set of codes. We have four text classification tasks; *TC1: From-Sector, TC2: Action-Generic, TC3: Urgency Cues, TC4: Threatening Language*. TC1 has 8 class, TC2 has 3 classes, TC3 and TC4 have 2 classes. All four tasks have class imbalances as shown in the 'Labels' column of Table 1.

Text extraction tasks. We have three text extraction tasks; *TE1: From-Company Name, TE2: Action-Specific, TE3: Main Topic*, that we implement using Question Answering models. A Question Answering (QA) model in NLP identifies and extracts the exact segment of text from a given passage that directly answers a posed question. Such a model is a good choice for these three tasks because they involve extracting a span of text from the email. The questions framed for these three tasks are shown below:

1. **Main Topic:** What is the primary purpose or main topic of the email based on the provided description?
2. **Action - Specific:** What is the specific action requested from the receiver of the email?
3. **From - Company Name:** What is the name of the organization that the email is from?

Train, Validation, Test sets. For each task, we exclude any data points with missing labels or answers. Subsequently, we partition the dataset to train the models, allocating 80% of the labeled data for our training set, 10% for validation, and the remaining 10% for the test set.

5.3 Experiments and Models

Choosing the right model to fine-tune requires careful consideration of factors such as model size, computational resources, and task compatibility. We experiment with two transformer-based architectures: BERT [9] and RoBERTa [19]. Both are pre-trained models on the English language using a masked language modeling (MLM) objective and are primarily aimed at being fine-tuned on tasks that use the whole sentence (potentially masked) to make decisions, such as sequence classification, token classification, or question answering. These model choices are ideal as they strike a balance between computational efficiency and performance, with manageable model sizes that are feasible to fine-tune on standard resources. Both BERT and RoBERTa have been successfully fine-tuned for various tasks, demonstrating their versatility and effectiveness in handling diverse NLP challenges. Furthermore, they can be fine-tuned for both text classification and question-answering tasks, making it a good choice for this study. We used two variations of these models: bert-base-uncased[5](Model size: 110M

[5] `https://huggingface.co/google-bert/bert-base-uncased`

params), bert-large-uncased[6] (Model size: 336M params), roberta-base[7] (Model size: 125M params), and roberta-large[8] (Model size: 355M params).

6 Evaluation

In this section, we present the results of the experiments.

6.1 Text Classification Tasks

We measure the efficiency of the fine-tuned text classification models using the standard evaluation metrics: accuracy, precision, recall, and F1 score. All metrics used are scaled from 0 to 1, with higher values indicating better performance. The results for the F1 Score, precision (P), and recall (R) provided below are the weighted average. It is a way to combine the per-class scores into a single metric that accounts for the different sizes of the classes. This is particularly useful when you have an imbalanced dataset, where some classes have many more instances than others.

Action-Generic Classification. Table 3 provides a summary of the classification results for the 'Action-Generic' task. All models achieved high accuracy, with BERT-base and RoBERTa-base scoring approximately 92.45% and RoBERTa-large slightly higher at 94.34%. The F1 scores also reflected strong performance for the majority class (class 0) and reasonable performance for the minority class (class 1), with scores between 0.90 and 0.93. However, all models struggled significantly with class 2, which had only one sample in the validation set. This resulted in zero precision, recall, and F1-score for class 2 across all models, highlighting a critical issue of class imbalance and insufficient data representation.

From-Sector Classification. Table 4 provides a summary of the classification results for the 'From-Sector' task. The BERT-base model achieved an accuracy of 84.91% and an F1 score of 0.830, indicating a decent performance but with room for improvement. The BERT-large model showed a slight improvement with an accuracy of 86.79% and an F1 score of 0.869. RoBERTa-base significantly outperformed both BERT models with an accuracy of 94.34% and an F1 score of 0.943. RoBERTa-large further improved the results, achieving the highest accuracy of 96.23% and an F1 score of 0.965. However, all models consistently struggled with class 1 ('service provider') and class 4 ('shopping'). BERT-base and BERT-large models failed to identify class 1 accurately, resulting in zero precision, recall, and F1-score. Similarly, the RoBERTa-base and RoBERTa-large models had difficulties with some minority classes, although they showed

[6] `https://huggingface.co/google-bert/bert-large-uncased`
[7] `https://huggingface.co/FacebookAI/roberta-base`
[8] `https://huggingface.co/FacebookAI/roberta-large`

better performance overall. The issue with class 1 can be attributed to the class imbalance in the dataset, where minority classes have significantly fewer samples compared to the majority classes. This imbalance makes it challenging for the models to learn and correctly classify the underrepresented classes.

Table 3: Evaluation of Action-Generic

Model Name	P	R	F1	Acc
bert-base	0.91	0.92	0.91	0.92
bert-large	0.93	0.94	**0.93**	**0.94**
roberta-base	0.91	0.92	0.92	0.92
roberta-large	0.93	0.94	0.93	**0.94**

Table 4: Evaluation of From Sector

Model Name	P	R	F1	Acc
bert-base	0.85	0.85	0.83	0.85
bert-large	0.88	0.87	0.87	0.87
roberta-base	0.96	0.94	0.94	0.94
roberta-large	0.98	0.96	**0.96**	**0.96**

Threatening Language Classification. Table 5 provides a summary of the classification results for the 'Threatening Language' task. The BERT-base and BERT-large models both achieved an accuracy of 83.33% with F1 scores of 0.833 and 0.831, respectively, indicating reasonable performance but with room for improvement. Both models exhibited good precision and recall for the majority class ('none'), but slightly lower performance for the minority class ('threat'). The RoBERTa-base model significantly outperformed both BERT models, achieving an accuracy of 98.15% and an F1 score of 0.982. This indicates a superior ability to classify threats accurately, with high precision and recall for both classes. The RoBERTa-large model further improved these results, achieving perfect accuracy and an F1 score of 1.00, demonstrating its capability in handling the classification task with flawless precision and recall for both classes.

Urgency Cues Classification. Table 6 provides a summary of the classification results for the 'Urgency Cues' task. The BERT-base model achieved an accuracy of 68.52% and an F1 score of 0.664. The BERT-large model showed substantial improvement, achieving an accuracy of 92.59% and an F1 score of 0.926. This indicates a balanced and high-level performance, with both precision and recall scores around 0.89 for the minority class, showing its robustness in identifying urgent messages. The RoBERTa-base model also demonstrated strong performance, achieving an accuracy of 87.04% and an F1 score of 0.867. While slightly lower than BERT-large, it maintained high precision and recall for the majority class and improved recall for the minority class compared to BERT-base. The RoBERTa-large model further improved results, achieving an accuracy of 88.89% and an F1 score of 0.887, showing balanced performance with both high precision and recall for both classes. These findings highlight the significance of model architecture and size, with larger and more advanced models like BERT-large and RoBERTa-large providing superior performance, particularly in handling class imbalance.

Table 5: Evaluation of Threat Language

Model Name	P	R	F1	Acc
bert-base	0.83	0.83	0.83	0.83
bert-large	0.83	0.83	0.83	0.83
roberta-base	0.98	0.98	0.98	0.98
roberta-large	**1.00**	**1.00**	**1.00**	**1.00**

Table 6: Evaluation of Urgency Cues

Model Name	P	R	F1	Acc
bert-base	0.67	0.69	0.66	0.68
bert-large	0.93	0.93	**0.92**	**0.93**
roberta-base	0.87	0.87	0.87	0.87
roberta-large	0.89	0.89	0.89	0.89

6.2 Text Extraction Tasks

The performance evaluation of question-answering models for text extraction tasks relies on a set of well-established evaluation metrics, primarily derived from the Stanford Question Answering Dataset (SQuAD) benchmark [23]. We use two metrics recommended by HuggingFace: Exact Match and F1 Score. The exact match metric quantifies the proportion of predictions that exactly match the ground truth answer spans. It provides a stringent measure of model accuracy, requiring precise alignment between predicted and true answers. The F1 score in the context of text extraction evaluates the model's performance by considering the overlap between predicted and true answer spans. It balances precision and recall, providing a holistic measure of model effectiveness. The values reported for both these measures range on a 0 to 1 scale, with higher values indicating better performance.

Table 7 reports the results for extracting the **Main Topic** from the email text. For this task, roberta-base achieved the highest scores with an Exact Match of 0.40 and an F1 score of 0.71, outperforming both the BERT and roberta-large models. Table 8 reports the results for extracting the **Action-Specific** from the email text. For this task, bert-base-uncased demonstrated superior performance with an Exact Match of 0.58 and an F1 score of 0.78, whereas roberta models lagged behind with lower scores, particularly roberta-large with an Exact Match of 43.40 and an F1 score of 70.37. For the **From - Company Name** task, reported in Table 9, bert-large-uncased emerged as the best performer with both an Exact Match and F1 score of 0.90, surpassing the other models which scored slightly lower, with roberta-large scoring the least with an Exact Match of 0.85 and an F1 score of 0.87. These results indicate that language models fine-tuned for QnA can play a crucial role in text extraction from phishing emails. Although, lower than the text classification results, we still see promising results. We could improve performance with more training data.

6.3 Results

The results from our experiments highlight the potential of fine-tuned language models to create stronger phishing mitigation tools. The experiments, conducted with multiple models (bert-base-uncased, bert-large-uncased, roberta-base, and roberta-large), show that fine-tuning allows these models to adapt to the specific nuances of phishing emails, thereby improving their accuracy and reliability.

Table 7: Evaluation of Main Topic

Model Name	Exact Match	F1
bert-base	0.38	0.69
bert-large	0.30	0.68
roberta-base	**0.40**	**0.71**
roberta-large	0.32	0.69

Table 8: Evaluation of Action-Specific

Model Name	Exact Match	F1
bert-base	**0.58**	**0.78**
bert-large	0.47	0.77
roberta-base	0.49	0.74
roberta-large	0.43	0.70

Table 9: Evaluation of From - Company Name

Model Name	Exact Match	F1
bert-base	0.88	0.88
bert-large	**0.90**	**0.90**
roberta-base	0.88	0.88
roberta-large	0.85	0.87

For instance, roberta-base excelled in extracting main topics, while bert-base-uncased and bert-large-uncased were particularly effective in action-specific and company name identification tasks, respectively. The ability of these models to accurately extract contextual information from phishing emails makes them a valuable tool, offering a significant improvement over traditional text-embedding techniques. This way we can create descriptive and standardized representations of phishing emails. The study demonstrates that integrating AI and ML through frameworks like PhishCoder can significantly bolster the capabilities of cybersecurity measures, providing a promising avenue for future research and application in phishing threat mitigation.

7 Multi-Task Classification

In this section, we introduce a multi-task classifier to investigate whether a single model could produce acceptable results in the text classification tasks described above. We use a multi-head model, which is a neural network architecture designed to perform multiple tasks or make multiple predictions simultaneously using a shared underlying representation. The main advantage of such a model is its ability to leverage the shared knowledge and features learned by the base network, improving efficiency and potentially enhancing performance across all tasks. This approach also reduces the computational overhead of training and maintaining separate models for each task, thus saving computational resources.

Design. We designed and implemented a multi-task learning model to perform all four text classification tasks simultaneously. The dataset for each task was tokenized and processed into batches for efficient training using PyTorch's DataLoader. The model architecture consists of a common backbone network that processes the input data followed by multiple output "heads," each specialized for a different task or aspect of the data. We employed a shared encoder

Table 10: Evaluation of Text Classification for Urgency Cues

Model Name	Action		Sector		Threat		Urgent	
	F1	Acc	F1	Acc	F1	Acc	F1	Acc
bert-base	0.75	0.83	0.13	0.29	**0.50**	**0.63**	**0.62**	**0.73**
bert-large	**0.84**	**0.87**	**0.28**	0.35	0.48	0.56	0.59	0.67
roberta-base	0.75	0.83	0.13	0.29	0.49	**0.63**	**0.62**	**0.73**
roberta-large	0.75	0.83	0.23	**0.40**	0.20	0.37	**0.62**	**0.73**

to extract common features and separate decoders for each task to focus on task-specific details. Each decoder produced a probability distribution over the possible classes for its respective task, and the outputs were compared with the ground truth labels to compute the loss. We used cross-entropy loss for all tasks and used the average loss over all tasks to update the model, ensuring that the gradients from each task contributed to updating the shared encoder and the task-specific decoders.

Training and Evaluation. During training, the model was optimized using the Adam optimizer, and we employed a learning rate scheduler to adjust the learning rate dynamically, aiding in better convergence. The training loop iterated over multiple epochs, where each epoch comprised iterating through the batches of each task. After each epoch, the model's state and the optimizer's state were saved, enabling checkpointing and future retraining or fine-tuning. For evaluation, we utilized separate validation datasets for each task, processed similarly to the training data. The evaluation involved computing metrics like accuracy and F1-score to quantify the model's performance on each task.

The results of the multitask model evaluation reveal varying degrees of performance across different tasks. BERT-large stands out with the highest overall F1 score of 0.55 and accuracy of 0.61. BERT-base and RoBERTa-base models exhibit similar performance, both achieving an overall F1 score of 0.50 and leading in threat and urgent tasks with F1 scores of 0.50 and 0.62, respectively, and an overall average accuracy of 0.62. RoBERTa-large, although not leading in any specific task, demonstrates a balanced performance with an overall F1 score of 0.45 and accuracy of 0.58. Notably, the 'Sector' classification task remains challenging for all models, with the highest F1 score being 0.28 from BERT-large. This could be because of the class imbalance issue. For the multi-headed classification experiments, we did not stratify the sampling of training, validation, and test sets. This means that the distribution of classes might have been uneven across the different sets, leading to potential overfitting on the training data and underperformance on the validation and test sets.

8 Conclusion

In this paper, we propose a novel framework called *PhishCoder* to extract contextual information from phishing emails using pre-trained language models.

This approach aims to create a new contextual and descriptive representation of emails using a set of human-centric features proposed in the Phishing Codebook [25]. We fine-tuned four transformer-based models using a small dataset of 521 manually annotated phishing emails. Our findings indicate that fine-tuned language models provide a promising approach for extracting contextual information from phishing emails, offering a new direction for security researchers. This method enables us to go beyond traditional text representation, which is not always explainable, towards more nuanced representations that can be highly valuable for certain phishing mitigation tasks. Additionally, we developed a multi-headed classification model to simultaneously perform all four classification tasks. The results indicate that individual models outperformed the combined model when trained with limited data.

In our future work, we have two main goals. First, we aim to investigate other language models capable of handling these tasks, including large language models (LLMs) such as GPT or LLama. Additionally, while our initial experiments in multi-headed classification show promise, they require further work. We intend to annotate more data from multiple sources to enhance our training and evaluation. Second, we plan to explore the various practical applications of PhishCoder, particularly its potential to detect phishing campaigns and develop phishing auto-response tools.

Limitations. One major limitation of our proposed approach is the issue of class imbalance. Class imbalance is a common challenge in machine learning and natural language processing tasks, where certain classes are significantly underrepresented compared to others. In the context of text classification tasks such as action-generic and from-sector, an imbalanced dataset means the model may not learn the nuanced characteristics of the minority classes, leading to lower precision, recall, and F1 scores for those classes. Addressing class imbalance often requires techniques like data augmentation, resampling (oversampling the minority class or undersampling the majority class), and adjusting class weights during training. Another limitation of this study is the lack of access to publicly available datasets. The Nazario Phishing dataset is a collection of phishing emails received by a single person, and may not necessarily be representative or generalizable, which can cause overfitting. In future work, we plan to combine multiple phishing email datasets to create a more representative dataset to conduct experiments.

References

1. Abu-Nimeh, S., Nappa, D., Wang, X., Nair, S.: A comparison of machine learning techniques for phishing detection. In: Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit. pp. 60–69 (2007)
2. Alhogail, A., Alsabih, A.: Applying machine learning and natural language processing to detect phishing email. Computers & Security **110**, 102414 (2021)
3. Althobaiti, K., Vaniea, K., Wolters, M.K., Alsufyani, N.: Using clustering algorithms to automatically identify phishing campaigns. IEEE Access (2023)

4. (APWG), A.P.W.G.: Phishing activity trends report, 4th quarter 2023 (2024), https://apwg.org/trendsreports/, accessed on 4th March 2024
5. Bengio, Y., et al.: A neural probabilistic language model. Journal of Machine Learning Research **3**, 1137–1155 (2003)
6. Bountakas, P., Koutroumpouchos, K., Xenakis, C.: A comparison of natural language processing and machine learning methods for phishing email detection. In: Proceedings of the 16th International Conference on Availability, Reliability and Security. pp. 1–12 (2021)
7. Cowie, J., Lehnert, W.: Information extraction. Communications of the ACM **39**(1), 80–91 (1996)
8. Dagdelen, J., Dunn, A., Lee, S., Walker, N., Rosen, A.S., Ceder, G., Persson, K.A., Jain, A.: Structured information extraction from scientific text with large language models. Nature Communications **15**(1), 1418 (2024)
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
10. Hamid, I.R.A., Abawajy, J.H.: An approach for profiling phishing activities. Computers & Security **45**, 27–41 (2014)
11. Han, Y., Shen, Y.: Accurate spear phishing campaign attribution and early detection. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing. pp. 2079–2086 (2016)
12. IBM: Cost of a data breach report 2023 (2023), https://www.ibm.com/reports/data-breach, accessed on 4th March 2023
13. Jenkins, A., Kokciyan, N., Vaniea, K.E.: Phished: Automated contextual feedback for reported phishing. In: 18th Symposium on Usable Privacy and Security. Usenix (2022)
14. Karim, A., Azam, S., Shanmugam, B., Kannoorpatti, K., Alazab, M.: A comprehensive survey for intelligent spam email detection. IEEE Access **7**, 168261–168295 (2019)
15. Khonji, M., Iraqi, Y., Jones, A.: Phishing detection: a literature survey. IEEE Communications Surveys & Tutorials **15**(4), 2091–2121 (2013)
16. Laclavík, M., Dlugolinskỳ, Š., Šeleng, M., Kvassay, M., Gatial, E., Balogh, Z., Hluchỳ, L.: Email analysis and information extraction for enterprise benefit. Computing and informatics **30**(1), 57–87 (2011)
17. Lain, D., Kostiainen, K., Čapkun, S.: Phishing in organizations: Findings from a large-scale and long-term study. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 842–859. IEEE (2022)
18. Listık, V., Let, Š., Šedivỳ, J., Hlavác, V.: Phishing email detection based on named entity recognition. In: Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP). pp. 252–256. SCITEPRESS – Science and Technology Publications, Lda (2019). https://doi.org/10.5220/0007314202520256
19. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
20. Mahlawi, A.Q., Sasi, S.: Structured data extraction from emails. In: 2017 International Conference on Networks & Advances in Computational Technologies (NetACT). pp. 323–328. IEEE (2017)
21. Nazario, J.: Phishing corpus. Online: http://monkey. org/% 7Ejose/wiki/doku. php (2007)

22. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding with unsupervised learning. Tech. rep., OpenAI (2018)
23. Rajpurkar, P., Zhang, J., Liang, P.: Know what you don't know: Unanswerable questions for squad. In: ACL 2018 (2018)
24. Sage, C., Douzon, T., Aussem, A., Eglin, V., Elghazel, H., Duffner, S., Garcia, C., Espinas, J.: Data-efficient information extraction from documents with pre-trained language models. In: Document Analysis and Recognition–ICDAR 2021 Workshops: Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part II 16. pp. 455–469. Springer (2021)
25. Saka, T., Jain, R., Vaniea, K., Kökciyan, N.: Phishing codebook: A structured framework for the characterization of phishing emails. arXiv preprint arXiv:2408.08967 (2024), https://arxiv.org/abs/2408.08967
26. Saka, T., Vaniea, K., Kökciyan, N.: Context-based clustering to mitigate phishing attacks. In: Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security. pp. 115–126 (2022)
27. Salloum, S., Gaber, T., Vadera, S., Shaalan, K.: Phishing email detection using natural language processing techniques: a literature survey. Procedia Computer Science **189**, 19–28 (2021)
28. Salloum, S., Gaber, T., Vadera, S., Shaalan, K.: A systematic literature review on phishing email detection using natural language processing techniques. IEEE Access **10**, 65703–65727 (2022)
29. Somesha, M., Pais, A.R.: Classification of phishing email using word embedding and machine learning techniques. Journal of Cyber Security and Mobility pp. 279–320 (2022)
30. Song, J., Eto, M., Kim, H.C., Inoue, D., Nakao, K.: A heuristic-based feature selection method for clustering spam emails. In: Neural Information Processing. Theory and Algorithms: 17th International Conference, ICONIP 2010, Sydney, Australia, November 22-25, 2010, Proceedings, Part I 17. pp. 290–297. Springer (2010)
31. Toolan, F., Carthy, J.: Feature selection for spam and phishing detection. In: 2010 eCrime Researchers Summit. pp. 1–12. IEEE (2010)
32. Vaswani, A., et al.: Attention is all you need. In: Advances in neural information processing systems. vol. 30, pp. 5998–6008 (2017)
33. Verma, R., Hossain, N.: Semantic feature selection for text with application to phishing email detection. In: international conference on information security and cryptology. pp. 455–468. Springer (2013)
34. Verma, R., Shashidhar, N., Hossain, N.: Detecting phishing emails the natural language way. In: Computer Security–ESORICS 2012: 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings 17. pp. 824–841. Springer (2012)
35. Williams, H., Leggett, O., Coleman, N., Navin Shah, J., Furnell, S.: Cyber security breaches survey 2021 – chapter 5: Incidence and impact of breaches or attacks (March 2021)
36. Yearwood, J., Webb, D., Ma, L., Vamplew, P., Ofoghi, B., Kelarev, A.: Applying clustering and ensemble clustering approaches to phishing profiling. In: Proceedings of the Eighth Australasian Data Mining Conference-Volume 101. pp. 25–34. Citeseer (2009)
37. Zeng, V., Baki, S., Aassal, A.E., Verma, R., De Moraes, L.F.T., Das, A.: Diverse datasets and a customizable benchmarking framework for phishing. In: Proceedings of the Sixth International Workshop on Security and Privacy Analytics. pp. 35–41 (2020)

Appendix: Phishing Codebook

This codebook is a framework to extract important information from phishing emails in a structured format. While coding only consider the user-facing part of the email; subject line, displayed header, and the main body visible to end-users.

From – Company Name If there is a named company in the from address, subject line, or email body that the email claims to be from. State the name. For example, "PayPal". If two companies are named, include both. Only code for companies or organizations, ignore the reference to a named person. If there are two variants of the same company, name both with the more recognizable name first and separated by commas. For example, 'Microsoft, Outlook'. If there is no name specified in the email, then use one of the following:

- Monkey – Email says it is from some group associated with monkey.org.
- Organization – The email is claiming or implying to be from an organization the user works for. They may use generic terms like "HR" or "Manager" that are associated with a company. Often the organization is not named, but word usage implies that the email is internal.
- None - The email makes no claim about who it is from, or the coder cannot infer a specific organization name from it.

From – Sector The type of sector that the email claims to be from. Refer to the list below to code this. If a company is involved in several different types of sectors, code for the one that the email content refers to. For example, Amazon is both a Shopping website and a Logistics service, if the email is about package delivery, then Logistics should be coded and not Shopping.

- Financial – banks, credit cards, investment company, cryptocurrency
- Email – email provider,or a department that manages email accounts.
- Document share – Online service that allows users to share documents with each other. For example: DocuSign or OneDrive.
- Logistics – shipping and delivery of goods and parcels
- Shopping – purchasing goods or services online, companies like Amazon.
- Service provider – organization that provides online services and does not fall into any sector described above.
- Security – any company that claims to be a security provider, anti-virus service, or regarding identity protection due to a security breach.
- Government – any email from a government organization. For example, tax-related emails, HMRC, or VISA-related emails, and so on.
- Unknown – the coder can't tell from just the email or does not know what the company is.

Salutation The type of salutation used to address the user. Some phishers tailor emails to users by using their name or email to make it seem legitimate, while others send out mass emails with no specific salutation. Code if the email addresses the user specifically, either in a proper salutation or in the email body text. IGNORE the email header for this code.

- Name - if the name of the user is used. For example: 'Dear Jose', 'Attention Jose'
- Email - if user email id is used in the salutation
- Generic - if no name or personal salutation is used. For example: 'Dear User'
- None - if there is no salutation at all or a reference to an individual. For example: 'Hello'

Threatening language Phishers often use a threat or warning to scare users into taking action. Code if the email subject line or message contains any threatening language or tone. This includes talking about negative consequences or loss.

- Threat - If there is a DIRECT threat statement or wording in the email. For example: "account will be deleted" or "money will be lost".
- None- If the email has no threat or incentive mentioned. Or if you cannot accurately infer one from the email. Also includes cases where the outcome of a process is described (I.e. mail being held) which may be undesirable, but no direct threat statement exists.

Urgency Cues Phishers often try to create a sense of urgency to encourage, or even demand, immediate action in a bid to fluster the receiver. Code if the email message or subject line contains time pressure or urgency cues, including implied.

- Urgent – if there is a mention of time limit or mention any urgency words. For example: 'files will be lost in 24hrs' or 'expire in 3 days' or 'click Immediately' or 'soon'.
- None – If there is no such time pressure scare or wording used in the email. Or if you cannot accurately infer from the message.

Action – Generic Code the action being prompted in the email. This only includes clearly stated explicit actions, and not any implied actions.

- Click – if the email is asking you to click on any links
- Download – if the email is asking you to download an attachment or application
- Reply/Email – if the email is asking you to reply or send an email to a given address
- Call – if the email is asking you to call a number
- Other – If any other action is mentioned that is not covered above.
- None – No clear action is requested, or the coder is unsure what is being asked for.

Action – Specific Provide details regarding the reason the phisher has given for the action. We will use in vivo coding for this column - copy and paste a few words/phrases directly from the email text. For example: For the action 'click link to verify account', the code is 'to verify account'.

Main Topic Code the main purpose of the email. Use a few words or a phrase from the email to summarize the main topic. For example: 'package has been returned'.