machine learning & knowledge extraction

MDPI

Article

Prompt Engineering or Fine-Tuning? A Case Study on Phishing Detection with Large Language Models

Fouad Trad * and Ali Chehab

Electrical and Computer Engineering, American University of Beirut, Beirut 1107-2020, Lebanon; chehab@aub.edu.lb
* Correspondence: fat10@mail.aub.edu

Abstract: Large Language Models (LLMs) are reshaping the landscape of Machine Learning (ML) application development. The emergence of versatile LLMs capable of undertaking a wide array of tasks has reduced the necessity for intensive human involvement in training and maintaining ML models. Despite these advancements, a pivotal question emerges: can these generalized models negate the need for task-specific models? This study addresses this question by comparing the effectiveness of LLMs in detecting phishing URLs when utilized with prompt-engineering techniques versus when fine-tuned. Notably, we explore multiple prompt-engineering strategies for phishing URL detection and apply them to two chat models, GPT-3.5-turbo and Claude 2. In this context, the maximum result achieved was an F1-score of 92.74% by using a test set of 1000 samples. Following this, we fine-tune a range of base LLMs, including GPT-2, Bloom, Baby LLaMA, and DistilGPT-2—all primarily developed for text generation—exclusively for phishing URL detection. The fine-tuning approach culminated in a peak performance, achieving an F1-score of 97.29% and an AUC of 99.56% on the same test set, thereby outperforming existing state-of-the-art methods. These results highlight that while LLMs harnessed through prompt engineering can expedite application development processes, achieving a decent performance, they are not as effective as dedicated, task-specific LLMs.

Keywords: large language models; prompt engineering; fine-tuning; phishing detection

Citation: Trad, F.; Chehab, A. Prompt Engineering or Fine-Tuning? A Case Study on Phishing Detection with Large Language Models. *Mach. Learn. Knowl. Extr.* **2024**, *6*, 367–384. https://doi.org/10.3390/make6010018

Academic Editor: Francesco Buccafurri

Received: 21 November 2023
Revised: 10 January 2024
Accepted: 26 January 2024
Published: 6 February 2024

1. Introduction

In recent years, ML application development has witnessed a transformative impact due to the emergence of LLMs. These models, notable for their vast training across diverse datasets, have demonstrated an impressive adaptability and performance across a wide array of tasks [1–4]. This development signifies a shift toward utilizing LLMs through prompting for various applications, challenging the conventional need for intensive, task-specific training and the maintenance of ML models. This paradigm shift has propelled ML toward a new era where the development of specialized models for each task is being questioned since LLMs already perform a multitude of tasks in a decent way [5,6].

The primary motivation behind this research lies in understanding the extent to which LLMs can replace dedicated models for specific tasks. Our investigation delves into the critical domain of cybersecurity, with a specific focus on the detection of phishing URLs—an enduring and sophisticated online threat [7]. In this realm, the conventional approach has primarily revolved around the use of rule-based systems [8,9], ML algorithms like decision trees, support vector machines [10,11], or neural networks trained on specific phishing characteristics [12–14]. These traditional methods often require extensive feature engineering and are limited by the need for constant updates to keep pace with the evolving nature of phishing attacks. We aim to assess whether LLMs, with their broad training and adaptability, can provide a more efficient yet effective alternative in this critical domain.

Specifically, two novel approaches are adopted, the prompt engineering and fine-tuning of LLMs, to assess their efficacy in the context of detecting phishing URLs. Prompt engineering involves crafting specific input prompts to guide the LLM toward desired

outputs without modifying the model itself [15], a new technique that emerged with the rise of LLMs and not previously applied in the phishing context. Fine-tuning, on the other hand, involves relying on a pretrained model and adjusting its parameters on a dataset specific to the task at hand [16], a method also novel in the phishing domain. This dual-strategy approach offers a new perspective in cybersecurity research, moving away from the traditional focus on predefined algorithms or feature-dependent models. It enables a comprehensive comparison between the prompt engineering and fine-tuning of LLMs for a specific application.

In this work, we examine various prompt-engineering strategies, such as zero-shot, role-playing, and chain-of-thought prompting, while applying them to two widely used chat models, GPT-3.5-turbo and Claude 2. Our findings reveal that Claude 2 performs better than GPT-3.5-turbo for all prompts and achieves an F1-score of 92.74% with a prompt that combines role-playing and chain-of-thought prompting on a 1000-sample test set sourced from the phishing dataset provided by Hannousse and Yahiouche [17]. While this performance is acceptable given that no training has been conducted on the model, it is much less than what task-specific models with much fewer parameters have achieved in the literature [18].

As such, we delve into investigating how fine-tuning LLMs compares to prompt engineering and previous work. Notably, we fine-tuned foundational LLMs such as GPT-2, Bloom, Baby LLaMA, and DistilGPT-2, which have much less parameters than chat LLMs. These base LLMs were originally designed for text generation, and in this study, we fine-tune them to perform sequence classification to detect phishing URLs. When evaluated on the same 1000-sample test set used for the chat models, the minimal performance achieved for these models is an F1-score of 92.43% for Bloom, which is very similar to the highest performance attained in prompt engineering, and the peak performance achieved is an F1-score of 97.29% for GPT-2 (medium), which surpasses existing state-of-the-art techniques on this task.

To further assess the real-world applicability of these methods, we tested the best fine-tuned and prompt-engineered models on datasets with varying ratios of phishing URLs. Recognizing the importance of realistic testing conditions, we adjusted the phishing URL ratios in our test sets to reflect the varied prevalence of phishing URLs in actual internet traffic. These ratios ranged from as low as 5% to as high as 45%, thereby covering a broad spectrum of potential real-world scenarios. The results show that fine-tuned LLMs have more potential than those used with prompt engineering in real-world scenarios where the proportion of phishing URLs is lower than that of legitimate ones.

These findings underscore that models tailored for particular tasks often outperform general-purpose ones on these tasks, and the rise of LLMs does not negate the necessity for specialized models. Moreover, we show that fine-tuning LLMs to perform specific tasks presents a higher potential than prompt engineering and existing solutions in the literature.

In summary, the main contributions of this work are the following:

- This research is the first to offer a unique comparative analysis between the performance of prompt engineering and fine-tuning techniques for LLMs.
- Exploring prompt-engineering strategies for phishing URL detection and providing valuable insights into their effectiveness.
- The investigation of the fine-tuning of text-generation LLMs for phishing URL detection, showcasing its potential in this domain.
- This study achieves a remarkable performance of 97.29% as the F1-score for phishing URL detection, surpassing existing state-of-the-art techniques.

The rest of this paper is organized as follows: In Section 2, we provide essential background information on LLMs, prompt engineering, fine-tuning, and the challenges associated with phishing URL detection. Understanding these foundational concepts is crucial to grasp the context of our research. Section 3 presents some related work. In Section 4, we detail the methodology employed in our study, including the design and implementation of prompt-engineering strategies and the fine-tuning process. Section 5

offers a comprehensive overview of the experimental setup, experiments, and results. We provide insights into the effectiveness of each approach in Section 6 and compare their outcomes. Section 7 summarizes our key findings and contributions and discusses potential avenues for future research and improvements.

2. Background and Preliminaries

This section provides essential background information on key topics that form the foundation of our study.

2.1. Large Language Models (LLMs)

LLMs are a type of neural-network-based model designed to generate, understand, and interpret human language. They are characterized by their large number of parameters, deep architectures, and the extensive amount of data they are trained on. These models are adept at capturing the complexities and nuances of language, making them valuable for a wide range of applications from text generation to question-answering systems [19,20]. The introduction of the transformer architecture by Vaswani et al. marked a significant milestone in the evolution of LLMs [21]. Their work demonstrated the effectiveness of self-attention mechanisms, leading to substantial improvements in various natural language-processing tasks including language translation, sentiment analysis, and text generation. One of the key aspects of LLMs is their ability to learn contextual representations of words and phrases. Unlike earlier language models that relied on fixed word embeddings [22,23], LLMs use dynamic embeddings, allowing for a more nuanced understanding of language in different contexts.

The landscape of LLMs has witnessed the emergence of several landmark models. The GPT (Generative Pretrained Transformer) by OpenAI is one such example, which showcased the power of unsupervised learning for language understanding and generation [24]. Next, GPT-2 and GPT-3 further pushed the boundaries in terms of size and capabilities, highlighting the scalability of transformer architectures [25,26]. Building on the advancements of GPT-3, models like ChatGPT represent a significant evolution [27]. ChatGPT (https://chat.openai.com/ (accessed on 2 January 2024)) developed by OpenAI, is a variant of the GPT-3 model specifically fine-tuned for conversational responses. This model exemplifies the transition from broad language understanding to specialized, context-aware conversational applications, marking a pivotal step in the practical deployment of LLMs. Nowadays, the trend is shifting to rely on such black box models to build systems and applications without the need to train or maintain ML models.

2.2. Prompt Engineering

Prompt engineering is a strategy employed to harness the capabilities of LLMs for specific tasks. By providing carefully constructed prompts or instructions, LLMs are guided to produce desired responses or perform targeted tasks. Prompt-engineering techniques come in various forms. Some examples include zero-shot prompts, few-shot prompts, role-playing prompts, and chain-of-thought prompts. Zero-shot prompting, for instance, is the standard way of prompting, and it involves framing a task in a way that the LLM can comprehend and generate an appropriate response without explicit training [28]. Few-shot prompting involves giving some examples to the model in order to guide it on how to respond. Typically, it is used to control the output format by providing some examples to follow the structure of their responses and does not provide much help for reasoning [29]. Role-playing prompts encourage the LLM to simulate a particular persona or role when generating responses, enhancing its ability to provide contextually relevant information [30]. Chain-of-thought prompts ask the model to provide the reasoning step by step before reaching the end response. This helps the model make more informed decisions and allows it to understand the reason behind specific decisions [31]. These strategies play a crucial role in our study, where we explore their effectiveness in the context of phishing URL detection.

2.3. Fine-Tuning LLMs

Fine-tuning is a critical process in adapting pretrained LLMs for specialized tasks. It involves training the LLMs on task-specific datasets to improve their performance on particular domains [32]. Fine-tuning allows one to tailor the general language capabilities of LLMs to excel in specific applications, such as phishing URL detection. The process typically begins with a pretrained LLM, such as GPT, which has already learned a broad range of language patterns and semantics from large corpora of text data. Then, models are fine-tuned on a smaller dataset relevant to the specific task, effectively transferring the general language knowledge to the specialized domain [33]. This approach helps LLMs become highly proficient in specific tasks while retaining their overall language understanding. In this study, since the goal is phishing URL detection, we fine-tune LLMs to perform URL classification where they receive a URL as input and predict a class as an output. The process is detailed in the methodology section.

2.4. Phishing Detection

Phishing attacks continue to be a prominent threat in the cybersecurity landscape [34]. According to Cloudflare's 2023 phishing threats report [35], approximately 13 billion emails were processed between May 2022 and May 2023, out of which about 250 million malicious messages were blocked. The report highlights that deceptive links are the most common phishing tactic. Detecting phishing URLs poses a significant challenge due to the sophisticated and constantly evolving tactics of malicious actors. These URLs often closely mimic legitimate websites, making it difficult to distinguish them from genuine ones [36]. Attackers frequently use misleading domain names, embed trusted brand names within URLs, or employ homoglyphs—characters that visually resemble each other—to create seemingly authentic URLs [37,38]. The use of legitimate elements, such as valid TLS certificates [39] and brand logos [40], further complicates their detection. Additionally, the adoption of URL shortening services and redirection tactics helps attackers to conceal the true nature of malicious URLs [41,42]. Attackers' frequent changes in tactics and URL obfuscation underscore the need for a robust understanding of URL structures and content analysis to discern the subtle differences between legitimate and phishing URLs. This study aims to leverage the power of LLMs to effectively identify phishing URLs.

3. Related Work

In the realm of machine-learning-based phishing detection, significant emphasis has been placed on URL-based methods. These methods primarily analyze various features and patterns inherent in URLs, such as the length, the inclusion of special characters, and the utilization of subdomains [10,43,44]. Techniques like lexical and token analysis are instrumental in deciphering URL structures, aiding in the identification of phishing attempts. The application of ML algorithms, including SVM, decision tree, and Random Forest, has been a cornerstone in classifying URLs based on these extracted features [45,46]. Advancements have led to the integration of deep learning techniques, including convolutional neural networks (CNNs) and natural language processing (NLP), further enhancing the automation of cybersecurity tasks [47,48]. Pioneering work by researchers such as Le et al. [49] who proposed the URLNet framework and others like Tajaddodianfar et al. [50] and Jiang et al. [51] showcases the application of character-level deep neural networks in this field. Besides URL-based approaches to detect phishing, other approaches rely on additional information such as the content, appearance, or behavior of the page [52,53]. While these techniques are promising, they might require more computation in order to classify a website as phishing or legitimate, yet they do not offer much more advantage in terms of performance compared to URL-based methods. The focus of this study is to predict phishing websites based solely on URL analysis. Specifically, we rely on the capabilities of LLMs to perform this classification, using prompt engineering and fine-tuning. In both cases, the advantage is that a raw URL can be inputted directly

into the LLM without the need for any kind of feature extraction. Then, both techniques are compared with the performance achieved in state-of-the-art methods.

4. Methodology

In this section, we provide an overview of the methodology employed in our study, detailing the steps taken to investigate the effectiveness of LLMs in detecting phishing URLs by using prompt engineering and fine-tuning techniques.

4.1. Prompt Engineering

Prompt engineering refers to the process of carefully crafting prompts to elicit desired responses from an LLM such as ChatGPT, Google Bard, LLaMA2, etc. In this technique, the architecture of the LLM remains the same; only the input prompt is altered to observe its impact on the output. To investigate how prompt-engineering strategies affect the abilities of chat-completion LLMs in detecting phishing URLs, we use a subset of 1000 URLs for testing. Feeding all URLs simultaneously to the model is impractical as it would exceed the allowed context length. Therefore, we adopt the following process:

1. We divide the list of 1000 URLs into 20 groups, each containing 50 URLs.
2. For each subset, we craft a prompt asking the LLM to classify each URL within it, specifying the response format.
3. We feed the prompt to the LLM.
4. We collect the responses, which adhere to the requested format.
5. We aggregate the responses from all groups and convert them into a data frame for analysis. This allows us to compute classification metrics by comparing them with ground-truth data.

These steps are illustrated in Figure 1 by using a basic zero-shot prompt. The experiments section provides more details about the various prompt types and chat-completion LLMs, but these steps remain consistent throughout.

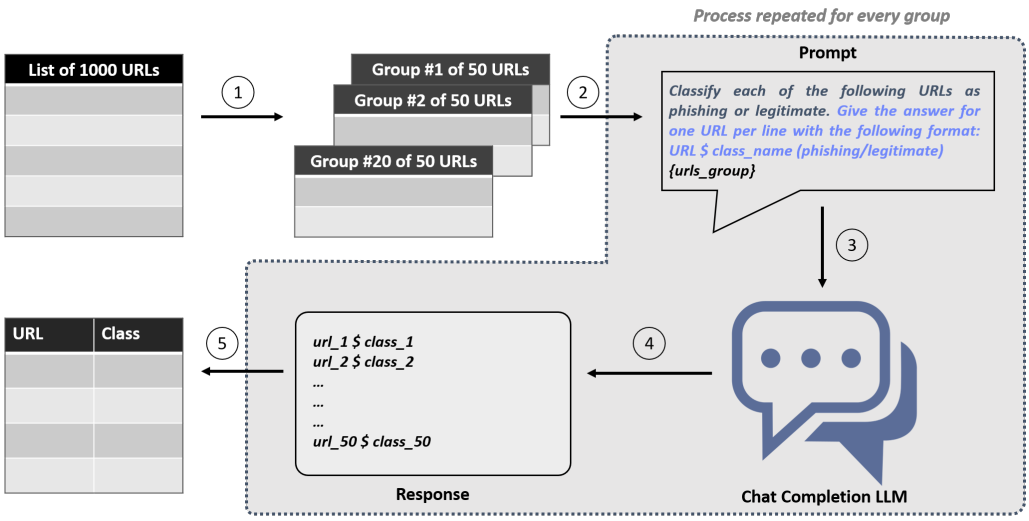

Figure 1. Prompt-engineering process.

4.2. Fine-Tuning

The goal of fine-tuning an LLM is to tailor it more specifically for a particular task. In this study, we investigate the fine-tuning of pretrained text-generation LLMs for phishing URL detection. For all LLMs used, we follow a consistent fine-tuning process. This involves loading the LLM with pretrained weights for the embedding and transformer layers and adding a classification head on top, which categorizes a given URL as phishing or legitimate. This makes the LLM dedicated to performing URL classification. For the data to be processed by the LLM, it must be tokenized. For each LLM, we use its corresponding tokenizer, setting a maximum length of 100 tokens with right padding. Then, we

train the complete architecture for several epochs on the training data while tuning some hyperparameters on the validation data. Finally, we evaluate the model by using the same 1000 testing samples as in the prompt-engineering method. The full architecture through which a URL is processed for classification is depicted in Figure 2. The specific models utilized for fine-tuning are detailed in the experiments section.

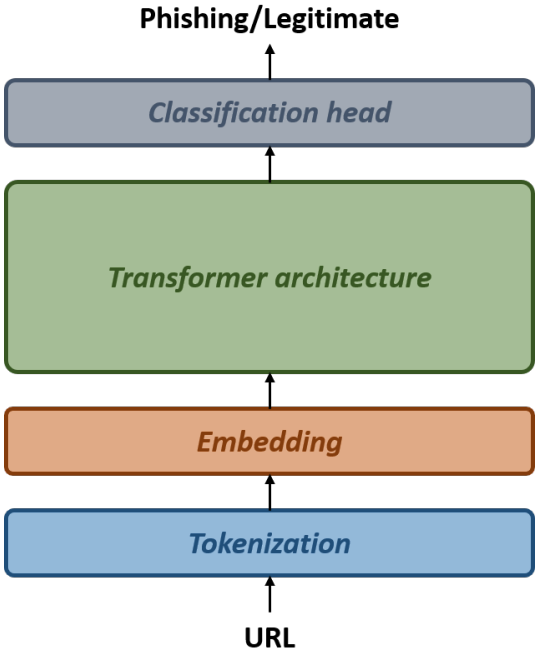

Figure 2. Fine-tuning process.

5. Experiments

5.1. Experimental Setup

5.1.1. Dataset

In this study, we utilize the phishing dataset provided by Hannousse and Yahiouche [17], available for download at Mendeley Data (https://data.mendeley.com/datasets/c2gw7fy2j4/3 (accessed on 2 January 2024)), comprising a total of 11,430 URL samples. The dataset is balanced, containing an equal number of phishing and legitimate URLs, with each category represented by 5215 samples. Each URL in the dataset is accompanied by 87 extracted features and a classification label denoting whether it is legitimate or phishing. Details about the data collection and feature-extraction processes can be found in [54].

For the purpose of this study, we focus exclusively on analyzing the raw URLs by using LLMs while disregarding the extracted features. This approach enables us to evaluate the LLMs' capability to discern phishing URLs based solely on their textual characteristics.

To assess the performance of the prompt engineering and fine-tuning methods, we extract a subset of 1000 samples from the dataset. This subset maintains the dataset's balanced nature, comprising 500 phishing and 500 legitimate URLs. The remaining 10,430 samples are divided into training and validation sets, with 90% (9387 samples) used for training and 10% (1043 samples) for validation. This division is employed to fine-tune the LLMs effectively for the phishing URL detection task. For all divisions of the dataset, we ensure consistency and reproducibility by setting the random_state to 42 by using the scikit-learn Python library [55].

5.1.2. Models

For prompt engineering, we utilized two chat models, GPT-3.5-turbo and Claude 2. These models are accessible via API, making them suitable for developing AI applications,

which is why we selected them. For fine-tuning, we chose LLMs that specialize in text generation, focusing on those with a minimal number of parameters. This approach allows us to compare their performance with that of chat models, which possess a significantly higher number of parameters. Specifically, we selected the following Hugging Face models [56]: openai-gpt, gpt2, gpt2-medium, distilgpt2, baby-llama-58m, and bloom-560m. For all models, we used the Adam optimizer, a batch size of 16, and a learning rate of 5×10^{-5}. The number of training epochs was determined by early stopping, based on the validation data, to avoid overfitting. The fine-tuning parameters are presented in Table 1.

Table 1. Number of training epochs for each LLM.

LLM	openai-gpt	gpt2	gpt2-medium	distilgpt2	baby-llama-58m	bloom-560m
Epochs	5	5	3	5	5	7

5.1.3. Evaluation Metrics

In both prompt engineering and fine-tuning, evaluating the performance of LLMs is crucial. Since the goal is to classify URLs as phishing or legitimate, we use the following classification metrics:

- Accuracy: This is the most intuitive performance measure and is simply the ratio of correctly predicted observations to the total observations. It is particularly useful when the target classes are well-balanced. However, its utility is limited in scenarios with significant class imbalance, as it can yield misleading results.
- Precision: Also known as the positive predictive value, precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision, which indicates a low rate of false positives, is critical in phishing detection, where mistakenly labeling legitimate URLs as phishing can have serious consequences.
- Recall: Also referred to as sensitivity, recall is the ratio of correctly predicted positive observations to all actual positives. This metric is essential in phishing detection as it is vital to identify as many phishing instances as possible to prevent data breaches.
- F1-score: The F1-score is the harmonic mean of precision and recall. It is a more reliable measure than accuracy, particularly when dealing with unevenly distributed datasets. It considers both false positives and false negatives, making it suitable for scenarios where both precision and recall are important.

When comparing our results with state-of-the-art models, we employ additional metrics such as:

- Area Under the Curve (AUC): this metric measures the ability of a classifier to distinguish between classes and is used as a summary of the Receiver Operating Characteristic (ROC) curve.
- True Positive Rate at a Given False Positive Rate (TPR@FPR): This metric evaluates the model's ability to correctly identify positives at a specific false positive rate. It is particularly useful in scenarios where maintaining a low rate of false positives is crucial, which is the case in phishing detection.

5.2. Prompt Engineering

We use the flow outlined in the methodology section and we apply it to two chat-completion models: GPT-3.5-turbo and Claude 2. We vary the input prompts to observe how this affects the results. Notably, we investigate three prompt templates for phishing URL detection, as shown in Figure 3:

- Prompt 1 (zero-shot): We start with a baseline prompt that simply asks the LLM to classify the given URLs while generating the response according to a specific output format. The accuracy, precision, recall, and F1-score of both LLMs for this prompt on the test set are reported in Figure 4.

- Prompt 2 (role-playing): We modify the baseline prompt to ask the LLM to assume the role of a cybersecurity specialist analyzing URLs for a company. This approach is intended to help the model adopt a specific mindset while responding, which is expected to enhance its responses. We apply this prompt to both LLMs, and the results are shown in Figure 5.
- Prompt 3 (chain-of-thought): We further modify the second prompt (role-playing) by asking the model to provide succinct reasoning for classifying a given URL as phishing or legitimate. This approach encourages the LLM to classify based on specific criteria that it articulates, which is expected to improve performance. The results of this prompt for both LLMs are illustrated in Figure 6.

During this process, a notable observation was made with GPT-3.5-turbo. The model occasionally refrained from classifying certain URLs due to internal content filters being triggered by specific words. This observation was very minor, affecting only one URL for prompts 1 and 2 and no URLs for prompt 3. Consequently, it did not impact the overall results of the study.

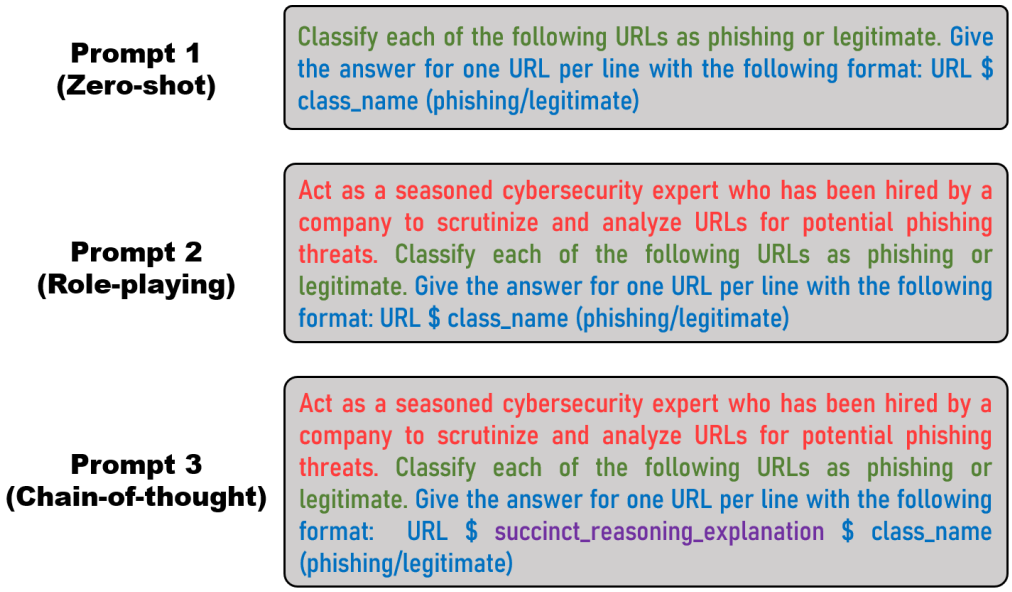

Figure 3. The three investigated prompts.

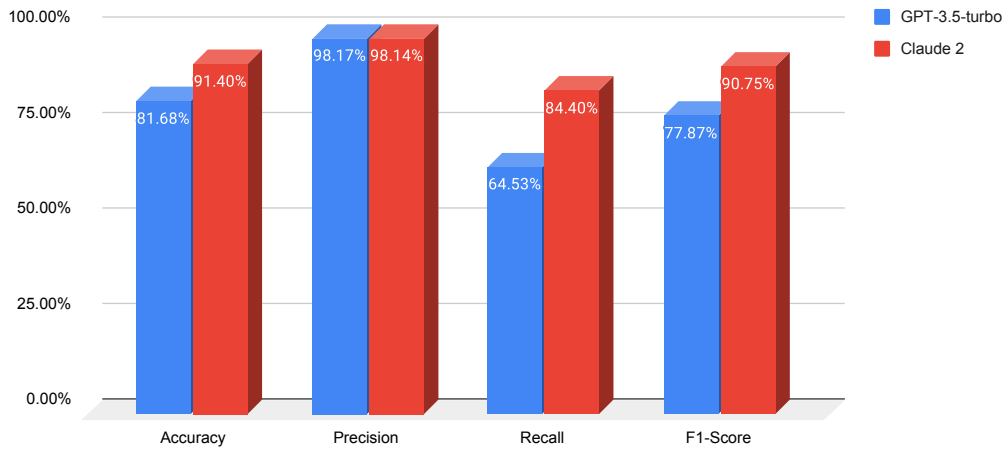

Figure 4. Performance metrics of GPT-3.5-turbo and Claude 2 using the zero-shot prompt.

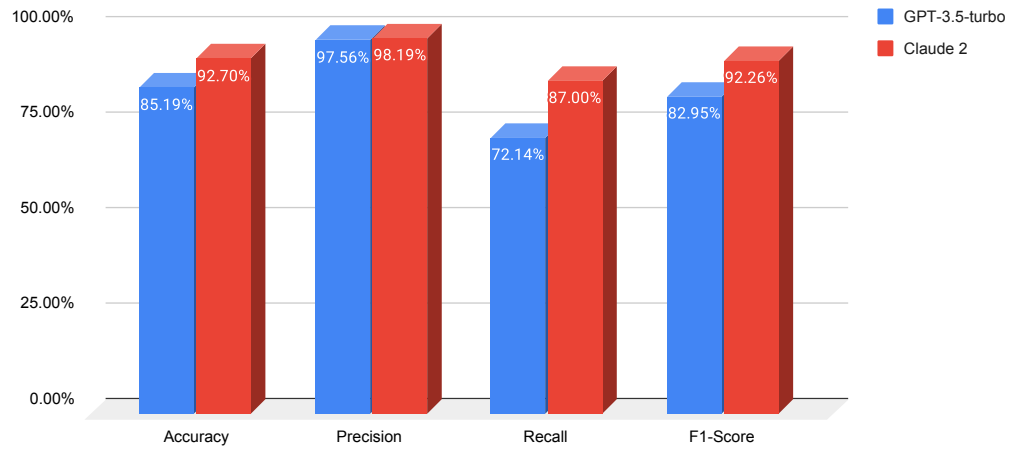

Figure 5. Performance metrics of GPT-3.5-turbo and Claude 2 using the role-playing prompt.

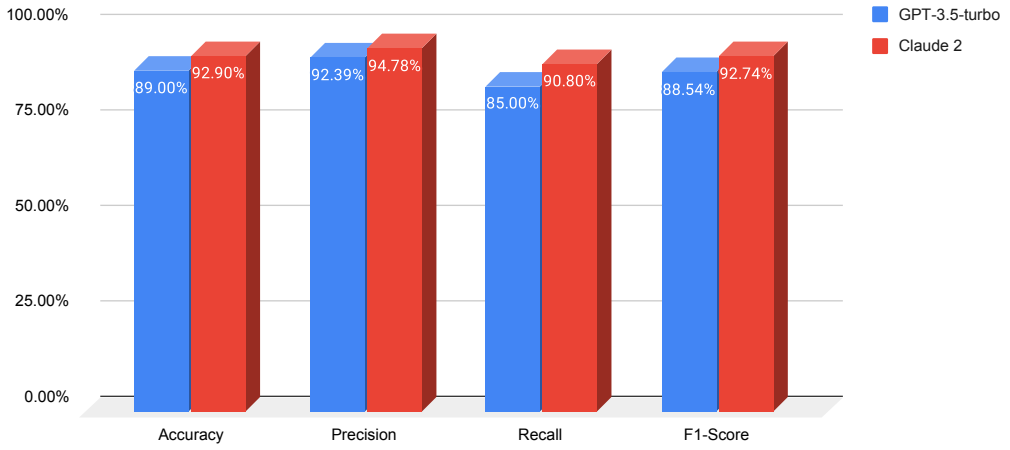

Figure 6. Performance metrics of GPT-3.5-turbo and Claude 2 using the chain-of-thought prompt.

5.3. Fine-Tuning

For fine-tuning, we consider several base models that are pretrained for text generation. We fine-tune each of these models following the steps detailed in the methodology section. The LLMs we fine-tuned, available on Hugging Face, are as follows:

- openai-gpt: The first iteration of the Generative Pretrained Transformer models developed by OpenAI. It provides a solid baseline for natural language understanding and generation tasks and has 110 million parameters.
- gpt2: An improved version of the original GPT, GPT-2 offers a larger model size for enhanced performance across a broader range of tasks and the ability to generate more coherent and contextually relevant text. The version we used is the smallest and has 117 million parameters.
- gpt2-medium: A midsized variant of GPT-2, this model balances computational efficiency and performance, suitable for tasks requiring in-depth language understanding without the largest model size. It has 345 million parameters.
- distilgpt2: A distilled version of GPT-2 that retains most of the original model's performance but with fewer parameters, enhancing efficiency without a significant loss in quality. It has 82 million parameters.
- baby-llama-58m: A smaller model with 58 million parameters, Baby LLaMA is a distilled version of small LLaMA models and GPT-2 [57]. It is designed for efficiency and can perform various language tasks, optimized for environments with limited computational resources.

- bloom-560m: Part of the Bloom series [58], this large-scale multilingual language model is designed to understand and generate text in multiple languages. With 560 million parameters, it offers substantial capability in language-processing tasks.

We assess the fine-tuned LLMs by using the same test set that was utilized for assessing prompt-engineering techniques. The results are presented in Table 2, organized in ascending order based on their F1-scores.

Table 2. Performance metrics of the fine-tuned LLMs.

Model	Accuracy	Precision	Recall	F1-Score
bloom-560m	92.40%	92.06%	92.80%	92.43%
distilgpt2	95.90%	94.22%	97.80%	95.98%
openai-gpt	96.10%	96.01%	96.20%	96.10%
baby-llama-58m	96.60%	96.05%	97.20%	96.62%
gpt2	96.60%	95.87%	97.40%	96.63%
gpt2-medium	97.30%	97.78%	96.80%	97.29%

6. Discussion

Our investigation into the effectiveness of prompt engineering and fine-tuning strategies for LLMs in phishing URL detection has provided new insights. In this section, we discuss the results achieved with each approach. Subsequently, we compare these approaches both against each other and with state-of-the-art methodologies. Finally, we delve into the performance of these models under imbalanced test conditions, which simulate real-world scenarios where phishing URLs are less prevalent than legitimate ones.

6.1. Prompt Engineering

In our assessment of prompt-engineering capabilities using both LLMs, we observed an increase in performance as the prompts were refined. Notably, Claude 2 outperformed GPT-3.5-turbo. To compare the effectiveness of the various prompt types, we report the F1-scores achieved for each when applied to both LLMs, and we present these results in Figure 7 for easier comparison. Specifically, the zero-shot prompt yielded F1-scores of 77.87% for GPT-3.5-turbo and 90.75% for Claude 2. The role-playing prompt, which asks the LLM to act as a cybersecurity consultant, significantly improved the performance of both models, resulting in F1-scores of 82.95% for GPT-3.5-turbo and 92.26% for Claude 2. Lastly, the chain-of-thought prompt, which requires the LLM to provide reasoning behind its classification, further enhanced the performance, achieving F1-scores of 88.54% for GPT-3.5-turbo and 92.74% for Claude 2.

We noticed that Claude 2 consistently outperformed GPT-3.5-turbo across all prompt types. However, the reason for this is not entirely clear, as both models offer limited information about their training processes and are generally treated as 'black boxes' by users. Interestingly, when the prompts were improved, GPT-3.5-turbo showed more significant gains than Claude 2, likely because Claude's baseline performance was already high (an F1-score of 90.75% for the zero-shot prompt). Therefore, its room for improvement was more limited compared to GPT-3.5-turbo, which started with a baseline F1-score of 77.87%.

On the other hand, the results achieved with prompt engineering are remarkable, considering that no specific training was conducted to enable the LLMs to distinguish between phishing and legitimate URLs. The effectiveness of a simple zero-shot prompt in detecting phishing demonstrates the inherent capabilities of such models. Moreover, throughout all prompt-engineering techniques, we observed a trend where precision was consistently higher than recall. This likely indicates that the LLMs, when prompted, were more inclined to accurately identify true positive cases (legitimate URLs correctly identified as legitimate) but were somewhat less effective in correctly identifying all phishing instances, leading to a higher rate of false negatives. This pattern suggests that while LLMs

were efficient in minimizing false positives, this was at the expense of potentially missing some phishing cases.

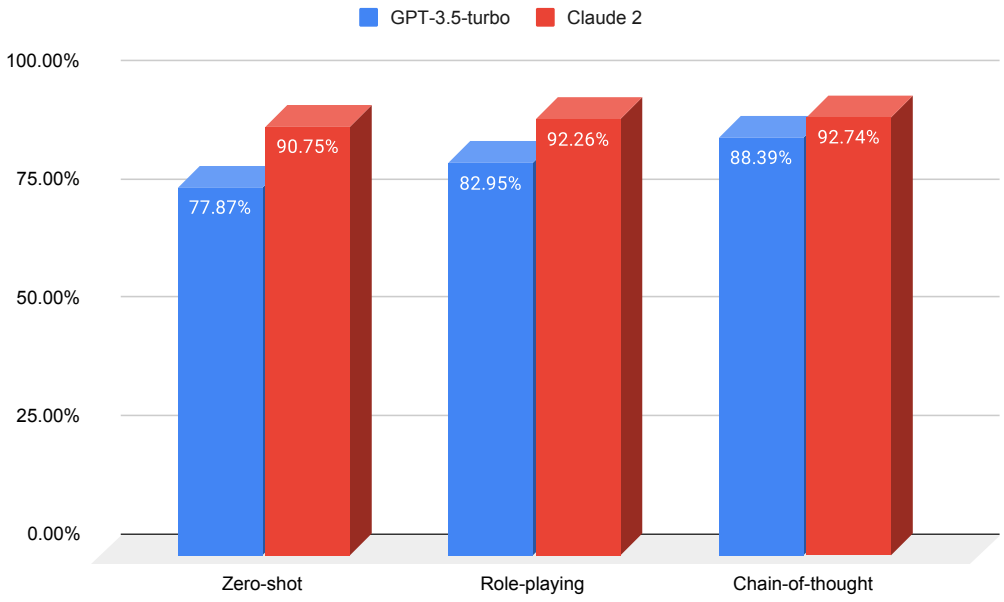

Figure 7. Comparing the performance of the three used prompts on GPT-3.5-turbo and Claude 2.

6.2. Fine-Tuning

When fine-tuning, we observe that LLMs achieve a very high performance with minimal training, such as after only a few epochs. It is noteworthy that the GPT models outperform Bloom, despite the latter having more parameters. This discrepancy could be attributed to the different training settings used for each model. The Baby LLaMA model achieved a performance comparable to the GPT family, which can be explained by its design, as it is distilled from both LLaMA models and GPT. The lowest performance was achieved by bloom-560m with an F1-score of 92.43%. The highest performance was recorded by gpt2-medium, with an F1-score of 97.29%, surpassing the state-of-the-art model, a point we discuss further later in this section.

6.3. Comparing Prompt Engineering and Fine-Tuning

In this part, we compare prompt engineering and fine-tuning LLMs from multiple aspects.

- Performance differences: In our analysis, we found significant performance differences between prompt-engineered and fine-tuned LLMs. Fine-tuning generally results in better performance. For example, the least performing fine-tuned model, bloom-560m, achieved an F1-score of 92.43%, which is comparable to the highest performance in prompt engineering with the chain-of-thought prompt on Claude 2 (92.74%). Notably, GPT-2-medium, with fewer parameters, significantly outperforms prompt-engineered models, achieving an F1-score of 97.29%. This is particularly striking when considering that chat-completion models like GPT-3.5-turbo and Claude 2 have substantially more parameters. This disparity highlights the potential of fine-tuning in enhancing the specialization and effectiveness of smaller models for specific tasks like phishing URL detection.
- Data privacy and security: When using prompt engineering, interacting with LLMs via their APIs, as commonly performed in AI development, involves data transmission to third-party servers. This raises data privacy and security concerns. In contrast, fine-tuning as outlined in this study generally involves downloading the model for local adjustments, which enhances data security and minimizes risks of data leakage.

- Resource requirements: The resource demands of the two approaches differ significantly. Prompt engineering is generally less resource intensive, requiring minimal adjustments to apply various prompts. This makes it more accessible and practical, particularly in resource-limited settings. On the other hand, fine-tuning demands more substantial resources, including a significant amount of domain-specific training data and computational power, which can be a limiting factor in its scalability and practicality.
- Model maintenance: The maintenance approaches for prompt-engineered and fine-tuned models differ considerably. For prompt-engineered models like GPT-3.5-turbo and Claude 2, updates and maintenance are typically handled by the companies that developed them, such as OpenAI for GPT-3.5-turbo and Anthropic AI for Claude 2. This means that improvements and adaptations to evolving data or tasks are managed centrally, relieving users from the burden of continual model updates. However, this also means that users are dependent on the companies for timely updates. In contrast, fine-tuned models require the users to actively manage and update the models. This might involve retraining the models as new data become available or as the nature of tasks, such as phishing URL detection, evolves. While this allows for more control and customization, it also adds to the resource intensity and demands ongoing attention from the users.

To conclude, our analysis of prompt engineering and fine-tuning in LLMs has illuminated distinct strengths and limitations for each method. Prompt engineering, while offering flexibility and lower resource requirements, tends to fall short in performance compared to fine-tuning. This is particularly evident when considering the superior outcomes achieved by fine-tuning, even with models having fewer parameters than large chat-completion models like GPT-3.5-turbo and Claude 2. Fine-tuning, despite its higher demands for resources and active maintenance, provides a notable increase in specialization and effectiveness, especially in specific tasks such as phishing URL detection. Additionally, fine-tuning affords enhanced data security through local processing as opposed to the potential privacy concerns associated with using third-party servers in prompt engineering. The choice between these approaches should be made based on the specific requirements of the task at hand, weighing factors such as performance, data security, resource availability, and the need for ongoing model maintenance and adaptability.

6.4. Comparison with State-of-the-Art Approaches

In this section, we compare the best results achieved by both methods with state-of-the-art approaches. Initially, we contrast these results with studies that used the same dataset to ensure a fair comparison, as shown in Table 3. Although prompt engineering theoretically performs well, it did not achieve the performance level of state-of-the-art methods, indicating that prompt-engineering techniques prioritize convenience over performance. However, it is evident that the fine-tuned GPT-2 model surpasses state-of-the-art techniques in all metrics, demonstrating the significant potential of fine-tuning LLMs for specific tasks.

Table 3. Comparing our results with previous studies using our same dataset.

Study	Year	Accuracy	Precision	Recall	F1-Score	AUC	Model
Nepal et al. [59]	2022	94.30%	94.51%	94.59%	94.55%	-	CNN-LSTM
McConnell et al. [60]	2023	95.36%	96.29%	94.24%	95.26%	98.76%	Gradient Boosting
Rashid and Abdullah [61]	2023	96.41%	97.09%	95.80%	96.44%	-	Amazon Sagemaker—XGBoost
Uppalapati et al. [62]	2023	97.05%	97.27%	96.75%	97.01%	-	XGBoost
Our study	2024	92.90%	94.78%	90.80%	92.74%	-	Claude 2—prompt engineering
Our study	2024	97.30%	97.78%	96.80%	97.29%	99.56%	Fine-tuned GPT-2

Additionally, in Table 4, we compare the results achieved by the fine-tuned GPT-2 with two state-of-the-art models that were not trained on the same dataset. The purpose of this comparison is to provide an approximate indication of how the fine-tuned GPT-2 performs relative to these models. Specifically, we compare it with PhishBERT, the only model that uses LLMs for phishing detection, and URLNet, a state-of-the-art model that processes raw URLs in a manner similar to LLMs. This contrasts with other techniques that necessitate feature extraction.

Table 4. Comparing our results with previous studies using different datasets.

Study	Year	AUC	TPR@FPR 0.1	TPR@FPR 0.01	Model
Wang et al. [63]	2023	-	-	89.31%	PhishBERT
Le et al. [49]	2018	99.29%	98.58%	90.84%	URLNet
Our study	2024	99.56%	98.80%	91.20%	Fine-tuned GPT-2

6.5. Testing on Imbalanced Datasets

Evaluating models on imbalanced test sets is essential in phishing detection to mirror real-world conditions, where phishing URLs are typically less frequent than legitimate ones. Balanced datasets may not accurately represent these real-life scenarios, often leading to an overestimation of the model performance [64]. To address this, we selected the best-performing models from our initial experiments: Claude 2, used with a chain-of-thought prompt, and the fine-tuned gpt2-medium. These models were then assessed on imbalanced datasets to reflect the varying prevalence of phishing URLs in real-world settings. Given the uncertainty in the actual distribution of phishing versus legitimate URLs in real-world settings, we altered the phishing ratios in our test sets to include 5%, 10%, 15%, 20%, 25%, 30%, 35%, 40% and 45%, all derived from our original balanced test set. This method allowed for an evaluation of the models' effectiveness across different scenarios, particularly in environments where phishing is less prevalent than legitimate content.

Across the varying phishing ratios, we observed significant differences in the effectiveness of the models. Claude 2 showed increased performance at higher phishing ratios, indicating better detection in datasets with a larger presence of phishing URLs. However, in scenarios more representative of real-world conditions, where phishing URLs are less common, the fine-tuned GPT-2 consistently outperformed Claude 2 across all metrics. This was particularly noticeable at lower phishing ratios. For example, at 5% and 10% phishing ratios, GPT-2 achieved F1-scores of 76.19% and 86.73%, respectively, significantly surpassing Claude 2's F1-scores of 59.15% and 74.58%, respectively. This trend of superior performance by GPT-2-medium persisted across all evaluated ratios, culminating in an F1-score of 96.02% at a 45% phishing ratio, demonstrating its enhanced effectiveness in realistic settings with a lower incidence of phishing. The results, including precision, recall, and F1-scores for both models at the varying phishing URL ratios, are presented in Figures 8–10. Notably, these findings suggest that fine-tuned LLMs are more adept than their prompt-engineered counterparts in handling real-world scenarios.

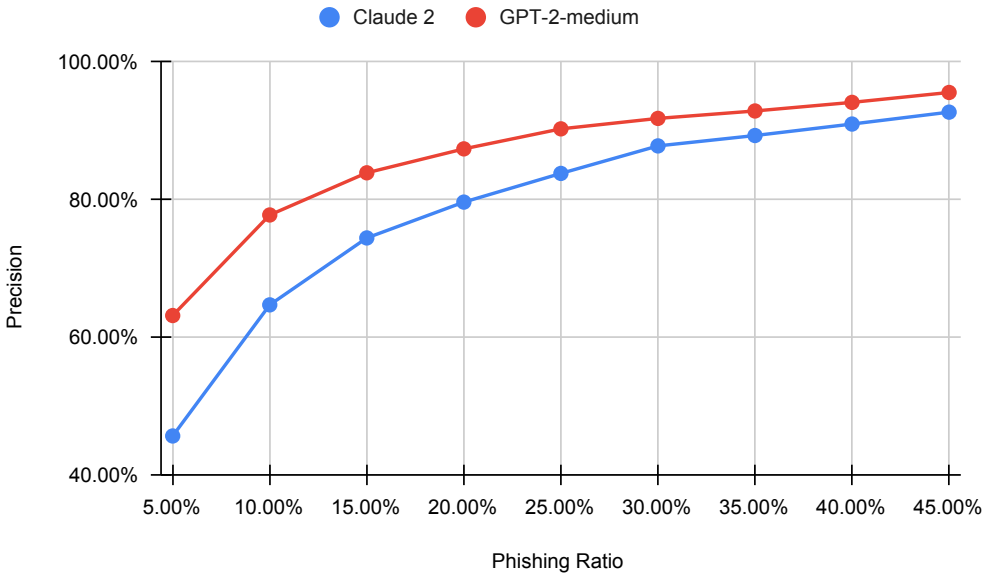

Figure 8. Precision of GPT-2-medium and Claude 2 with varying phishing URL ratios.

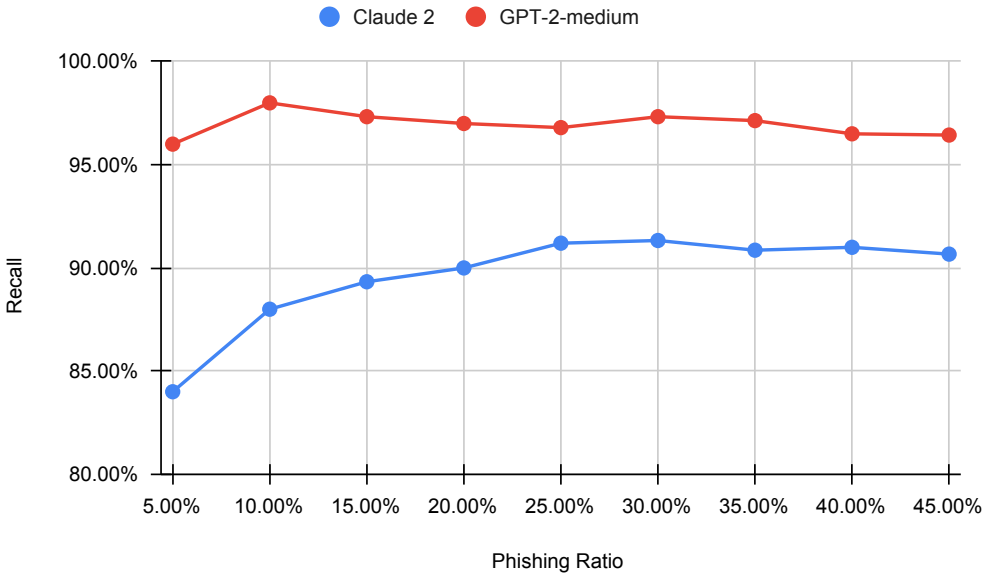

Figure 9. Recall of GPT-2-medium and Claude 2 with varying phishing URL ratios.

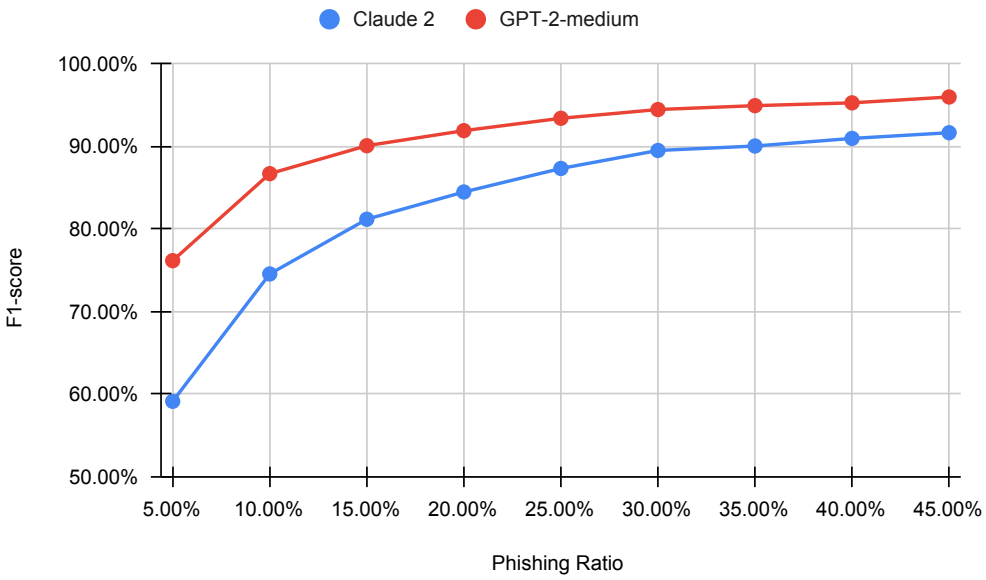

Figure 10. F1-score of GPT-2-medium and Claude 2 with varying phishing URL ratios.

7. Conclusions

In this study, we explored the effectiveness of LLMs in detecting phishing URLs, focusing on prompt engineering and fine-tuning strategies. Our investigation encompassed a variety of prompt-engineering mechanisms, as well as multiple LLMs for fine-tuning. We found that although prompt engineering facilitates the construction of AI systems without the need for training or monitoring ML models, it does not match the superior performance of the fine-tuned LLMs. Notably, the fine-tuned LLMs achieved an F1-score of 97.29%, surpassing current state-of-the-art benchmarks. Moreover, in realistic scenarios where the ratio of phishing to legitimate URLs is low, the fine-tuned LLMs significantly outperformed those used with prompt engineering. This work presents a detailed comparison between prompt engineering and the fine-tuning of LLMs, offering comprehensive insights into the appropriate scenarios for employing each technique.

For future research, we suggest exploring hybrid approaches that combine the convenience of prompt engineering with the high performance of fine-tuning in phishing URL detection. It is also crucial to address the resilience of LLM-based detection methods against adversarial attacks, necessitating the development of robust defense mechanisms. Additionally, optimizing real-time detection systems, mitigating biases in LLMs, and incorporating multimodal cues for enhanced detection accuracy are key areas that warrant further investigation and research. These efforts will contribute to more effective and reliable phishing-detection tools in the rapidly evolving landscape of cybersecurity.

Author Contributions: F.T.: conceptualization, methodology, software, visualization, validation, data curation, and writing—original draft preparation; A.C.: methodology, supervision, project administration, and writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: The authors would like to acknowledge that this work has been supported by the Maroun Semaan Faculty of Engineering and Architecture (MSFEA) at the American University of Beirut (AUB), and the APC was funded by MSFEA.

Data Availability Statement: The data used in this study are publicly available online. No new data were created.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mustroph, H.; Winter, K.; Rinderle-Ma, S. Social Network Mining from Natural Language Text and Event Logs for Compliance Deviation Detection. In *Cooperative Information Systems. CoopIS 2023*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2024; Volume 14353, pp. 347–365, ISBN 9783031468452. [CrossRef]
2. Liu, Z.; Zhong, A.; Li, Y.; Yang, L.; Ju, C.; Wu, Z.; Ma, C.; Shu, P.; Chen, C.; Kim, S.; et al. Tailoring Large Language Models to Radiology: A Preliminary Approach to LLM Adaptation for a Highly Specialized Domain. In *Machine Learning in Medical Imaging. MLMI 2023*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2024; Volume 14348, pp. 464–473, ISBN 9783031456725. [CrossRef]
3. Kirshner, S. GPT and CLT: The impact of ChatGPT's level of abstraction on consumer recommendations. *J. Retail. Consum. Serv.* **2024**, *76*, 103580. [CrossRef]
4. Caruccio, L.; Cirillo, S.; Polese, G.; Solimando, G.; Sundaramurthy, S.; Tortora, G. Can ChatGPT provide intelligent diagnoses? A comparative study between predictive models and ChatGPT to define a new medical diagnostic bot. *Expert Syst. Appl.* **2024**, *235*, 121186. [CrossRef]
5. Shi, Y.; Ren, P.; Wang, J.; Han, B.; ValizadehAslani, T.; Agbavor, F.; Zhang, Y.; Hu, M.; Zhao, L.; Liang, H. Leveraging GPT-4 for food effect summarization to enhance product-specific guidance development via iterative prompting. *J. Biomed. Inform.* **2023**, *148*, 104533. [CrossRef]
6. Escalante, J.; Pack, A.; Barrett, A. AI-generated feedback on writing: Insights into efficacy and ENL student preference. *Int. J. Educ. Technol. High. Educ.* **2023**, *20*, 57. [CrossRef]
7. Dhamija, R.; Tygar, J.D.; Hearst, M. Why phishing works. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, NY, USA, 22–27 April 2006; pp. 581–590. [CrossRef]
8. Moghimi, M.; Varjani, A.Y. New rule-based phishing detection method. *Expert Syst. Appl.* **2016**, *53*, 231–242. [CrossRef]
9. Mohammad, R.M.; Thabtah, F.; McCluskey, L. Intelligent rule-based phishing websites classification. *IET Inf. Secur.* **2014**, *8*, 153–160. [CrossRef]
10. Sahingoz, O.K.; Buber, E.; Demir, O.; Diri, B. Machine learning based phishing detection from URLs. *Expert Syst. Appl.* **2019**, *117*, 345–357. [CrossRef]
11. Tang, L.; Mahmoud, Q.H. A Survey of Machine Learning-Based Solutions for Phishing Website Detection. *Mach. Learn. Knowl. Extr.* **2021**, *3*, 672–694. [CrossRef]
12. Benavides, E.; Fuertes, W.; Sanchez, S.; Sanchez, M. Classification of Phishing Attack Solutions by Employing Deep Learning Techniques: A Systematic Literature Review. In *Developments and Advances in Defense and Security*; Smart Innovation, Systems and Technologies; Rocha, A., Pereira, R.P., Eds.; Springer: Singapore, 2020; pp. 51–64. [CrossRef]
13. Catal, C.; Giray, G.; Tekinerdogan, B.; Kumar, S.; Shukla, S. Applications of deep learning for phishing detection: A systematic literature review. *Knowl. Inf. Syst.* **2022**, *64*, 1457–1500. [CrossRef] [PubMed]
14. Do, N.Q.; Selamat, A.; Krejcar, O.; Herrera-Viedma, E.; Fujita, H. Deep Learning for Phishing Detection: Taxonomy, Current Challenges and Future Directions. *IEEE Access* **2022**, *10*, 36429–36463. [CrossRef]
15. White, J.; Fu, Q.; Hays, S.; Sandborn, M.; Olea, C.; Gilbert, H.; Elnashar, A.; Spencer-Smith, J.; Schmidt, D.C. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. *arXiv* **2023**, arXiv:2302.11382. [CrossRef]
16. Lv, K.; Yang, Y.; Liu, T.; Gao, Q.; Guo, Q.; Qiu, X. Full Parameter Fine-tuning for Large Language Models with Limited Resources. *arXiv* **2023**, arXiv:2306.09782. [CrossRef]
17. Hannousse, A.; Yahiouche, S. *Web Page Phishing Detection*; Mendeley Data: Amsterdam, The Netherlands, 2021; Volume 3. [CrossRef]
18. Asif, A.U.Z.; Shirazi, H.; Ray, I. Machine Learning-Based Phishing Detection Using URL Features: A Comprehensive Review. In *Stabilization, Safety, and Security of Distributed Systems*; Lecture Notes in Computer Science; Dolev, S., Schieber, B., Eds.; Springer: Cham, Switzerland, 2023; pp. 481–497. [CrossRef]
19. Zhao, W.X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. A Survey of Large Language Models. *arXiv* **2023**, arXiv:2303.18223. [CrossRef]
20. Yang, J.; Jin, H.; Tang, R.; Han, X.; Feng, Q.; Jiang, H.; Yin, B.; Hu, X. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. *arXiv* **2023**, arXiv:2304.13712. [CrossRef]
21. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2017; Volume 30.
22. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2023**, arXiv:1301.3781. [CrossRef]
23. Pennington, J.; Socher, R.; Manning, C. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543. [CrossRef]
24. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. *Improving Language Understanding by Generative Pre-Training*; OpenAI: San Francisco, CA, USA, 2018.
25. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.

26. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. *arXiv* **2020**, arXiv:2005.14165. [CrossRef]
27. Ray, P.P. ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet Things-Cyber-Phys. Syst.* **2023**, *3*, 121–154. [CrossRef]
28. Kojima, T.; Gu, S.S.; Reid, M.; Matsuo, Y.; Iwasawa, Y. Large Language Models are Zero-Shot Reasoners. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 22199–22213.
29. Ye, X.; Durrett, G. The Unreliability of Explanations in Few-shot Prompting for Textual Reasoning. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 30378–30392.
30. Kong, A.; Zhao, S.; Chen, H.; Li, Q.; Qin, Y.; Sun, R.; Zhou, X. Better Zero-Shot Reasoning with Role-Play Prompting. *arXiv* **2023**, arXiv:2308.07702. [CrossRef]
31. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv* **2023**, arXiv:2201.11903. [CrossRef]
32. Hu, Z.; Wang, L.; Lan, Y.; Xu, W.; Lim, E.P.; Bing, L.; Xu, X.; Poria, S.; Lee, R.K.W. LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. *arXiv* **2023**, arXiv:2304.01933. [CrossRef]
33. Howard, J.; Ruder, S. Universal Language Model Fine-tuning for Text Classification. *arXiv* **2018**, arXiv:1801.06146. [CrossRef]
34. Wang, Y.; Ma, W.; Xu, H.; Liu, Y.; Yin, P. A Lightweight Multi-View Learning Approach for Phishing Attack Detection Using Transformer with Mixture of Experts. *Appl. Sci.* **2023**, *13*, 7429. [CrossRef]
35. Introducing Cloudflare's 2023 Phishing Threats Report. 2023. Available online: https://blog.cloudflare.com/2023-phishing-report (accessed on 8 January 2024).
36. Sahoo, D.; Liu, C.; Hoi, S.C.H. Malicious URL Detection using Machine Learning: A Survey. *arXiv* **2019**, arXiv:1701.07179. [CrossRef]
37. Woodbridge, J.; Anderson, H.S.; Ahuja, A.; Grant, D. Detecting homoglyph attacks with a siamese neural network. In Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 24 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 22–28.
38. Sern, L.J.; David, Y.G.P.; Hao, C.J. PhishGAN: Data Augmentation and Identification of Homoglyph Attacks. In Proceedings of the 2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI), Virtual, 3–5 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
39. Hageman, K.; Kidmose, E.; Hansen, R.R.; Pedersen, J.M. Can a TLS certificate be phishy? In Proceedings of the 18th International Conference on Security and Cryptography, SECRYPT 2021, Online, 6–8 July 2021; SCITEPRESS Digital Library: Setúbal, Portugal, 2021; pp. 38–49.
40. Bozkir, A.S.; Aydos, M. LogoSENSE: A companion HOG based logo detection scheme for phishing web page and E-mail brand recognition. *Comput. Secur.* **2020**, *95*, 101855. [CrossRef]
41. da Silva, C.M.R.; Feitosa, E.L.; Garcia, V.C. Heuristic-based strategy for Phishing prediction: A survey of URL-based approach. *Comput. Secur.* **2020**, *88*, 101613. [CrossRef]
42. Chhabra, S.; Aggarwal, A.; Benevenuto, F.; Kumaraguru, P. Phi.sh/$oCiaL: The phishing landscape through short URLs. In Proceedings of the 8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference, New York, NY, USA, 1–2 September 2011; pp. 92–101. [CrossRef]
43. Wei, W.; Ke, Q.; Nowak, J.; Korytkowski, M.; Scherer, R.; Woźniak, M. Accurate and fast URL phishing detector: A convolutional neural network approach. *Comput. Netw.* **2020**, *178*, 107275. [CrossRef]
44. Zouina, M.; Outtaj, B. A novel lightweight URL phishing detection system using SVM and similarity index. *Hum.-Centric Comput. Inf. Sci.* **2017**, *7*, 17. [CrossRef]
45. Mahajan, R.; Siddavatam, I. Phishing Website Detection using Machine Learning Algorithms. *Int. J. Comput. Appl.* **2018**, *181*, 45–47. [CrossRef]
46. Ahammad, S.H.; Kale, S.D.; Upadhye, G.D.; Pande, S.D.; Babu, E.V.; Dhumane, A.V.; Bahadur, M.D.K.J. Phishing URL detection using machine learning methods. *Adv. Eng. Softw.* **2022**, *173*, 103288. [CrossRef]
47. Huang, Y.; Yang, Q.; Qin, J.; Wen, W. Phishing URL Detection via CNN and Attention-Based Hierarchical RNN. In Proceedings of the 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), Rotorua, New Zealand, 5–8 August 2019; pp. 112–119. [CrossRef]
48. Mourtaji, Y.; Bouhorma, M.; Alghazzawi, D.; Aldabbagh, G.; Alghamdi, A. Hybrid Rule-Based Solution for Phishing URL Detection Using Convolutional Neural Network. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, e8241104. [CrossRef]
49. Le, H.; Pham, Q.; Sahoo, D.; Hoi, S.C.H. URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection. *arXiv* **2018**, arXiv:1802.03162. [CrossRef]
50. Tajaddodianfar, F.; Stokes, J.W.; Gururajan, A. Texception: A Character/Word-Level Deep Learning Model for Phishing URL Detection. In Proceedings of the ICASSP 2020—2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 2857–2861. [CrossRef]
51. Jiang, J.; Chen, J.; Choo, K.K.R.; Liu, C.; Liu, K.; Yu, M.; Wang, Y. A Deep Learning Based Online Malicious URL and DNS Detection Scheme. In *Security and Privacy in Communication Networks*; Lin, X., Ghorbani, A., Ren, K., Zhu, S., Zhang, A., Eds.;

Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering; Springer: Cham, Switzerland, 2018; pp. 438–448. [CrossRef]
52. Ozcan, A.; Catal, C.; Donmez, E.; Senturk, B. A hybrid DNN–LSTM model for detecting phishing URLs. *Neural Comput. Appl.* **2023**, *35*, 4957–4973. [CrossRef] [PubMed]
53. Tan, C.C.L.; Chiew, K.L.; Yong, K.S.C.; Sebastian, Y.; Than, J.C.M.; Tiong, W.K. Hybrid phishing detection using joint visual and textual identity. *Expert Syst. Appl.* **2023**, *220*, 119723. [CrossRef]
54. Hannousse, A.; Yahiouche, S. Towards benchmark datasets for machine learning based website phishing detection: An experimental study. *Eng. Appl. Artif. Intell.* **2021**, *104*, 104347. [CrossRef]
55. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *Mach. Learn. Python* **2011**, *12*, 2825–2830.
56. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv* **2020**, arXiv:1910.03771. [CrossRef]
57. Timiryasov, I.; Tastet, J.L. Baby Llama: Knowledge distillation from an ensemble of teachers trained on a small dataset with no performance penalty. *arXiv* **2023**, arXiv:2308.02019. [CrossRef]
58. Dakle, P.P.; Rallabandi, S.; Raghavan, P. Understanding BLOOM: An empirical study on diverse NLP tasks. *arXiv* **2023**, arXiv:2211.14865. [CrossRef]
59. Nepal, S.; Gurung, H.; Nepal, R. Phishing URL Detection Using CNN-LSTM and Random Forest Classifier. *Preprint* **2022**. [CrossRef]
60. McConnell, B.; Del Monaco, D.; Zabihimayvan, M.; Abdollahzadeh, F.; Hamada, S. Phishing Attack Detection: An Improved Performance Through Ensemble Learning. In *Artificial Intelligence and Soft Computing*; Lecture Notes in Computer Science; Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M., Eds.; Springer: Cham, Switzerland, 2023; pp. 145–157. [CrossRef]
61. Rashid, S.H.; Abdullah, W.D. Cloud-Based Machine Learning Approach for Accurate Detection of Website Phishing. *Int. J. Intell. Syst. Appl. Eng.* **2023**, *11*, 451–460.
62. Uppalapati, P.J.; Gontla, B.K.; Gundu, P.; Hussain, S.M.; Narasimharo, K. A Machine Learning Approach to Identifying Phishing Websites: A Comparative Study of Classification Models and Ensemble Learning Techniques. *ICST Trans. Scalable Inf. Syst.* **2023**, *10*. [CrossRef]
63. Wang, Y.; Zhu, W.; Xu, H.; Qin, Z.; Ren, K.; Ma, W. A Large-Scale Pretrained Deep Model for Phishing URL Detection. In Proceedings of the ICASSP 2023—2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 4–10 June 2023; pp. 1–5. [CrossRef]
64. Arp, D.; Quiring, E.; Pendlebury, F.; Warnecke, A.; Pierazzi, F.; Wressnegger, C.; Cavallaro, L.; Rieck, K. Dos and Don'ts of Machine Learning in Computer Security. In Proceedings of the 31st USENIX Security Symposium, Boston, MA, USA, 10–12 August 2022.