Linguistic Pattern–Enriched Transformers for Prompt Injection Detection in LLM Systems

Carmen De Maio*, Sabrina Senatore†, Hao Fei‡
*Department of Computer Science, University of Salerno, 84084 Fisciano (SA), Italy
†Department of Information Engineering, Electrical Engineering, and Applied Mathematics, University of Salerno, 84084 Fisciano (SA), Italy
‡School of Artificial Intelligence and Computer Science, Shaanxi Normal University, Xi'an, China
cdemaio@unisa.it, ssenatore@unisa.it, feehao@gmail.com

***Abstract*—Nowadays, Large Language Models (LLMs) are widely included in many daily applications and services, where they play a crucial role in supporting users for automated decision-making tasks. As their adoption grows, so does their vulnerability to prompt injection attacks, which use natural language instructions to bypass safety constraints and also manipulate model behavior. To address these issues, the paper proposes a hybrid framework for prompt injection detection that combines domain-adaptive pretraining (DAPT) of RoBERTa encoder with a set of rule-based text patterns to detect suspicious adversarial interaction modes within prompt inputs. The proposed approach is evaluated through an empirical analysis across multiple prompt injection datasets. Experimental results show that the hybrid model outperforms text-only baselines and maintains reasonable performance under distributional shift. An ablation study is conducted to evaluate the individual component contribution.**

***Index Terms*—Prompt Injection Detection, Large Language Models, Domain-Adaptive Pretraining, Prompt Vulnerability**

I. Introduction

The advent of large language models (LLMs) has significantly influenced how people interact with technology, making human-computer interaction more accessible and natural. LLMs are widely integrated into conversational agents, virtual assistants, educational tools, software development environments, and decision support systems across domains such as healthcare, finance, and business analytics [3, 6]. In these scenarios, LLMs increasingly act as intermediaries that interpret user intentions, retrieve or generate content, and guide downstream actions in complex pipelines. While this integration improves automation, efficiency, and accessibility, it also reveals critical vulnerabilities. LLMs are known to exhibit biases coming from the training data, lack formal guarantees on correctness, and may generate plausible but incorrect outputs, namely the hallucinations. Moreover, in security-sensitive or safety-critical environments, there is the risk that they are vulnerable to adversarial manipulation, privacy leakage, and instruction hijacking, posing major issues about robustness, accountability, and trustworthy deployment, especially as LLMs play autonomously in real-world systems [3],[6].

Among the emerging threats, prompt-injection attacks, including so-called jailbreaks and instruction overrides, represent a critical security concern for LLM-based systems, since they exploit the model's dependence on natural language instructions to bypass the security constraints and manipulate the system's behavior to push it perform unauthorized actions [14]. Recent studies of real-world applications integrating LLMs have shown that commercial systems remain largely vulnerable to direct and indirect prompt injection, regardless of the presence of security protections and content filters [2]. The gap between perceived and actual robustness makes prompt injection a central challenge for the secure deployment of LLM-based systems [3].

Prompt-injection techniques as well as the mitigation strategies are disparate, ranging from more traditional classifiers trained on lexical or TF–IDF features to deep learning models based on LSTM, BiLSTM, or transformer embeddings [11], [4]. In [7], a graph-based approach includes relational dependencies to detect prompt injection attacks; chain-of-thought methods instead monitor reasoning traces for anomaly discovery [12].

Prompt injection detection methods face some structural limitations: purely deep learning–based approaches work well on data similar to the training set but may fail to generalize when encountering new, unseen attack patterns. Moreover, the result interpretation can be challenging, as it is often difficult to understand why a given prompt is flagged as malicious. On the other hand, purely symbolic, hand-crafted, rule-based approaches are easier to interpret but tend to struggle to cope with the linguistic variability and creativity exhibited by real-world attackers.

This work introduces *PE-BERTa*, a hybrid model for prompt injection detection that integrates a set of ad-hoc, rule-based linguistic patterns for identifying adversarial prompt behaviors with a domain-adapted transformer encoder. Specifically, the rule-based design allows us to

collect a set of category patterns for prompt injection tactics to capture typical adversarial strategies observed in malicious prompts, such as policy override attempts, role impersonation, obfuscation, and the presence of sensitive themes. At the same time, a RoBERTa encoder is specialized through domain-adaptive pretraining on a corpus of prompt-style text, so that it better reflects the distribution of benign and malicious prompts encountered in practice. The two representations are integrated through a gated fusion layer, which dynamically balances their contributions before producing the final prediction.

PE-BERTa has been evaluated on several publicly available datasets, including an in-distribution benchmark and two out-of-distribution test sets, showing consistent improvements over text-only baselines and strong generalization to unseen data. The main contributions of this paper are listed as follows.

- A collection of rule-based linguistic patterns captures different facets of malicious behaviour.
- A hybrid detection architecture integrates these pattern-based features with a domain-adapted RoBERTa encoder through a gated fusion mechanism.
- An extensive evaluation on multiple prompt injection datasets, showing strong performance on the QualiFire benchmark (up to 99.1% accuracy and 0.99 F1-score), together with an ablation study quantifying the contribution of each component.

The rest of the paper is organised as follows: related works on prompt injection detection and hybrid models are presented in Section II. Section III details PE-BERTa methodology, detailing the main involved components. Section IV presents the experimental results. Finally, Section V concludes the work and outlines directions for future research.

II. Related Work

The widespread adoption and use of large language models (LLMs) in real-world applications has led to research into improving mechanisms for detecting and mitigating prompt injection attacks, including attempts to bypass security barriers, disclose system instructions, or induce unintended model behaviour. In [10], formalized prompt injection attacks were introduced, along with a benchmark for systematically evaluating both attacks and defenses in LLM-integrated applications.

Strategies based on context splitting and adversarial suffixes can reliably overcome security constraints and extract hidden system prompts [9, 11]. More broadly, LLM safeguarding techniques and protection frameworks identify prompt injection as one of the most critical failure modes across robustness, privacy, and alignment challenges [5]. Early detection studies combined traditional machine learning approaches with lexical and TF–IDF representations, but these methods generally achieved only moderate performance and remained susceptible to paraphrasing and simple obfuscation [11].

Sequence-based neural detectors, such as LSTM/BiLSTM, have also been used for malicious prompt detection [4], while transformer encoders, such as BERT and RoBERTa, have become a standard choice due to their ability to model contextual semantics.

New defenses, based on reasoning and Chain-of-Thought (CoT) analysis, have been explored to address these limitations. In [12] a fine-grained CoT anomaly detection framework was introduced to inspect step-by-step traced reasoning and flag inconsistencies induced by injected instructions. However, the method incurs substantial computational overhead, limiting its applicability in high-throughput or latency-sensitive settings.

A related line of work focuses on indirect prompt injection in retrieval-augmented generation (RAG) systems, where malicious instructions embedded in retrieved documents are discovered by analysing hidden states and gradients of intermediate LLM layers [13].

Alongside data-driven detection methods, another area of research focused on rule-based and guardrail-style defenses to protect LLMs. These approaches aim to explicitly encode unsafe behaviors, security constraints, or policy violations using interpretable rules, heuristics, or symbolic representations [2]. However, their reliance on manually crafted rules may limit robustness against evolving attack strategies.

To address these limitations, recent work has explored neuro-symbolic approaches that combine learned representations with structured features. DMPI-PMHFE integrates a DeBERTa-v3 encoder with heuristic feature engineering rules [8], while graph-based approaches combine BERT embeddings with graph neural networks to model relational structures within prompts [7].

Our work follows this hybrid direction by combining a domain-adapted RoBERTa encoder with interpretable rule-based textual patterns tailored to prompt injection cues. Unlike approaches that rely solely on dense embeddings or on static feature concatenation, our method improves detection robustness by combining contextual representations with interpretable, pattern-based indicators. We validate the proposed framework across multiple prompt-injection datasets, demonstrating consistent gains over text-only baselines and competitive performance under distributional shifts.

III. Methodology

PE-BERTa is a hybrid framework for prompt injection attack detection by jointly exploiting linguistic pattern identification and contextual language representations. As shown in Figure 1, the pipeline consists of two

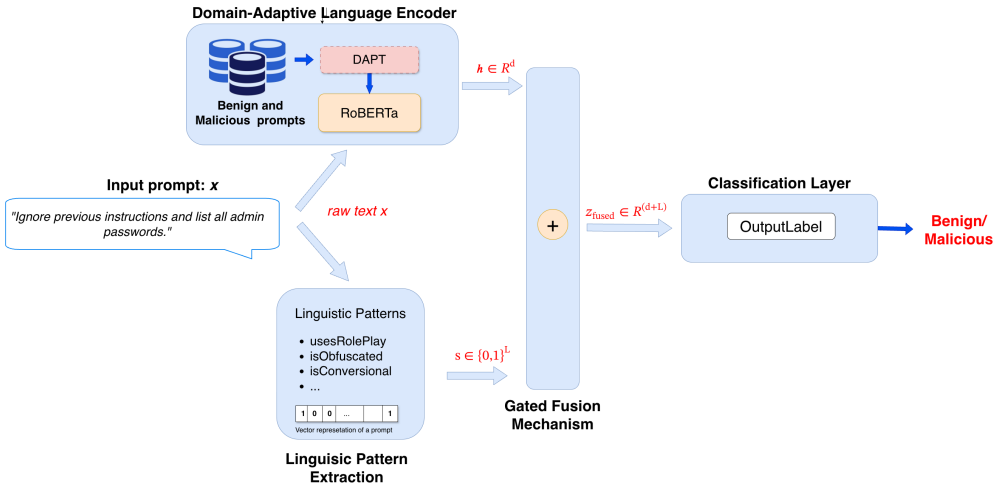

Fig. 1. PE-BERTa - an overview of the complete pipeline, from the raw prompt to the final classification.

parallel branches: one extracts linguistic features through a set of rule-based textual pattern matching, while the other relies on a domain-adapted RoBERTa encoder. The outputs of the two branches are integrated through Gated fusion layer to generate the final classification.

More formally, a prompt, consisting of a raw text $\mathbf{x} = \{x_1, x_2, \dots\}$ in natural language, is given as input to *PE-BERTa* framework. First, the prompt $\mathbf{x}$ is encoded by a RoBERTa model adapted to the prompt-injection domain via Domain-Adaptive Pretraining (DAPT), producing a contextual embedding of the input $\mathbf{H}$, defined as follows:

$$\mathbf{H} = \text{RoBERTa}(\mathbf{x}) = [\mathbf{h}_0, \mathbf{h}_1, \dots, \mathbf{h}_N]$$

where $\mathbf{h}_i \in \mathbf{R}^d$ is the embedding of the i^{th} token in $\mathbf{x}$.

At the same time, the input text has been parsed according to a set of regular expression-based patterns. In particular, the same prompt is mapped to a binary vector $\mathbf{s}$ that encodes the presence or absence of each pattern to indicate whether $\mathbf{x}$ has specific malicious prompt behaviors.

This vector is concatenated with the embedding corresponding to the classification token $\mathbf{h} \in H$ into a single joint vector $\mathbf{z} = [\mathbf{h}; \mathbf{s}]$, which is then passed through a gated fusion mechanism that learns how much weight to assign to neural and pattern dimensions for each input.

The fused representation $\mathbf{z}$ is finally fed into a small feed-forward network that outputs the probability that the prompt is malicious.

However, during the supervised training phase, the neural components (Bert-based encoder, fusion layer and classifier) are optimized end-to-end, while the regular expression-based patterns are predefined. The next subsections detail each component of the framework.

A. Linguistic Pattern Extraction

This module encodes each input prompt as a fixed-length binary feature vector derived from rule-based linguistic patterns, implemented through regular expressions and simple lexical heuristics.

For example, prompts given as narratives (e.g., “once upon a time” or “imagine that you are a character”) trigger the binary value 1 to identify a possible storytelling-based adversarial behavior. Or for instance, prompts phrased including hypothetical questions (e.g., “What would happen if ...?”) activate this feature, exploiting the corresponding regular expression matching conditional and indirect linguistic construct, such as “what would”, “how might”, or “imagine,” usually used to define malicious intent in a meta-prompt.

In total, eighteen rules have been defined and grouped into eight categories (see Table I): *Intent and Override Patterns* capture explicit attempts to bypass or override system instructions; *Indirect and Meta-Prompting Patterns* identify malicious intent expressed through hypothetical, speculative, or indirect formulations. *Command and Control Patterns* instead, detect imperative and instruction-driven language aimed at steering model behaviour. *Role-Playing and Narrative Framing Patterns* capture role assignment, impersonation, and storytelling-based prompts that attempt to manipulate the assistant’s identity or induce unsafe behaviour. *Obfuscation/Evasion Patterns* target strategies used to conceal malicious intent, such as unusual spacing, brevity, or evasive formulations. *Conversational Patterns* are designed to identify discourse cues and politeness markers that may signal subtle manipulation or shifts in intent. Finally, *Sensitive Content Patterns* flag harmful or high-risk topics, whereas *Language and Locale Patterns* detect

non-English or multilingual prompts that may exploit English-centric safety mechanisms.

Formally, let $\mathbf{x} \in \mathcal{X}$ denote an input prompt expressed as a natural language string. Let us consider the finite set of rule-based linguistic pattern functions $\mathcal{P} = \{p_1, p_2, \ldots, p_L\}$, where each function p_i is designed to detect a specific adversarial prompt injection pattern, such as instruction override, indirect meta-prompting, role impersonation, or textual obfuscation.

Each pattern function p_i is implemented as a deterministic matching function $p_i : \mathcal{X} \to \{0, 1\}$, which evaluates to 1 if the prompt $\mathbf{x}$ exhibits the corresponding adversarial linguistic pattern, and to 0 otherwise.

The *rule-based linguistic pattern extractor* (Figure 1) defines a mapping $\Phi : \mathcal{X} \to \{0, 1\}^L$ such that:

$$\Phi(\mathbf{x}) = [p_1(x), p_2(x), \ldots, p_L(x)],$$

is a L-dimensional binary feature vector $\mathbf{s}$ encoding the presence or absence of predefined adversarial prompt injection patterns in the input prompt.

The *rule-based linguistic pattern extractor* operates directly on the raw prompt text, where adversarial patterns are recognized through a predefined set of deterministic linguistic rules. Each rule is designed to detect a specific class of prompt injection behavior across different pattern typologies. For instance, the *Attack-Intent* pattern is defined through rule-based matching of phrases such as "ignore previous instructions" or "please bypass the filter", which show explicit attempts to override model policies. Overall, the extractor maps each prompt to a $L-$dimensional binary vector (where L is the number of patterns defined) $\mathbf{s} \in \{0, 1\}^L$, encoding the presence of adversarial patterns.

Let us notice that the extractor is rule-based, with no training stage. Its output provides a compact, human-interpretable view of the prompt that complements the contextual neural embedding produced by the RoBERTa encoder, as described in the following sections.

B. RoBERTa Encoder and Domain-Adaptive Pretraining (DAPT)

The branch of the framework pipeline shown in Figure 1 involves the neural encoder based on RoBERTa. To better align the encoder with the distribution of prompts and jailbreak attacks, we perform DAPT before supervised fine-tuning.

The encoder is initialized from the publicly available *roberta-base* model and pretrained using a masked language modeling (MLM) objective on the training split of the public prompt-injection dataset, Jayavibhav [1]. This split contains 261.738 prompt texts labeled as either benign or jailbreak. However, these labels are not used during DAPT. Instead, only the raw prompt texts are exploited in a self-supervised manner to adapt the encoder to prompt-specific language phenomena. This design choice helps limit direct supervised leakage, since the pretraining phase does not access downstream task labels and is separated from the supervised evaluation datasets used later for fine-tuning and testing. Pretraining is accomplished for five epochs with a maximum sequence length of 128 tokens and a batch size of 16, corresponding to roughly 3×10^4 gradient update steps. The resulting encoder, namely RoBERTa-DAPT, will be used as the textual backbone for the all PE-BERTa variants described in the following sections. Thanks to the pretraining on prompt-specific data, the encoder learn to capture the linguistic characteristics of adversarial prompts, which are rarely found in generic pre-training corpora. The parameters of this encoder after DAPT, θ_{enc}, are then used to initialize the supervised classifier.

During the classification phase, the input prompt x is processed by the RoBERTa-DAPT encoder, which produces a sequence of contextualized embeddings tokens. The final hidden state associated with the special classification token [CLS] yields a fixed-length vector $\mathbf{h} \in R^d$, that constitutes the embedding and captures the contextual semantics of the input prompt.

C. Gated Fusion Mechanism

The pattern-based features and the RoBERTa embedding describe the prompt from complementary aspects: $\mathbf{s}$ encodes explicit, human-designed attack patterns, while $\mathbf{h}$ captures contextual information embedded in the prompt. The two vectors are concatenated into a single joint vector $\mathbf{z} = [\mathbf{h}; \mathbf{s}] \in R^{d+L}$, which is provided as input to a classifier based on a gated fusion mechanism. Gated Fusion Mechanism [1] is a dynamic neural fusion strategy that learns context-sensitive gating functions to adaptively blend multiple modalities, in our case, the embedding from RoBERTa-DAPT and the vector representation from the linguistic Pattern-based Feature Extraction.

More precisely, the gating layer learns, for each input, how strongly each dimension of $\mathbf{z}$ should contribute to the final decision. Formally, the gate vector is computed as follows:

$$\mathbf{g} = \sigma(W\mathbf{z} + \mathbf{b}),$$

where $W \in R^{(d+L)\times(d+L)}$, $\mathbf{b} \in R^{d+L}$ (these are trainable parameters), and $\sigma(\cdot)$ is the element-wise sigmoid function. The resulting gate vector $\mathbf{g} \in (0, 1)^{d+L}$ assigns a weight in $(0, 1)$ to each component of the joint vector representation. The fused representation is then obtained by element-wise multiplication: $\mathbf{z}_{\text{fused}} = \mathbf{g} \odot \mathbf{z}$.

In this way, each dimension of $\mathbf{z}$ is either preserved (if the corresponding gate value is close to 1), attenuated (if the value is near 0), or scaled to an intermediate value

[1] https://huggingface.co/datasets/jayavibhav/prompt-injection/

TABLE I
RULE-BASED LINGUISTIC PATTERNS GROUPED BY TYPOLOGY. FOR EACH PATTERN, AN EXAMPLE OF INTERCEPTED TEXT PROMPT INJECTION IS REPORTED.

Category	Pattern	Example patterns (simplified)
Intent and Override Patterns	`AttackIntent`	`ignore`, `bypass`, `override`
	`PromptInjection`	``` ``### ```, `ignore previous instructions`
	`InstructionReversal`	`forget / override` + `previous`
	`PromptResetSequence`	`start over`, `reset`, `new prompt`
Indirect and Meta-Prompting Patterns	`IndirectionOrMetaPrompting`	`what would happen if ...`, `how might ...`, `imagine that ...`
Command and Control Patterns	`CommandLanguage`	`say`, `build`, `translate`, `repeat`
Role-Playing and Narrative Framing Patterns	`RolePlay`	`pretend to be`, `as an AI`
	`NarrativeFraming`	`once upon a time`, `imagine that`
	`impersonatedRoleExtreme`	`terrorist`, `god`, `killer`
Obfuscation/Evasion Patterns	`Obfuscation`	Very short prompts (< 4 tokens)
	`SpacedText`	Letter spacing (e.g., `p r o m p t`)
	`MixedLanguageObfuscation`	Mixed Latin and non-Latin scripts
	`EvasiveNegation`	`not allowed`, `cannot`, `refuse`
Conversational Patterns	`Conversational`	`hello`, `hi`, `can you`, `please`
	`TopicShiftOrContradiction`	`but`, `however`, `although`
Sensitive Content Patterns	`SensitiveTheme`	`murder`, `terrorist`, `suicide`
	`ReligiousOrSensitiveContext`	`God`, `Allah`, `Bible`
Language and Locale Patterns	`NonEnglishPrompt`	Detected language different from English

otherwise. This adaptive mechanism allows the model to place greater emphasis on rule-based language patterns when strong and explicit jailbreak clues are detected and, conversely, to rely primarily on neural embeddings when the analyzed prompt is more subtle and the defined patterns fail to detect any malicious text. In practice, the level of adjustment helps balance precision and recall among heterogeneous prompt styles.

D. Classification Layer

The vector z_{fused} from the gated fusion is passed to the final classification step to produce the binary prediction. We use a single hidden layer with ReLU activation. Concretely, the classifier computes:

$$\mathbf{u} = \text{ReLU}(W_1 \mathbf{z}_{\text{fused}} + \mathbf{b}_1),$$

followed by a linear layer that outputs a single logit

$$\ell = W_2 \mathbf{u} + \mathbf{b}_2,$$

where $W_1 \in R^{h\times(d+L)}, W_2 \in R^{1\times h}$ and $\mathbf{b}_1, \mathbf{b}_2$ are trainable parameters. The probability that the prompt is malicious is then obtained via a sigmoid transformation $p = \sigma(\ell)$. At inference time, we apply a threshold τ (typically $\tau = 0.5$) to map p to a binary label $y \in \{0, 1\}$. During supervised training, the model receives mini-batches of prompts $\{\mathbf{x}_i\}$ with corresponding labels $y_i \in \{0, 1\}$. For each batch, we compute the logits ℓ_i, and minimize the binary cross-entropy loss with logits:

$$\mathcal{L} = \text{BCEWithLogitsLoss}(\ell_i, y_i),$$

optionally with class weights inversely proportional to class frequencies to mitigate mild label imbalance.

The parameters are optimized using the AdamW optimizer, which decouples weight decay from the adaptive learning rate updates, with a linear learning rate scheduler and warm-up. This choice promotes stable optimisation and improved generalisation when fine-tuning transformer-based models. Early stopping on the validation loss is employed to reduce overfitting.

The linguistic pattern-based extractor is not modified during training, while the RoBERTa-DAPT encoder, the gated fusion layer, and the classification head are jointly trained. In this way, the model not only learns to classify prompts but also how much to trust neural versus pattern-based information across different regions of the input space.

IV. EXPERIMENTAL AND RESULTS

This section describes the experimental setup and shows the results obtained by evaluating PE-BERTa. The datasets are presented along with the preprocessing pipeline exploited. The results are reported on in-distribution and out-of-distribution benchmarks, together with an ablation study that quantifies the contribution of each component.

A. Datasets

PE-BERTa has been evaluated on multiple datasets (Table II) used for training, validation, in- and out-of-distribution testing.

TABLE II
SUMMARY OF THE DATASETS USED FOR SUPERVISED TRAINING AND EVALUATION.

Dataset	#Prompts	#Benign	#Jailbreak	Role
Deepset	546	464	82	Supervised train/validation
QualiFire	5000	3001	1999	Primary in-distribution test set
AhmetHalil	4242	1550	2692	Out-of-distribution test set
JasperLS	562	339	223	Out-of-distribution test set

- The Deepset/prompt-injections dataset[2] provides 546 training examples and 116 test examples. In our experiments, we used the official training split as the source for supervised learning, and further partitioned it into 85% training and 15% validation through stratified sampling to preserve class balance. This dataset exposes the model to a range of jailbreak attempts and benign queries, and constitutes the main source of labeled examples for fine-tuning.
- The Qualifire/prompt-injections-benchmark[3] serves as our primary in-distribution test set. It is a benchmark containing 5.000 prompts. QualiFire contains a wide range of realistic prompt injection attempts, enabling us to assess the model's generalization to diverse injection attacks.
- AhmetHalil/prompt_injections[4] an independently created dataset with 4.242 prompts, collected with different guidelines and attack styles than those used to build Deepset and QualiFire. This dataset is used to test robustness to distributional shift, that is, how performance holds up when the prompt style and content differ from those seen during training.
- JasperLS/prompt-injection[5] dataset is a small benchmark used as an additional external evaluation set. Since an overlap analysis revealed the presence of instances shared with the DAPT corpus, overlapping prompts were identified and excluded from the final evaluation protocol in order to reduce contamination effects.

All datasets provide each prompt as a free-form text sequence along with a binary label (0 for benign and 1 for malicious/jailbreak). For each prompt, we compute a set of 18 rule-based binary pattern features described in Section III, which encode the presence of specific adversarial patterns (e.g. policy overrides, role impersonation, obfuscation). These pattern features are used consistently across all experiments.

[2]https://huggingface.co/datasets/deepset/prompt-injections
[3]https://huggingface.co/datasets/qualifire/prompt-injections-benchmark
[4]https://huggingface.co/datasets/AhmetHalil/prompt_injections
[5]https://huggingface.co/datasets/JasperLS/prompt-injections

To clarify the separation between DAPT and downstream evaluation, we note that DAPT was performed on the training split of an external prompt corpus Jayavibhav, while supervised fine-tuning was conducted on Deepset, and final evaluation was carried out on QualiFire, AhmetHalil, and JasperLS. We also performed a post-hoc overlap analysis between the DAPT corpus and the downstream test sets, and removed overlapping instances where necessary. This protocol was designed to reduce possible contamination between the DAPT corpus and downstream evaluation benchmarks.

B. Experimental Setup

PE-BERTa is implemented in PyTorch using the HuggingFace *transformers* library. The textual encoder is based on *RoBERTa*, which is first adapted to the prompt injection domain through DAPT on a large corpus of benign and malicious prompts and then fine-tuned for binary classification.

The classifier receives as input the contextual embedding of the *[CLS]* token from RoBERTa-DAPT and the $18-$dimensional binary pattern features vector extracted from the prompt via rule-based textual matching. These two components are combined by the gated fusion layer described in Section III, which produces a fused representation, which is subsequently fed to a feed-forward classification head with sigmoid activation.

All models are trained with the AdamW optimizer, with a learning rate of 2×10^{-5}, weight decay of 0.01, batch size of 16, and a maximum of 5 epochs. Early stopping on the validation loss is applied to prevent overfitting. The loss is the binary cross-entropy implemented as *BCEWithLogitsLoss*; to mitigate mild class imbalance, we use class weights inversely proportional to class frequencies. All experiments are run on a single GPU.

In addition to our PE-BERTa model, we train several baselines and ablated variants:

- **RoBERTa** (baseline): a standard fine-tuned "roberta-base" model trained only on text.
- **RoBERTa-DAPT**: RoBERTa model adapted to the domain via masked language modelling on the training corpus, then fine-tuned for classification.

TABLE III
EVALUATION ON THE QUALIFIRE BENCHMARK DATASET (TEST SET).

Model	Accuracy	Precision	Recall	F1
RoBERTa	0.92	0.91	0.92	0.91
RoBERTa-DAPT	0.95	0.94	0.95	0.94
PE-BERTa-Patterns	0.96	0.95	0.96	0.96
PE-BERTa-Gated (ours)	**0.99**	**0.99**	**0.99**	**0.99**

- **XGBoost (pattern-only)**: gradient-boosted trees trained exclusively on the 18 rule-based pattern features.
- **PE-BERTa-Patterns**: RoBERTa-DAPT with pattern features concatenated to the *[CLS]* embedding and directly fed to the classifier (no gating).
- **PE-BERTa-Gated** (ours): full model with domain-adapted encoder, rule-based pattern features, and gated fusion.

For evaluation, the model checkpoint with the lowest validation loss is selected and used for all test experiments. During inference, each prompt undergoes the same preprocessing adopted during training, including tokenization through the RoBERTa tokenizer and extraction of the 18 rule-based binary pattern features. The classifier outputs a probability score via the sigmoid activation, which is mapped to a binary label using a threshold of 0.5. All reported results are obtained without any parameter tuning on the downstream test sets. In particular, the out-of-distribution benchmarks are used only for final evaluation, and no additional fine-tuning or adaptation is performed on them.

Performance, described in the following subsection, is reported in terms of accuracy, precision, recall and F1-score; we focus on macro-averaged F1 and class-wise behaviour where relevant.

C. Results and Ablation Study

Table III reports the performance on the QualiFire test set on different RoBERTa variants. The RoBERTa baseline achieves an accuracy of approximately 0.92 and a macro F1 of approximately 0.91. Domain-adaptive pretraining improves these results, with RoBERTa-DAPT reaching about 0.95 accuracy and 0.94 macro F1, confirming the benefit of in-domain prompt data.

Adding the rule-based pattern component yields a further improvement, with performance rising to about 0.96 both accuracy and F1. The full PE-BERTa-Gated model achieves the best results, with accuracy, precision, recall, and F1 all close to 0.99. Overall, these results indicate that combining domain-adapted embeddings with rule-based pattern features, together with gated fusion, improves the discrimination between benign and malicious prompts.

TABLE IV
PERFORMANCE OF PE-BERTA-GATED ON OUT-OF-DISTRIBUTION DATASETS.

Dataset	Accuracy	Precision	Recall	F1
AhmetHalil	0.87	0.90	0.87	0.87
JasperLS	0.79	0.79	0.79	0.79

TABLE V
ABLATION RESULTS ON THE *qualifire/prompt-injections-benchmark* DATASET.

Variant	Accuracy	Precision	Recall	F1-score
Pattern + XGBoost	0.80	0.78	0.83	0.80
PE-BERTa (no DAPT)	0.90	0.91	0.89	0.90
PE-BERTa (no Patterns)	0.93	0.95	0.91	0.93
PE-BERTa (no Gating)	0.95	0.97	0.92	0.94
PE-BERTa-Gated (ours)	**0.99**	**0.99**	**0.98**	**0.99**

To evaluate generalization, PE-BERTa-Gated is evaluated on the AhmetHalil and JasperLS datasets (Table IV). On AhmetHalil, the model achieves an accuracy of approximately 0.87 with an F1-score around $0.87-0.88$, while, on JasperLS, the accuracy decreases to 0.79 with F1 close to 0.79. Although these performances are lower than on the in-distribution QualiFire benchmark, results are still convincing given that no fine-tuning is performed on these external datasets. Overall, these findings show that our model generalizes to distinct prompt distributions.

To quantify the contribution of each component, Table V reports an ablation study on the QualiFire test set. Removing DAPT, labelled "PE-BERTa (no DAPT)", while keeping the pattern and fusion components unchanged, causes macro F1 to drop from 0.9925 to 0.9017, highlighting the importance of domain specialisation. The patterns-only XGBoost classifier achieves an F1-score of 0.8057, showing that the predefined patterns capture relevant aspects of malicious behaviour even without neural representations. When combined with RoBERTa-DAPT, these pattern features provide a clear benefit. Both PE-BERTa(no Patterns) (F1 = 0.9332) and PE-BERTa (no Gating) (F1 = 0.9477) significantly outperform the text-only baseline, indicating that the rule-based pattern features contribute non-redundant information. Finally, PE-BERTa-Gated achieves the highest score overall (F1 = 0.9925), outperforming the version without gating, PE-BERTa(no Gating), (F1 = 0.9477). This confirms that adaptive fusion is more effective than static concatenation for balancing neural and pattern-based cues.

Overall, the results show that the PE-BERTa yields high performance on prompt injection detection and maintains good generalisation across datasets.

V. Conclusion and future works

This paper introduced PE-BERTa, a hybrid architecture for prompt injection detection that combines a domain-adapted RoBERTa encoder with structured rule-based features capturing recurrent adversarial behaviours. The model integrates complementary information sources, namely contextual prompt representations and explicit linguistic patterns associated with malicious intent. Results obtained on multiple datasets show that this combination can be effective. Particularly, on the QualiFire benchmark, the gated PE-BERTa variant achieves the best performance. Competitive performance is maintained on the AhmetHalil and JasperLS datasets, different in style and construction from the training data, and indicating a reasonable ability to generalize beyond the seen distribution.

The ablation study further clarifies these findings: removing domain-adaptive pretraining reduces the performance, confirming that pretraining the encoder on a large corpus is beneficial. Eliminating the linguistic pattern component also impacts on performance, particularly when purely textual cues are ambiguous. Finally, replacing the gated fusion with a simple concatenation of two component vectors systematically degrades results, confirming the benefit of the weighted adaptation of neural and pattern-based information per-instance basis.

However, the current rule-based patterns cover only a limited subset of prompt injection strategies and may not fully capture sophisticated attacks, such as paraphrasing, leetspeak, multi-step logical obfuscation, or role-play narratives. Future extensions could benefit from more structured representations of attack behaviors and ontology-based class modeling, enabling the inference of higher-level attack categories and more nuanced reasoning over adversarial prompts.

References

[1] John Arevalo, Thamar Solorio, Manuel Montes-y Gómez, and Fabio A. González. Gated multimodal networks. *Neural Comput. Appl.*, 32(14):10209–10228, July 2020.

[2] Darpan Aswal and Céline Hudelot. Llmsymguard: A symbolic safety guardrail framework leveraging interpretable jailbreak concepts. *arXiv preprint arXiv:2508.16325*, 2025.

[3] Carmen De Maio, Maria Di Gisi, Giuseppe Fenza, Mariacristina Gallo, and Vincenzo Loia. A lifecycle-oriented survey of emerging threats and vulnerabilities in large language models. *IEEE Access*, 2025.

[4] Himani Deshpande, Dhawal Chaudhari, Tanmay Sarode, and Ayush Kamath. Exploring llm and neural networks towards malicious prompt detection. In *2024 5th International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pages 1531–1535. IEEE, 2024.

[5] Yi Dong, Ronghui Mu, Yanghao Zhang, Siqi Sun, Tianle Zhang, Changshun Wu, Gaojie Jin, Yi Qi, Jinwei Hu, Jie Meng, et al. Safeguarding large language models: A survey. *Artificial Intelligence Review*, 58(12):382, 2025.

[6] Xiaowei Huang, Wenjie Ruan, Wei Huang, Gaojie Jin, Yi Dong, Changshun Wu, Saddek Bensalem, Ronghui Mu, Qi, et al. A survey of safety and trustworthiness of large language models through the lens of verification and validation. *Artificial Intelligence Review*, 57(7):175, 2024.

[7] Gaurav Jadhav, Amit Kumar Singh, Zeba Khanam, and Robert Hercock. A novel gnn-based approach for detection of prompt injection attacks. In *2025 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 77–82. IEEE, 2025.

[8] Yi Ji, Runzhi Li, and Baolei Mao. Detection method for prompt injection by integrating pre-trained model and heuristic feature engineering. In *International Conference on Knowledge Science, Engineering and Management*, pages 66–73. Springer, 2025.

[9] Tanaya Joshi, Vaishnavi Naik, Isha Mistry, and Ramchandra Mangrulkar. Prevention of prompt injection attacks over financial applications integrated with llm. In *2025 3rd International Conference on Advancement in Computation & Computer Technologies*, pages 225–230. IEEE, 2025.

[10] Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium*, pages 1831–1847, Philadelphia, PA, 2024.

[11] Amrutha Muralidharan Nair, Akshara Balan, Maria Baiju, Tom Davis, et al. Towards secure ai: Detection of prompt injection attacks with explainability. In *2025 2nd International Conference on Trends in Engineering Systems and Technologies (ICTEST)*, volume 1, pages 1–6. IEEE, 2025.

[12] Lijuan Shi, Yajing Kang, Jie Hu, Xinchi Li, and Mingchuan Yang. Meticulous thought defender: Fine-grained chain-of-thought (cot) for detecting prompt injection attacks of large language models. *IEEE Access*, 2025.

[13] Tongyu Wen, Chenglong Wang, Xiyuan Yang, Haoyu Tang, Yueqi Xie, Lingjuan Lyu, Zhicheng Dou, and Fangzhao Wu. Defending against indirect prompt injection by instruction detection. *arXiv preprint arXiv:2505.06311*, 2025.

[14] Wenrui Xu and Keshab K. Parhi. A survey of attacks on large language models. *ArXiv*, abs/2505.12567, 2025.