# Prompt Engineering for Detecting Phishing

Darrell L. Young[a], Eric C. Larson[a], and Mitchell A. Thornton[a]

[a]Darwin Deason Institute of Cybersecurity, Southern Methodist University, Dallas, TX 75205, USA

## ABSTRACT

"This paper introduces an adversarial framework using two Large Language Models (LLMs) tailored through prompt engineering to advance phishing detection capabilities. The first LLM operates as a generator, crafted prompts guiding it to produce sophisticated phishing emails that mimic legitimate interactions. The second LLM acts as a discriminator, with prompts designed to enhance its ability to detect and classify these generated phishing attempts accurately from genuine emails. This setup leverages the dynamic capabilities of prompt engineering to refine the models' responses, facilitating an ongoing evolution in both phishing simulation and detection. By implementing this methodology, we aim to improve the robustness and adaptability of AI-driven security defenses against complex cyber threats."

**Keywords:** Phishing detection, adversarial framework, large language models, prompt engineering, incremental learning, user education, cybersecurity

## 1. INTRODUCTION

Phishing attacks have become a pervasive and rapidly escalating threat, compromising individuals, organizations, and governments worldwide. According to the *APWG Phishing Activity Trends Report for Q3 2024*,[1] there were over 932,923 reported phishing attacks in one quarter alone. This increase is compounded by the misuse of **Large Language Models (LLMs)** that enable attackers to generate highly convincing, personalized phishing emails at scale.[2] There has been a significant rise in the sophistication of phishing attacks due to the use of Large Language Models (LLMs) by cyber attackers.[3] Such capabilities outpace traditional defenses and necessitate new adaptive detection methodologies.

The evolution of phishing has reached a critical tipping point. High-profile incidents demonstrate the catastrophic consequences of phishing attacks. Modern attackers employ LLMs to generate emails that appear authentic, making it challenging for the reactive network techniques shown in Table 1 to keep pace. AI-enabled zero-day attacks can filter through network defenses. The last defense is a combination of a text-based classifier and an educated user with the discernment to recognize possible phishing attacks and other forms of online deception.

Further author information: (Send correspondence to Darrell L. Young)
Darrell L. Young: E-mail: dlyoung@smu.edu

### 1.1 LLM-Enabled Outcome Engineering for Enhanced User Education

A long-term goal of this research is to improve user outcomes by enhancing their ability to recognize and appropriately respond to phishing attacks. We propose the concept of *Outcome Engineering* as a strategic framework for using AI to achieve tangible, measurable improvements in organizational and personal decision-making.[4] In our context, the desired outcome is a better-educated user—one who is equipped to discern phishing attempts and other forms of online deception with greater accuracy and confidence.

Table 1. Security Approaches and Descriptions

| Security Approach | Description |
|---|---|
| **DNS-Based Defenses:** | Using advanced DNS-layer security to block malicious sites before a connection is established. |
| **Secure Access Service Edge (SASE):** | Securing network access for users regardless of location. |
| **Endpoint Protection:** | Employing machine learning and behavioral analysis to detect threats at the endpoint level. |
| **Email Security:** | Focusing on protecting against email-based threats with advanced threat detection and threat intelligence. |
| **Zero Trust Security:** | Enforcing continuous authentication and authorization to validate user access. |
| **Extended Detection and Response (XDR):** | Integrating data from multiple security products for coordinated threat response. |

Phishing attacks continue to evolve in sophistication, increasingly exploiting the capabilities of large language models (LLMs) to generate convincing, personalized fraudulent communications. Traditional phishing detection methods rely primarily on static URL indicator analysis and retrained models, which can quickly become outdated as attackers adapt their tactics.

Our framework introduces a dual-LLM approach: one LLM (the **Generator**) produces phishing and legitimate using the phishing URL dataset, while the other LLM (the **Detector**) classifies emails and, crucially, provides detailed explanations. This educates users by highlighting the red flags in suspicious emails. In this study, we further enhance the efficacy of our approach by comparing the performance of the **Detector** LLM with and without the application of DSPy's MIPROv2 optimizer,[5] which programmatically refines the detection prompts to improve both accuracy and the quality of explanations provided.

### 1.2 Motivation and Need for New Approaches

The limitations of conventional phishing detectors include:

- **Lack of Interpretability:** Traditional classifiers output a binary decision without explaining the rationale, leaving users uncertain about why an email was flagged.
- **Static Nature:** Once trained, models do not easily adapt to new phishing tactics without costly retraining.

- **User Education:** Without clear explanations, even high-accuracy models fail to educate users, thereby limiting their ability to recognize future threats.

In contrast, our approach leverages prompt engineering to empower an LLM-based detector to generate explanations with confidence scores. These explanations not only validate the detection decision but also provide actionable insights that can be used to educate users on phishing red flags.

## 2. SYSTEM ARCHITECTURE

Our system combines an incremental URL indicator classifier with an LLM-based phishing detector, operating block-by-block where the optimized detector from the previous block is applied to the current block. Figure 1 illustrates the high-level architecture.

### 2.1 Incremental Learning

The incremental classifier described in[6] continuously ingests URL data, extracting features such as URL length, domain characteristics, TLD legitimacy, and other heuristics. It uses the strengths of multiple models to make a classification. We emulate this approach by combining the relative advantages of the BernoulliNB and PassiveAgressiveClassifier from the Python sklearn library.[7]

Our emulated algorithm quickly adapts to changing threat landscapes, achieving high accuracy on incoming data. However, it lacks the capability to explain its decisions in a human-understandable way.

### 2.2 LLM-Based Generation and Detection with Explanation

The URL dataset described in[?] does not include email subject line and body. We use the LLM generator uses the input URL and label, phishing or legitimate, from the dataset to produce a a synthetic subject line and email body. The augmented dataset is then sent to the LLM-based optimized and non-optimized detector. Both detectors are required to provide reasoning outputs.

### 2.3 DSPy Optimization

DSPy, or Declarative Self-improving Python, revolutionizes programming language models by facilitating rapid iterations in building modular AI systems with enhanced features for prompt and weight optimization. Unlike traditional methods that focus on prompt strings, DSPy allows developers to employ structured, declarative code modules, enabling more reliable and maintainable AI systems. This framework supports everything from simple classifiers to complex recursive agent loops.

DSPy provides tools that convert high-level code into low-level computations, effectively aligning language models with a program's structure and metrics. Its optimization tools, such as dspy.BootstrapRS, dspy.MIPROv2, and dspy.BootstrapFinetune, enhance prompt effectiveness and finetune weights based on quality metrics.[8]

The DSPy community, fostered by over 250 contributors, supports an open-source ecosystem that enhances the compositional architectures and inference strategies for language model programs.[9] Initiated by Stanford NLP in 2022, DSPy has grown rapidly, contributing significantly to AI research and practical applications, thereby increasing control over AI systems and improving outcomes in various real-world applications.

## 3. EXPERIMENTS AND RESULTS

In this section, we detail preparation and analysis of the *PhiUSIIL_Phishing_URL_Dataset.csv*, focusing on achieving a balanced representation of data through stratification. This foundational step ensures that the subsequent training of our models is balanced on a block basis. We then explore the performance of an incremental machine learning approach on just the URL and associated features compared to two language models—the **Generator** and the **Detector**. The Generator is tasked with simulating realistic email interactions, while the Detector evaluates these interactions for signs of phishing, using both standard and enhanced prompts. Our results shed light on the nuanced capabilities of these models to differentiate between benign and malicious intents, underscoring the potential of AI LLM technology to bolster cybersecurity defenses and educate users.

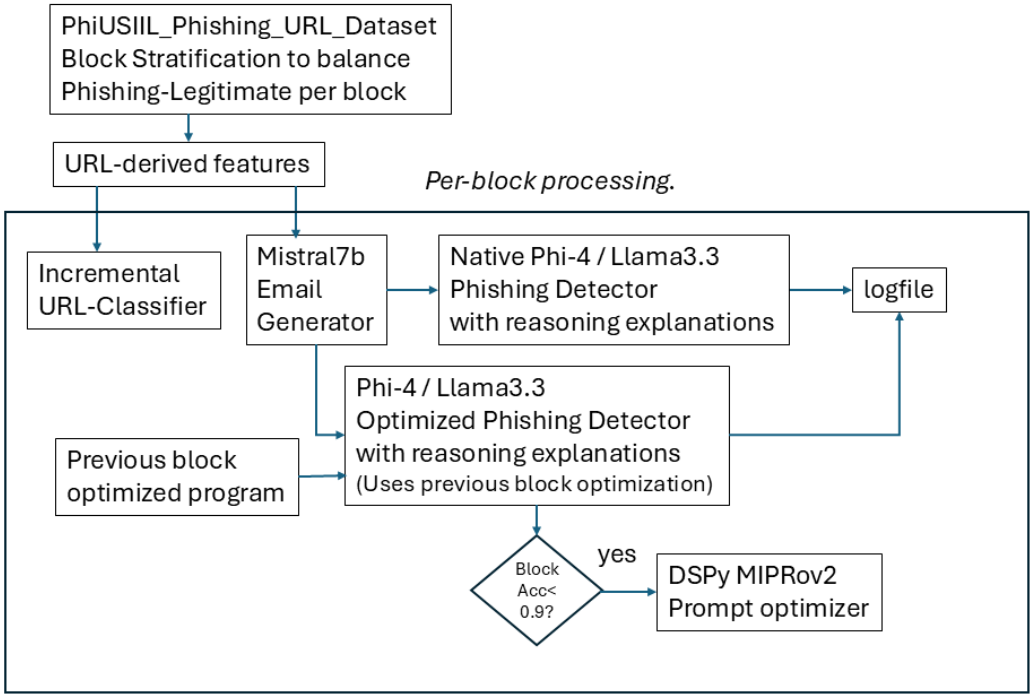


Figure 1. Overview of the adversarial phishing detection architecture. The Generator LLM produces realistic phishing emails incorporating URL indicators, while the Detector LLM evaluates these emails, providing classifications with confidence scores and detailed explanations.

### Stratification Procedure

We performed preprocessing on the dataset titled *PhiUSIIL_Phishing_URL_Dataset.csv* to ensure an even distribution of labels across the data blocks. This preprocessing was critical for maintaining

consistency in the training and evaluation of our machine learning models. The dataset consists of the following variables:

To balance the dataset, we utilized the `StratifiedKFold` technique from the `sklearn.model_selection` module, configured as follows:

- **Number of Splits**: The dataset was divided into blocks of 20 entries each, calculated based on the total number of samples divided by the block size.
- **Shuffling**: Enabled to ensure random distribution of data points within each block.
- **Random Seed**: Set to 42 to allow reproducibility of the dataset partitioning.

This method ensures each block contains a proportional number of samples from each class, mirroring the overall distribution of labels in the dataset. The blocks were then concatenated into a single DataFrame and saved as *PhiUSIIL_Phishing_URL_Dataset_balanced.csv*, preserving the order and integrity of the data for subsequent analyses.

To illustrate the effect of stratification on label distribution, we analyzed the first ten blocks of the dataset, both before and after applying stratification. The stratification process aimed to ensure a more balanced distribution of labels (Phishing and Legitimate) across each block. We calculated the difference in the proportion of phishing labels between the original and balanced datasets for each block. The blocks with the most significant changes were selected for visualization to demonstrate the impact of this balancing technique effectively.

Figure 2 shows two selected blocks where the difference in label distribution before and after stratification was most pronounced. These examples highlight how stratification helps achieve a more uniform distribution of labels across different subsets of the dataset, which is crucial for maintaining the consistency and reliability of model training and evaluation processes.

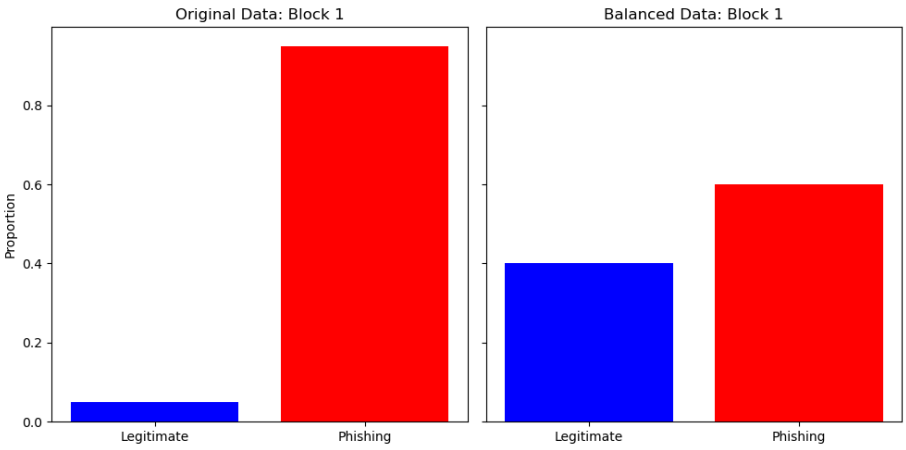


Figure 2. Comparison of label distribution in selected blocks before and after applying stratification. The blocks were chosen based on the most significant changes in the proportion of phishing labels, showcasing the effectiveness of stratification in balancing the dataset.

The `extract_url_features` function computes several features from URLs to be used by the classifier:

- **URLLength**: The total number of characters in the URL. This feature helps in differentiating between typically longer, complex phishing URLs and shorter, legitimate ones.
- **IsDomainIP**: A binary indicator (0 or 1) that checks if the domain part of the URL is an IP address, a common characteristic in phishing attacks to avoid traceable domain name registrations.
- **NoOfSubDomain**: This feature counts the number of subdomains in the URL minus two, capturing the complexity or attempts to mimic legitimate websites by adding additional layers of subdomains.
- **IsHTTPS**: A binary feature indicating whether the URL's scheme is HTTPS. Although HTTPS indicates encryption, many phishing sites also use HTTPS to appear secure.
- **HasObfuscation**: Checks for common URL obfuscation techniques like the presence of '%' (indicating URL encoding) or '@' (used in phishing to confuse users about the actual domain being accessed).

These features are transformed into a numpy array, which is then ready to be standardized and used in predictions, ensuring that the classifier's training and prediction processes are based purely on observable characteristics of the URL without any leakage from the labels.

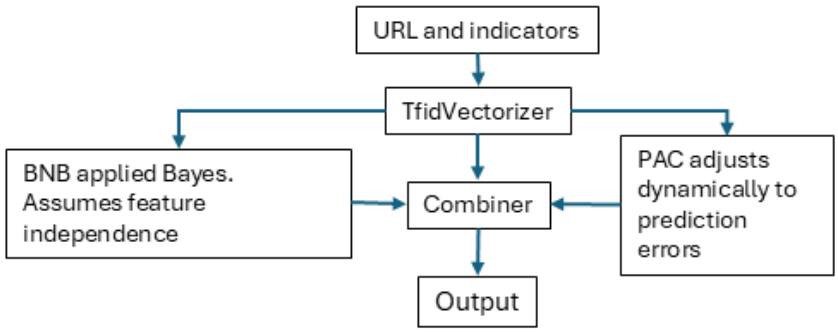


Figure 3. URL data and indicators transformed by TfidfVectorizer, then processed through both the Bernoulli Naive Bayes (BNB) and Passive Aggressive Classifier (PAC). BNB applies Bayesian logic assuming feature independence, while PAC dynamically adjusts to prediction errors. The outputs from both classifiers are then combined to produce a robust final output that balances immediate adaptability with long-term trend analysis.

To create a reference baseline incremental machine learning phishing detection, we follow an approach similar to what is described by Prasad et al.[6] which uses the Bernoulli Naive Bayes (BNB) and Passive Aggressive (PAC) classifiers together, along with the TfidfVectorizer from the Scikit-learn library,[7] to implement an incremental learning system for URL classification (see Figure 3). The TfidfVectorizer transforms URL indicators into a matrix of TF-IDF features, quantifying the importance of terms within the dataset, providing a numerical representation for analysis. BNB, suited for binary features, applies Bayes' theorem under the assumption of feature independence, effectively handling data where feature presence strongly indicates class membership. In contrast,

PAC adjusts its model dynamically in response to prediction errors, making it ideal for environments with evolving data like phishing detection. Both classifiers are trained incrementally, allowing continual adaptation without full retraining. The script combines the predictive outputs of both classifiers by averaging their individual predictions, thereby enhancing decision accuracy and stability. This method leverages the structured input from TfidfVectorizer and ensures that the final classification decision is robust, taking into account the strengths of both classifiers to maintain optimal phishing detection capabilities as data evolves. The Passive Aggressive Classifier excels in swiftly adjusting to short-term changes, while the Bernoulli Naive Bayes optimizes performance against overall long-term trends in the data.

The results indicate that the incremental ML URL-based classifier exhibits strong performance, consistent with findings reported by Prasad et al.,[6] suggesting its effectiveness in real-world applications.

Table 2. Incremental ML Classifier Performance

| Block | Accuracy (%) | Precision (%) | Recall (%) | F1 Score (%) |
|---|---|---|---|---|
| 1 | 96.67 | 93.75 | 100.0 | 96.77 |
| 2 | 100.0 | 100.0 | 100.0 | 100.0 |
| 3 | 96.67 | 94.44 | 100.0 | 97.14 |
| 4 | 100.0 | 100.0 | 100.0 | 100.0 |
| 5 | 100.0 | 100.0 | 100.0 | 100.0 |
| 6 | 100.0 | 100.0 | 100.0 | 100.0 |
| 7 | 100.0 | 100.0 | 100.0 | 100.0 |
| 8 | 100.0 | 100.0 | 100.0 | 100.0 |
| 9 | 100.0 | 100.0 | 100.0 | 100.0 |
| 10 | 100.0 | 100.0 | 100.0 | 100.0 |

### 3.1 Mistral-7b Subject Line and Email Generator

Mistral 7B is the name of the LLM model used to generate the emails using the URL dataset. Mistral 7B is a versatile 7.3 billion parameter language model that excels in performance and efficiency.[10] Mistral AI, a pioneering French artificial intelligence startup founded in April 2023.

The generation of email subject lines and bodies is a crucial step in simulating realistic email communications for our experiments. As detailed in Figure 1, the process utilizes the Mistral 7b model to create content based on the nature of the email, whether it is legitimate or phishing.

For legitimate emails, the prompt instructs the model to generate a professional and coherent subject and body without any indicators of suspicious email characteristics. This is intended to mimic genuine communication from a reputable source. Conversely, for phishing emails, the prompt strategically encourages the model to include elements designed to manipulate the recipient, using tactics that invoke fear, greed, or urgency, among others. The goal is to craft an email that compels the recipient to take an action against their best interest, such as clicking on a malicious link or opening an attachment, while still maintaining a veneer of legitimacy.

The prompt explicitly requires responses in JSON format with keys 'subject' and 'body', ensuring structured output that can be easily parsed and utilized in further processing or analysis within

our system. This approach allows us to systematically evaluate the effectiveness of different detection techniques against varied and nuanced phishing tactics generated by the advanced language capabilities of Mistral 7b.

### 3.2 Detector Manual Prompt Construction

Below is the non-optimized prompt to detect phishing emails.

```
# manually constructed non-optimized prompt
prompt = (
    "You are an expert email security analyst. Analyze the following email and
    classify it as either 'phishing' or 'legitimate'. Use your internal
    knowledge of phishing tactics to infer hidden reasons. Consider the following
    URL and its features as part of the context for your analysis: \n" +
    glossary + " \nProvide a detailed explanation in natural language. Respond
    ONLY with a JSON object that includes exactly three keys: 'label' (phishing
    or legitimate), 'confidence' (a float between 0 and 1), and 'reasoning' (an
    array of strings). Do not include any extra text. Subject: " +
    email.get('subject') + "\nBody: " + email.get('body')
)
```

Figure 4. Listing of the prompt used for the base method in the email classification process.

In our method, we leverage a detailed prompt to guide the large language model in classifying emails accurately. The prompt, detailed in Figure 4, combines contextual information about the email with a set of well-defined criteria for the analysis. This structured input is designed to maximize the model's understanding and output precision.

### 3.3 Optimized Prompt from Signature

The optimized prompt instructions specifically requests the following attributes to be considered in the analysis: *email subject*, *email body*, *URL* (URL), *URL length*, *domain IP status*, *number of subdomains*, *HTTPS status*, and *presence of obfuscation*. Based on these inputs, the model is expected to produce a classification label (either 'phishing' or 'legitimate') and a detailed reasoning behind this classification.

The table depicted in Table 3 shows examples of MIPROv2[5] optimizer prompt instructions from the Llamma3.3 phishing detection experiments. Each entry in the table corresponds to a specific instructional scenario presented to the model. These scenarios delineate various aspects to be considered by the model, such as analyzing email characteristics, determining the legitimacy of the content, and recognizing phishing tactics. This explicit directive ensures that the model's responses are aligned with the cybersecurity objectives, providing a structured and detailed methodology for its reasoning process. The prompts are designed to reflect the complexities and subtleties involved in distinguishing between legitimate correspondence and phishing attempts, highlighting the model's capacity to handle nuanced language tasks.

| Example of Prompt Optimizer Generated Instructions |
|---|
| Classify an email as "phishing" or "legitimate" based on its subject, body, machine learning context, URL characteristics (length, domain type, number of subdomains, HTTPS usage), and presence of obfuscation techniques. Provide a detailed reasoning for the classification, considering factors such as urgency creation, security concerns, emotional manipulation, suspicious links, and syntax. Ensure the analysis is thorough and informative to effectively determine the email's legitimacy. |
| You are a cybersecurity expert tasked with protecting a high-profile government official's email account from phishing attacks. Given the fields `email_subject`, `email_body`, `ml_context`, `url_length`, `is_domain_ip`, `no_of_subdomains`, `is_https`, `has_obfuscation`, you must produce the fields `label` and `reasoning` to classify an incoming email as either "phishing" or "legitimate". The official's email account has been targeted by sophisticated phishing attempts in the past, and it is crucial that you accurately identify any potential threats to prevent a security breach. The `label` should be either "phishing" or "legitimate", and the `reasoning` should provide a clear explanation for your classification decision, including any relevant factors such as suspicious keywords, unusual sender behavior, or malicious links. Your prompt and accurate response is critical to ensuring the official's email account remains secure. |
| Classify an email as either "phishing" or "legitimate" based on its content and technical characteristics. To accomplish this task, carefully analyze the provided input fields: `email_subject`, `email_body`, `ml_context`, `url_length`, `is_domain_ip`, `no_of_subdomains`, `is_https`, and `has_obfuscation`. Use these inputs to identify potential indicators of phishing attempts, such as suspicious links, urgent language, or obfuscated content. Then, generate a detailed list of reasons (`reasoning`) supporting your classification decision and assign a corresponding label (`label`) indicating whether the email is "phishing" or "legitimate". Ensure that your reasoning is thorough and takes into account various factors, including the email's syntax, semantics, and any relevant technical indicators. |
| You are a cybersecurity expert tasked with classifying emails as either phishing or legitimate in a high-stakes scenario where a major corporation's email system has been compromised. The CEO's personal email account has been hacked, and it's up to you to identify which emails are malicious and which are genuine. Given the fields `email_subject`, `email_body`, `ml_context`, `url_length`, `is_domain_ip`, `no_of_subdomains`, `is_https`, `has_obfuscation`, produce the fields `label` (either 'phishing' or 'legitimate') and `reasoning` (a list of reasons why you classified the email as such). Your classification will directly impact the corporation's security measures, so it's crucial that you get it right. Use your expertise to analyze the input fields carefully and provide a detailed explanation for your classification. |

Table 3. Extracted instructions from the log file.

### 3.4 Phi-4 Detector Performance

Microsoft's Phi-4 is a 14-billion parameter language model that emphasizes data quality over volume. Unlike traditional models that use organic data, Phi-4 integrates synthetic data throughout its training. It builds on the Phi series and GPT-4's capabilities, especially in STEM-focused question-answering. Despite minimal changes from its predecessor Phi-3, Phi-4 excels in reasoning benchmarks due to its refined training curriculum and innovative post-training techniques.[11]

Table 4. Phi-4 LLM Detector Performance with Mistral7B v2 as email subject and body Generator

| Block Index | Base Performance | | | | Optimized Performance | | | | Improvement |
|---|---|---|---|---|---|---|---|---|---|
| | Acc | Prec | Recall | F1 | Acc | Prec | Recall | F1 | F1 |
| 1 | 50.0% | 50.0% | 100.0% | 66.7% | N/A | N/A | N/A | N/A | N/A |
| 2 | 60.0% | 60.0% | 100.0% | 75.0% | 100.0% | 100.0% | 100.0% | 100.0% | +25.0% |
| 3 | 56.7% | 56.7% | 100.0% | 72.3% | 66.7% | 64.0% | 94.1% | 76.2% | +3.9% |
| 4 | 53.3% | 53.3% | 100.0% | 69.6% | 73.3% | 66.7% | 100.0% | 80.0% | +10.4% |
| 5 | 70.0% | 70.0% | 100.0% | 82.4% | 86.7% | 84.0% | 100.0% | 91.3% | +8.9% |
| 6 | 56.7% | 56.7% | 100.0% | 72.3% | 76.7% | 72.7% | 94.1% | 82.1% | +9.8% |
| 7 | 53.3% | 53.3% | 100.0% | 69.6% | 76.7% | 71.4% | 93.8% | 81.1% | +11.5% |
| 8 | 56.7% | 56.7% | 100.0% | 72.3% | 80.0% | 73.9% | 100.0% | 85.0% | +12.7% |
| 9 | 56.7% | 56.7% | 100.0% | 72.3% | 76.7% | 75.0% | 88.2% | 81.1% | +8.8% |
| 10 | 53.3% | 53.3% | 100.0% | 69.6% | 80.0% | 66.7% | 100.0% | 80.0% | +10.4% |

### 3.5 Llama3.3 Detector Performance

Llama 3.3 revolutionizes language model performance by delivering output quality comparable to the 405-billion-parameter Llama 3.1, but with only 70 billion parameters, enhancing efficiency and broadening functionality. It supports eight languages—English, French, Italian, Portuguese, Hindi, Spanish, Thai, and German—and integrates seamlessly with third-party tools for complex function calling tasks. This model excels in multilingual customer support and automated analysis, matching or surpassing competitors like Gemini Pro 1.5 and GPT-4 in instruction following, reasoning, and handling of long contexts, despite slightly lagging in math capabilities.[12]

Table 5. Llama3.3 Detector Performance with Mistral7B v2 as email subject and body Generator

| Block Index | Base Performance | | | | Optimized Performance | | | | Improvement |
|---|---|---|---|---|---|---|---|---|---|
| | Acc | Prec | Recall | F1 | Acc | Prec | Recall | F1 | F1 |
| 1 | 56.7% | 53.6% | 100.0% | 69.8% | N/A | N/A | N/A | N/A | N/A |
| 2 | 73.3% | 69.2% | 100.0% | 81.8% | 80.0% | 83.3% | 83.3% | 83.3% | +1.5% |
| 3 | 70.0% | 65.4% | 100.0% | 79.1% | 83.3% | 80.0% | 94.1% | 86.5% | +7.4% |
| 4 | 63.3% | 59.3% | 100.0% | 74.4% | 66.7% | 65.0% | 81.3% | 72.2% | -2.2% |
| 5 | 80.0% | 77.8% | 100.0% | 87.5% | 73.3% | 84.2% | 76.2% | 80.0% | -7.5% |
| 6 | 63.3% | 60.7% | 100.0% | 75.6% | 93.3% | 94.1% | 94.1% | 94.1% | +18.5% |
| 7 | 70.0% | 64.0% | 100.0% | 78.0% | 76.7% | 76.5% | 81.3% | 78.8% | +0.8% |
| 8 | 60.0% | 59.3% | 94.1% | 72.7% | 80.0% | 82.4% | 82.4% | 82.4% | +9.7% |
| 9 | 73.3% | 68.0% | 100.0% | 80.9% | 76.7% | 77.8% | 82.4% | 80.0% | -0.9% |
| 10 | 73.3% | 68.0% | 100.0% | 80.9% | 80.0% | 77.8% | 82.4% | 81.1% | +0.2% |

The tables now properly display the lack of optimized metrics for the first block, ensuring clarity in understanding that there were no optimization comparisons for that block.

Whenever the accuracy of the optimized LLM-based detector fell below the threshold of 0.9, a DSPy MIPROv2 optimization cycle was triggered. This optimization process utilized labeled examples from the block to refine and enhance the model's accuracy in detecting phishing emails. This iterative process aims to improve the detection capabilities of the system by adjusting and fine-tuning the LLM's response to the sophisticated tactics employed in the generated emails

The performance of the LLM-based detector is relatively poor. This may be partly due to the sophisticated nature of the deceptive emails generated by the Mistral 7B model, which challenges the detector's ability to discern between legitimate and phishing emails. However, it is clear that prompt tuning increased the performance for both Phi-4 and Llama3.3 detectors.

### 3.6 Illustrative Examples

In the experiment documented, a phishing detection scenario was staged using the URL `https://www.sunyocc.edu` as input, which generated an email with the subject "Suspicious Account Activity Alert - Sunyocc.edu" and a detailed body asserting suspicious activities in a student's account, urging immediate action as shown in Table 6.

Both the non-optimized Table 7 and optimized Table 8 provided several reasoning points, such as the creation of urgency, the use of emotional appeals, and discrepancies between the displayed and actual links. These reasoning points are useful for user education on phishing identification

Overall, while both the non-optimized and optimized prompts offer useful insights for user education on phishing, the optimized detector's higher overall accuracy shows the value of prompt programming.

Table 6. Email Subject and Body Content

| **Feature** | **Description** |
|---|---|
| Email Subject | Suspicious Account Activity Alert - Sunyocc.edu |
| Email Body | Attention Student!<br><br>We have detected an unusual activity in your Sunyocc.edu account and we believe it may be compromised. To secure your account immediately, click on the provided link below to verify your identity and regain access: [Verify Now] (suspicious_link)<br><br>Failure to do so within 24 hours will result in permanent suspension of your account and loss of all data.<br><br>Don't miss this opportunity to protect your valuable academic records! Act now before it's too late!<br><br>Sincerely,<br>Sunyocc.edu Security Team |

## 4. DISCUSSION AND FUTURE WORK

There are many avenues to explore for future work: 1. The LLM optimization could be modified to be more like incremental learning approach to take advantage of previously learned prompts. The DSPy optimizers do offer warm-start option however it only intializes the optimization cycle with a previously optimized program; 2. Prompt optimization "works better together"[13] with model fine tuning; 3) Adversarial feedback from the detector to the generator could lead to an arms race to make both better; 4) Use the incremental machine learning classification results to confidently guide when to expose the LLM phishing explanations to the user.

Table 7. Non-Optimized LLM Classification and Reasoning

| Feature | Description |
|---|---|
| Non-Optimized LLM Classification | Phishing |
| Non-Optimized Reasoning | • The email creates a sense of urgency with a time limit to respond, which is a common phishing tactic.<br>• It uses emotional appeals like the potential loss of academic records to prompt immediate action without careful consideration.<br>• Although the URL provided in the context appears legitimate, the actual link in the email body is described as 'suspicious_link', suggesting it may not match the given URL, which could be a phishing technique to mislead.<br>• The presence of HTTPS in the given URL does not guarantee the legitimacy of the email since many phishing sites now use HTTPS.<br>• The lack of obfuscation and the single subdomain in the provided URL do not necessarily indicate legitimacy when considering the suspicious link in the email body. |

Table 8. Optimized LLM Classification and Reasoning

| Feature | Description |
|---|---|
| Optimized Classification | Phishing |
| Optimized Reasoning | • The optimized detection method confirms the phishing attempt through deeper analysis of the URL features and the urgency created in the email content.<br>• The optimized model recognizes patterns associated with phishing, such as the discrepancy between the displayed URL and the actual link, which is a common tactic in sophisticated phishing schemes. |

## 5. CONCLUSION

The near-perfect performance of the incremental machine learning approach on the just the URL features draws a sharp contrast to poor to middling performance of the non-optimized and optimized LLM models. If the task had been to simply classify potential phishing threats, the machine learning approach offers higher performance at far less resource cost. This shows LLMs are not always the right tool. However, the machine learning approach lacked the ability to explain or educate the user on how to recognize phishing. LLMs can provide the user education which is also necessary to help users recognize increasingly sophisticated phishing attacks. Prompt optimization offers principled method to develop the prompt in a way that can respond to changing nature of phishing attacks.

## Acknowledgments

We thank our colleagues at the Darwin Deason Institute for Cybersecurity[14] for their valuable feedback and support.

## REFERENCES


[1] APWG, "Phishing activity trends report for q3 2024," *APWG Reports* (2024). Accessed: [Insert Date].

[2] Review, H. B., "The threat of ai-driven phishing," *Harvard Business Review* (2024). Accessed: [Insert Date].

[3] Afane, K., Wei, W., Mao, Y., Farooq, J., and Chen, J., "Next-generation phishing: How llm agents empower cyber attackers," in [*2024 IEEE International Conference on Big Data (BigData)*], 2558–2567, IEEE (2024).

[4] D. L. Young, E. C. L. and Thornton, M. A., "Ai-assisted outcome engineering for mapping natural language to radar configuration files," in [*Proceedings of the IEEE Dallas Circuits and Systems Conference (DCAS)*], (2025). Under review.

[5] Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., Vardhamanan, S., Haq, S., Sharma, A., Joshi, T. T., Moazam, H., Miller, H., Zaharia, M., and Potts, C., "Dspy: Compiling declarative language model calls into self-improving pipelines," *The Twelfth International Conference on Learning Representations* (2024).

[6] Prasad, A. and Chandra, S., "Phiusiil: A diverse security profile empowered phishing url detection framework based on similarity index and incremental learning," *Computers & Security* **136**, 103545 (2024).

[7] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É., "Scikit-learn: Machine learning in Python," (2011).

[8] "Dspy: Declarative self-improving python." https://dspy.ai/. Accessed: 2023-12-07.

[9] Group, S. N., "Dspy github repository." https://github.com/stanfordnlp/dspy. Accessed: 2023-12-07.

[10] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E., "Mistral 7b," (2023).

[11] Research, M., "Phi-4: A 14-Billion Parameter Language Model." https://www.microsoft.com/en-us/research/uploads/prod/2024/12/P4TechReport.pdf (2024). Accessed: 2025-01-23.

[12] Llama, "Llama 3.3 model card and prompt formats." https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_3/ (2023). Accessed: 2023-02-24.

[13] Soylu, D., Potts, C., and Khattab, O., "Fine-tuning and prompt optimization: Two great steps that work better together." Stanford University (2024). Available online: https://ai.stanford.edu.

[14] "Darwin Deason Institute for Cybersecurity." https://www.smu.edu/lyle/centers-and-institutes/ddi. Accessed: 2024-02-25.