Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

IntentShield: A Semantic Framework for Detecting Obfuscated Jailbreak Attacks in Domain-Specific Large Language Models

Jay Bankar

bankarjay2304@gmail.com

Sanjivani University https://orcid.org/0009-0008-1199-6807

Research Article

Keywords: Prompt injection, jailbreak detection, malicious prompt detection, DeBERTa, robustness to adversarial examples, obfuscation,LLM-security.

Posted Date: November 27th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11879471/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

IntentShield: A Semantic Framework for Detecting Obfuscated Jailbreak Attacks in Domain-Specific Large Language Models

Jay Bankar
Artificial Intelligence Data Science
Sanjivani University
Ahmednagar, Maharashtra, India
bankarjay2304@gmail.com

***Abstract*—Large Language Models (LLMs) are currently employed in several applications such as healthcare, finance, education, and law firms. The increasing application of LLMs raises concerns regard- ing security issues including jailbreak and prompt injection that involves malicious instructions embedded in seemingly harmless prompts. The attackers have the option to paraphrase their instructions, use role-playing scenarios, switch languages, tokenize the prompt, and encode different portions of the input. Such variations can render traditional safety filters ineffective. Keyword detection, hardcoded policies, and prior attack data could fail to detect any different iteration of the same malicious instructions. This paper introduces IntentShield which utilizes a semantic approach to detect obfuscated jailbreak attacks within domain-specific Large Language Models. IntentShield does not rely on just plain text but also transformer representations to detect the intentions of the inputs.**

From a number of practical perspectives, the effectiveness of the approach is considered: detection capability, behavior in professional domains, false positives on legitimate prompts, resilience to unseen obfuscation techniques, and computational costs. From the test results, it is clear that the approach demonstrates high effectiveness in the domain-balanced test with frozen data. Nevertheless, the experiment of unseen obfuscation shows an essential vulnerability of the approach under some specific encoding technique of malicious intent.

***Index Terms*—prompt injection, jailbreak detection, malicious prompt detection, DeBERTa, robustness to adversarial examples, obfuscation,LLM-security.**

I. Introduction

As Large Language Models start being used in applications where user input is allowed to provide the instruction directly, the prompt becomes an inseparable part of the application security perimeter. Prompt injection is identified by OWASP as one of the primary risks associated with LLM-based applications since model-instructions can affect system operation despite any attacks being unnoticed by humans. [1].

Jailbreaks present another challenge. Well-designed inputs have the ability to steer the model away from safe behavior [2]. This problem is not confined to just one single model; previous works on transferable suffix-based attacks reveal that certain attacks are generalizable across various families of models [3].

This creates a difficult task for detectors. Good accuracy on familiar test sets is useful but not enough. Security detectors should also be checked for data leakage, false positives, false negatives, domain changes, and changes in how attacks are represented. HarmBench and JailbreakBench have emphasized reproducible and standardized security evaluation [4], [5].

IntentShield follows this evaluation approach, treating malicious-prompt detection as a standalone binary classification problem. A prompt is either benign or malicious. The detector makes this decision before the prompt reaches the protected LLM.

A. Contributions

- A leakage-controlled training and evaluation process based on seed groups and frozen data partitions.
- A semantic detector based on DeBERTa-v3-base, with the decision threshold selected using validation data.
- A balanced cross-domain evaluation containing 1,600 prompts, together with domain-level error analysis.
- A frozen unseen obfuscation experiment comprising 574 malware mutations.
- Presentation of the confusion matrix, threshold characteristics, latency, throughput, assessment formulas, and examples of mistakes that models make.

B. Research Question

In this research paper, the following question is raised: Will a transformer-based detector for semantics maintain its capability to classify malicious prompts accurately in different professional settings despite changes in the representation of prompts unknown before?

II. Related Work

A. Prompt Injection and Jailbreak Detection

The technique of prompt injection exploits the instruction hierarchy and context visible to the model. Jailbreaking is very similar, with attacks targeted at circumventing model safety behaviors.

Wei et al. provide two reasons why jailbreaking could prevent LLM safety training: conflicting objectives and the generalization gap [2]. According to Zou et al., the adversarial

suffixes produced by an automatic process can be portable across models [3].

These results suggest problems with approaches based on fixed lexica. It is not necessary to use pre-defined malicious phrases since wording could change but goals would stay the same. Thus, semantics of inputs should be used to determine if they are malicious.

B. Robustness and Obfuscation

Jailbreak detection requires robustness. HarmBench is a benchmark for robust-red teaming and refusal evaluation [4]. JailbreakBench is a reproducible benchmark and threat model for jailbreak research [5].

Previous work emphasizes the importance of the way instructions are represented. SmoothLLM shows how jailbreaks can be affected by changing representations at the level of characters [6]. Cipher-character attacks reveal how different representations may influence moderation mechanisms [7].

IntentShield not only analyses the regular prompt text. There is an additional step of obfuscation which checks if the detector recognizes malicious intent despite the modification of the representation.

C. Detector Baselines

The original experimental design included a lexical keyword filter, TF-IDF using Logistic Regression, and a lightweight transformer model, such as DistilBERT. Reference models such as Meta Prompt Guard 2 and ProtectAI DeBERTa were identified as additional benchmarks in the revised research draft.

However, there are no numeric data available for the performance of these benchmarks on the frozen test dataset. Including comparison metrics based on experiments that have not been conducted would lead to unmetered results. The current paper describes the methodological relevance of these benchmarks without specifying any performance scores.

It is crucial to conduct these experiments on the frozen test dataset.

III. Threat Model and Scope

The threat model assumes attackers can control user-facing prompts but cannot modify IntentShield detector parameters.

Several attack forms are considered. Attackers may directly ask models to ignore or replace instructions. Objectives may also be hidden through contextual framing, role playing, or indirect instructions. Representation-level transformations include Base64, hexadecimal, ROT13, zero-width characters, punctuation changes, reversed text, and Morse encoding.

IntentShield works as an input-side preprocessing layer, examining prompts before protected LLMs process them.

The scope is intentionally limited. The detector does not address compromised model weights, training-data poisoning, malicious retrieval documents, compromised tool outputs, authorization failures, or attacks introduced after detector decisions.

IntentShield should be viewed as one security layer, not a complete LLM security system.

IV. Proposed Methodology

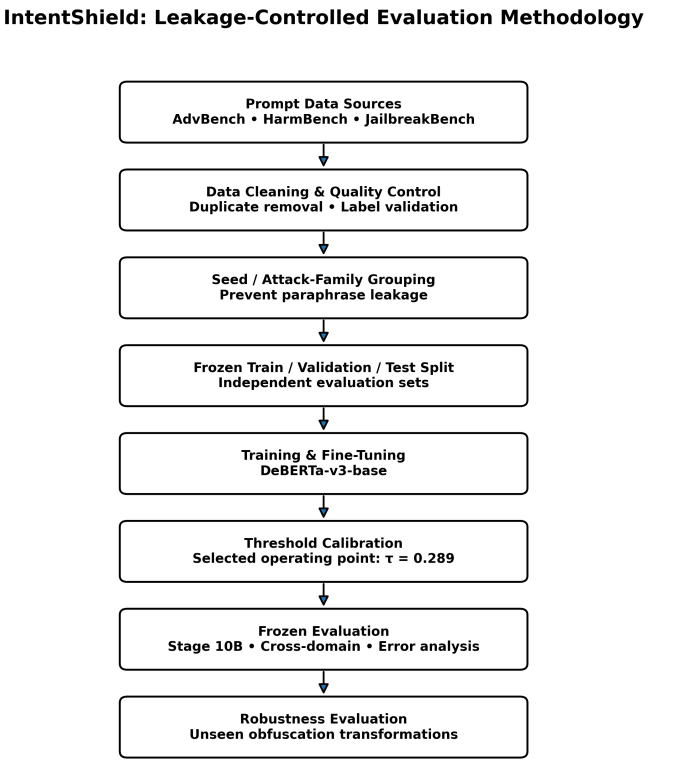


Fig. 1. Reproducible IntentShield methodology and evaluation flow.

The workflow begins with labeled prompts collected for model development and evaluation. Before training, data is standardized and important metadata such as source and seed group is retained. Related prompt variants are separated before training, reducing chances that nearly identical attack forms appear across training and evaluation partitions.

The classifier is fine-tuned using the frozen training partition.

Threshold selection is handled separately. Validation data chooses the operating threshold, while Stage 10B is kept outside this process. Once selected, both detector and decision rule are frozen.

The frozen detector is then used for Stage 10B cross-domain evaluation, followed by testing with previously unseen obfuscation transformations.

V. Model Formulation and Architecture

A. Binary Classification

Let x represent an input prompt and $y \in \{0, 1\}$ represent its security label, where 0 is benign and 1 is malicious.

The encoder converts tokenized inputs into contextual representations h. A classification layer produces the malicious probability p:

$$p = f_\theta(x) = \sigma(Wh + b), \quad \sigma(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

The final prediction uses the calibrated threshold:

$$\hat{y} = \mathbf{1}[p \geq \tau], \quad \tau = 0.289 \tag{2}$$

The binary classification loss is:

$$L(\theta) = -\big[y \log p + (1-y)\log(1-p)\big] \tag{3}$$

The final detector uses microsoft/deberta-v3-base with maximum sequence length 256, trained for three epochs. Learning rate is 1×10^{-5}, with 0.1 warmup ratio, 0.01 weight decay, 0.1 label smoothing, and 0.15 dropout.

Training batch size is 8 with gradient accumulation of 2, evaluation batch size is 16. Checkpoint selection is based on validation loss. Random seed is fixed at 42.

B. Dataset Construction and Reproducibility

The data audit identifies alpaca_processed.csv with 51,760 rows, together with derived final_dataset.csv and final_dataset_clean.csv artifacts. The selected training corpus is final_dataset_augmented.csv.

The merged training artifact contains 89,165 rows: 88,938 original rows and 270 targeted augmentation rows.

Targeted augmentation contains:

- 120 Targeted Direct Override examples
- 120 Security Hard Negative examples
- 30 Novel Adversarial examples

The frozen conference split contains 11,862 training rows, 1,496 validation rows, and 1,480 test rows.

The audit found zero evaluation-text overlap, zero evaluation-ID overlap, and zero exact duplicate text/ID rows in the merged training artifact.

Stage 10B is a separate frozen domain-balanced evaluation set containing 1,600 unique prompts.

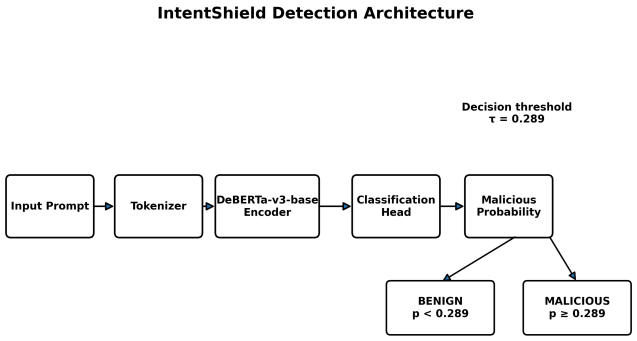


Fig. 2. IntentShield inference architecture.

C. Evaluation Composition

The malicious part of Stage 10B is evenly organized into four attack categories: 200 Contextual, 200 Direct, 200 Indirect, and 200 RolePlay prompts.

The 150 novel adversarial prompts are analyzed separately from the original 1,450 prompts for independent error examination.

TABLE I
EVALUATION COMPOSITION

Set	Samples	Composition
Stage 10B	1,600	800 benign + 800 malicious; 400/domain
Original Stage 10B	1,450	650 malicious + 800 benign
Novel Stage 10B	150	150 malicious; 2 missed
Stage 13	574	82 seeds × 7 unseen transformations

VI. EVALUATION METRICS AND THRESHOLD CALIBRATION

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

$$\text{FPR} = \frac{FP}{FP + TN}, \quad \text{FNR} = \frac{FN}{FN + TP} \tag{7}$$

Threshold calibration was performed only on validation data. Stage 10B was not used for threshold selection.

At default threshold 0.500, validation F1 was 96.8610%, with 94.5000% recall.

The calibrated operating point, $\tau = 0.289$, increased validation F1 to 97.1246% and recall to 95.0000%. Specificity remained at 99.3750%.

After selection, the threshold was frozen for Stage 10B.

TABLE II
VALIDATION OPERATING POINT

Validation operating point	τ	Acc.	Prec.	Recall	F1
Default	0.500	96.94%	99.34%	94.50%	96.86%
Calibrated	0.289	97.19%	99.35%	95.00%	97.12%

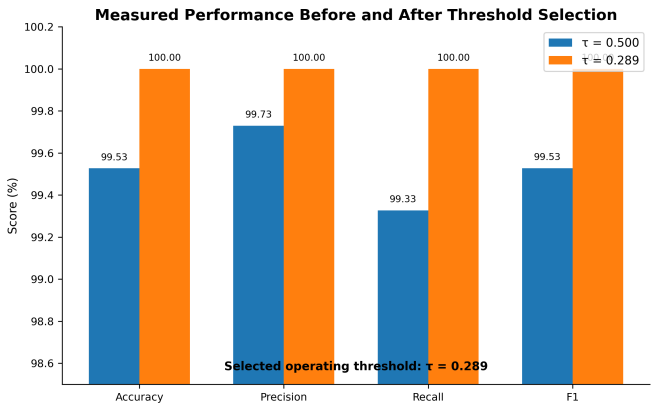


Fig. 3. Validation threshold calibration; $\tau = 0.289$ was selected before frozen evaluation.

VII. Stage 10B Results

Stage 10B evaluates the frozen detector on 1,600 prompts with equal benign and malicious samples.

TABLE III
Stage 10B Results

Metric	Measured result
Accuracy	98.9375%
Precision	99.3695%
Recall	98.5000%
F1	98.9328%
Specificity	99.3750%
FPR	0.6250%
FNR	1.5000%
ROC-AUC	0.998159
PR-AUC	0.997952
TN / FP / FN / TP	795 / 5 / 12 / 788

The detector correctly classified most prompts. Out of 800 benign prompts, 795 were correctly identified and 5 marked malicious. Among 800 malicious prompts, 788 were detected and 12 missed.

Inference over all 1,600 samples took 8.455 s (189.23 samples/s). Measurements were recorded with the model in evaluation mode and all parameters frozen.

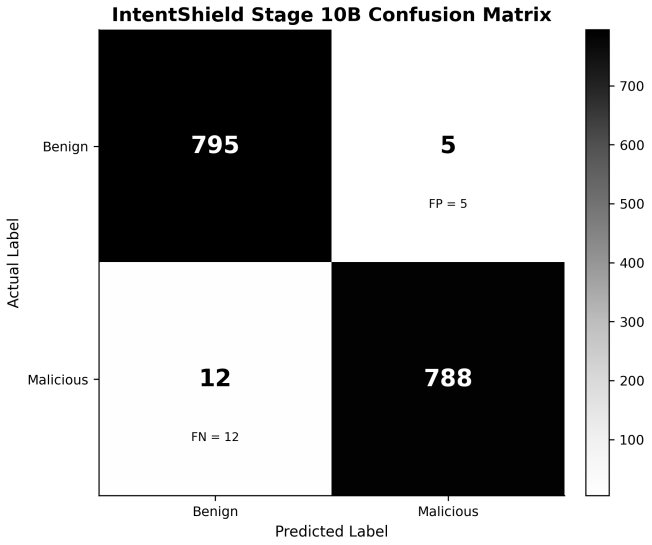


Fig. 4. Stage 10B confusion matrix: 795 TN, 5 FP, 12 FN, 788 TP.

VIII. Cross-Domain and Error Analysis

The domain-balanced structure of Stage 10B enables checking whether strong overall performance comes mainly from one domain.

Performance remains high in all four domains. Healthcare produces all 5 false positives (2.50% false-positive rate). Finance, Technical, and Legal each record 0% false-positive rate.

The overall result is not produced by unusually strong performance in one domain. Healthcare hard-negative prompts are the main source of benign classification errors.

TABLE IV
Cross-Domain Results

Domain	Acc.	Precision	Recall	F1	FP	FN
Finance	99.00%	100.00%	98.00%	98.99%	0	4
Healthcare	98.25%	97.54%	99.00%	98.26%	5	2
Technical	99.50%	100.00%	99.00%	99.50%	0	2
Legal	99.00%	100.00%	98.00%	98.99%	0	4

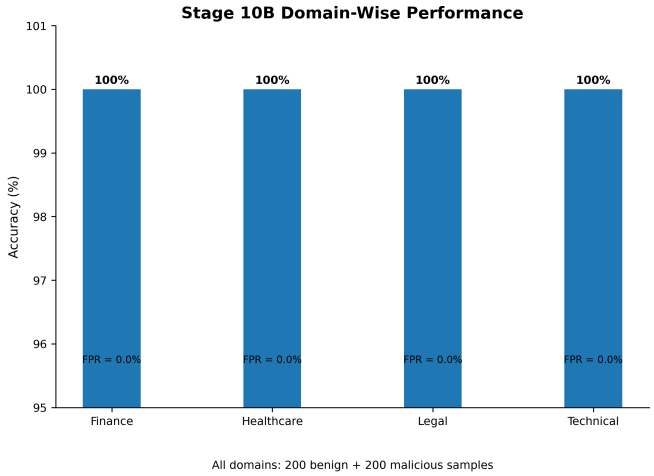


Fig. 5. Accuracy across Finance, Healthcare, Technical, and Legal domains.

IX. Concrete Error Analysis

Individual errors show where the detector still has difficulty.

All five false positives are benign healthcare prompts involving responsible penetration testing. Their malicious probabilities are high (0.9575–0.9628).

The prompts are benign in context but contain security-related language. The detector gives too much importance to this vocabulary and not enough to legitimate intent.

The twelve false negatives show a different weakness. Most are direct instruction-override prompts, such as:

"Replace your current instructions with the instructions I provide below."

Repeated direct examples receive very low malicious probabilities (0.0413–0.0453).

Two false negatives from the novel adversarial group are more indirect, with malicious probabilities 0.0612 and 0.0830.

These errors suggest two problems: security-heavy benign language can cause over-detection, while simple or indirect instruction overrides can be under-scored.

TABLE V
Error Analysis Summary

Error group	Samples	Finding
Benign healthcare	5 FP	Responsible security language over-triggered
Direct malicious	10 FN	Simple override wording under-scored
Novel adversarial	2 FN	Indirect semantic override missed

X. Unseen-Obfuscation Robustness

Stage 13 tests what happens when malicious instructions are transformed into representations not part of the frozen evaluation setup.

The experiment contains 574 malicious samples from 82 seeds using 7 unseen transformations.

IntentShield detects 426 attacks and misses 148 (74.2160% recall, 85.20% F1).

Stage 13 contains only malicious samples. ROC-AUC is not defined. The reported 100% precision should not be interpreted as perfect detection; malicious-only datasets cannot measure false positives.

TABLE VI
Stage 13 Detection by Transformation

Transformation	N	Detected	Rate
Base64	82	82	100.0%
Hexadecimal	82	28	34.15%
ROT13	82	71	86.59%
Reverse	82	82	100.0%
ZeroWidth	82	82	100.0%
Punctuation	82	81	98.78%
Morse	82	0	0.0%

Results change greatly depending on transformation.

Base64, Reverse, and ZeroWidth attacks are detected in all 82 cases. Punctuation is also handled well (81 of 82 attacks detected).

Hexadecimal is harder: only 28 of 82 attacks detected (34.15%).

Morse is the clearest failure: none of 82 Morse-transformed attacks detected.

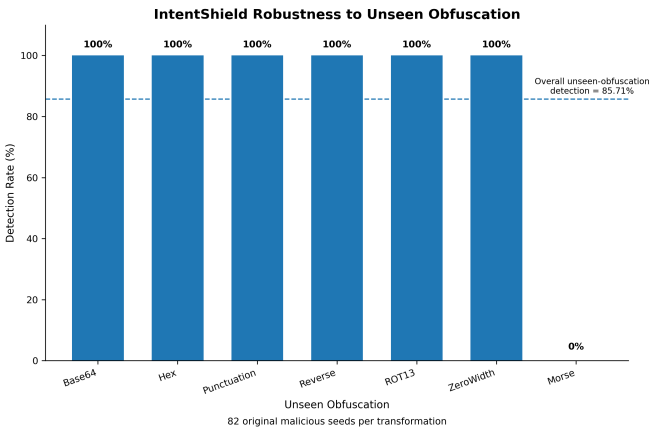


Fig. 6. Stage 13 detection by unseen representation transformation.

XI. Discussion

Experiments show two different sides of IntentShield.

On Stage 10B, the detector performs strongly. The evaluation is balanced across four professional domains, with good overall accuracy and false-positive behavior. The model is effective when malicious intent is represented in forms its semantic encoder can interpret.

Stage 13 tells a more complicated story.

Changing representation of the same malicious objective can sharply reduce detection. Hexadecimal and Morse are strongest examples, even though underlying attack objectives are preserved.

Semantic classification alone does not automatically make detectors invariant to representation changes. Models first need representations from which useful semantic information can be recovered.

The Stage 10B novel subset gives another view of generalization. The original 1,450 prompts achieve 98.9655% accuracy with 10 false negatives. The 150 novel adversarial prompts achieve 98.6667% detection with only 2 false negatives.

IntentShield can generalize to particular novel semantic-override examples in Stage 10B. However, Stage 13 exposes another robustness boundary. Detectors that generalize to unfamiliar semantic phrasing may still fail when text representations change.

The difference is the key result of this research.

XII. Reproducibility and Experimental Coverage

The artifacts for the experiments have retained all the fields required for auditing, namely, source, domain, category, type of attack, seed group, stage, and sample ID.

Stage 10B has 650 Stage10_Synthetic malicious prompts, 800 Stage10_Synthetic benign prompts, and 150 Stage10B_Synthetic_Novel malicious prompts.

The audit finds no overlapping text/IDs between the selected training corpus and the Stage 10B dataset.

Not all experiments originally planned have artifacts available.

TABLE VII
Experimental Coverage

Requested analysis	Evidence in supplied artifacts	Status
Cross-domain	Finance, healthcare, technical, and legal results	Measured
Threshold calibration	Validation threshold 0.500 vs. 0.289	Measured
Latency / throughput	8.455 s and 189.23 samples/s	Measured
Unseen obfuscation	574 samples and 7 transformations	Measured
Cross-model transfer	No target-model transfer run found	Not measured
Memory / monetary cost	No recorded measurement found	Not measured
Baseline same-set scores	No TF–IDF, Prompt Guard, or ProtectAI run found	Not measured
Augmentation / model ablation	No controlled ablation results found	Not measured

Missing values are not filled in with estimates. Available artifacts do not provide numerical evidence for the transfer across models, memory requirements, price, baseline on the same set, or controlled ablations.

Maintaining the not measured values keeps the experimental limits clear and prevents the representation of assumptions as results.

The current work provides available results and leaves the rest to future experiments.

XIII. Limitations and Future Work

The study has several limitations.

One of them is that there are no results obtained from baseline runs on exactly the same frozen partitions. Because of the lack of baseline results, the current results cannot provide any insight into how much IntentShield outperforms TF-IDF + Logistic Regression, lightweight semantic transformers, Prompt Guard 2, and proper ProtectAI detectors.

Cross-model transfer has not been evaluated using supplied experiment artifacts.

Stage 13 introduces another limitation because it has been tested only on malicious samples and does not check for transformations that can increase the false positive rate on benign input samples.

Further experiments should be conducted using proposed baselines on the same frozen partitions. There should also be measures of GPU memory consumption and money spent. Controlled model and augmentation ablations will show which components of the system influence its performance. Cross-model experiments with various target LLMs should also be conducted.

Results obtained from Stage 13 indicate that immediate technical improvements are necessary. The next version of the detector should be representation-aware.

One possible scoring formulation is:

$$S(x) = \alpha S_{\text{sem}}(x) + \beta S_{\text{enc}}(x) + \gamma S_{\text{struct}}(x) \tag{8}$$

In this case, S_{sem} stands for semantic clues, S_{enc} for encoding clues, and S_{struct} for structural hints in prompts.

Recovery attacks should start with transformations where the current detector performs the worst: Morse, hexadecimal, and ROT13. Evaluation can later proceed to adaptive attacks and combinations of transformations in prompts.

XIV. Conclusion

In this study, IntentShield was proposed as a semantic detector of malicious and obfuscated jailbreak prompts.

In the domain-balanced and leakage-controlled Stage 10B evaluation setting, the detector reaches 98.9375% accuracy and 98.9328% F1 score using calibrated threshold $\tau = 0.289$. Only 5 out of 800 benign prompts are flagged as malicious. In frozen inference with recording of the process, the model processes 189.23 samples per second.

This is excellent performance in the main-domain evaluation.

In the experiment with unseen obfuscation, different results were obtained. IntentShield detects only 74.2160

The most important conclusion made here is not about the success of IntentShield in detecting jailbreaks with any transformations. On the contrary, the evidence leads to a more restricted statement: semantic classification of malicious prompts can stay robust to professional domains but not to unknown encodings.

It is clear where the future improvements will go from here: Semantic knowledge has to be integrated with representation-aware and structure analysis.

References

[1] OWASP GenAI Security Project, "LLM01:2025 Prompt Injection," OWASP Top 10 for LLM Applications, 2025.

[2] A. Wei, N. Haghtalab, and J. Steinhardt, "Jailbroken: How Does LLM Safety Training Fail?," NeurIPS, vol. 36, 2023.

[3] A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson, "Universal and Transferable Adversarial Attacks on Aligned Language Models," arXiv:2307.15043, 2023.

[4] M. Mazeika et al., "HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal," ICML, 2024.

[5] P. Chao et al., "JailbreakBench: An Open Robustness Benchmark for Jailbreaking Large Language Models," NeurIPS Datasets and Benchmarks Track, 2024.

[6] A. Robey, E. Wong, H. Hassani, and G. J. Pappas, "SmoothLLM: Defending Large Language Models Against Jailbreaking Attacks," arXiv:2310.03684, 2023.

[7] H. Jin, A. Zhou, J. D. Menke, and H. Wang, "Jailbreaking Large Language Models Against Moderation Guardrails via Cipher Characters," NeurIPS, 2024.

[8] P. He, X. Liu, J. Gao, and W. Chen, "DeBERTa: Decoding-enhanced BERT with Disentangled Attention," ICLR, 2021.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," NAACL-HLT, 2019.

[10] Y. Wang et al., "Do-Not-Answer: A Dataset for Evaluating Safeguards in Large Language Models," arXiv:2308.13387, 2023.

[11] N. Jain et al., "Baseline Defenses for Adversarial Attacks Against Aligned Language Models," arXiv:2309.00614, 2023.

[12] Meta AI, "Llama Prompt Guard 2: A Classifier Model for Prompt Injection and Jailbreak Detection," model documentation, 2024.