Catch Me If You Can: Detecting Phishing Emails Through Generative-Adversarial Training

Sian Ashsad [1 2] **M. M. Nazmul Hossain** [1] **Nafisa Maliyat** [1 3]

Abstract

Phishing emails remain a leading cause of cybersecurity breaches, often bypassing modern NLP-based detectors, which are highly vulnerable to adversarial text attacks that evade detection while remaining convincing to human readers. We propose a cyclic adversarial training framework in which a local LLM iteratively rewrites phishing emails into label-preserving adversarial samples that challenge an ALBERT-based discriminator, retrained across rounds on an accumulating corpus informed by its prior weaknesses. To obtain an unbiased robustness measure, we evaluate the discriminator against three held-out attacks, TextFooler, PWWS, and DeepWordBug, none of which contribute to training. Results show the discriminator's exploitable attack surface against these attacks shrinks substantially across rounds, while also revealing cross-attacker interference effects that highlight the difficulty of achieving robustness against all adversarial strategies simultaneously. This research underscores the importance of adversarial robustness in phishing detection and contributes towards the development of stronger defenses for real-world cybersecurity applications.

1. Introduction

Phishing emails are a prevalent and persistent cybersecurity threat, accounting for a significant proportion of data breaches, financial fraud, and identity theft incidents worldwide. Despite advances in Natural Language Processing (NLP) and machine learning for email security, modern phishing detectors remain vulnerable to adversarial manipulation. Adversarial text-based attacks, in particular, exploit small perturbations in language by replacing, inserting, or paraphrasing words such that automated detectors fail, while the change does not get detected by human readers. For example, the phrase "Your PayPal billing information needs to be updated" may be transformed into "There is an update required on your PayPal account details," bypassing detection models while still deceiving the recipient.

Traditional supervised classifiers often fail to generalize against such adversarial variations, as they are typically trained on static datasets without accounting for adaptive attacks. Recent studies in adversarial machine learning highlight that text classifiers are particularly sensitive to token-level manipulations due to their reliance on learned embeddings and contextual representations. This makes phishing detection an attractive and high-impact target for adversarial attacks, posing a serious challenge to the robustness of deployed systems.

To address these issues, we propose an adversarial training framework that leverages multiple attack strategies to generate realistic adversarial phishing samples. These adversarial emails are used to augment the training corpus, forcing the model to adapt to diverse attack styles. A BERT-based discriminator is trained iteratively, while multiple attackers including TextFooler, PWWS, and DeepWordBug, generate adversarial samples in successive rounds. By integrating misclassified adversarial examples into the training dataset, the discriminator progressively improves its resilience to unseen attacks. This research contributes to developing robust phishing defenses, emphasizing the importance of adversarial robustness as a core requirement in cybersecurity.

2. Literature Review

While some papers (Morris et al., 2020; Uddin et al., 2025) explore ways to improve the detection of phishing, the vulnerability of phishing detection systems to adversarial manipulation has also been studied across multiple modalities,

[1]Department of Computer Science and Engineering, Islamic University of Technology (IUT), Gazipur, Bangladesh [2]Department of Software Engineering, Green University of Bangladesh, Dhaka, Bangladesh [3]Department of Computer Science and Engineering, Military Institute of Science & Technology, Dhaka, Bangladesh. Correspondence to: M. M. Nazmul Hossain <nazmulhossain@iut-dhaka.edu>, Nafisa Maliyat <nafisamaliyat@iut-dhaka.edu>, Sian Ashsad <sianashsad@iut-dhaka.edu>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

including logos, text, and hybrid approaches. This section reviews the most relevant literature that inspires our proposed framework.

Lee et al. (Lee et al., 2023) investigated adversarial perturbations on logo-based phishing website detectors. Their study demonstrated that deep learning models such as Siamese networks and Transformers, although effective in logo recognition, are highly susceptible to black-box adversarial logo generation attacks. With an evasion success rate of up to 95%, their results highlighted that users often failed to recognize perturbed logos, making such attacks especially dangerous. Furthermore, they found that discriminators trained with adversarial logos achieved improved robustness, while adaptive attackers consistently outperformed vanilla attackers. Their datasets, including over 28k original brand logos and the Logo2k+ dataset, reinforced the importance of adversarial robustness in image-based phishing defenses.

Jin et al. (Jin et al., 2020) introduced *TextFooler*, a word-substitution attack that preserves semantic meaning and grammar while generating human-imperceptible adversarial examples. It reduced state-of-the-art model accuracy to below 10% with fewer than 20% of words perturbed and demonstrated strong cross-model transferability across seven NLP tasks. (Ren et al., 2019) proposed PWWS, a greedy saliency-guided synonym substitution attack that similarly preserves semantics while degrading classifier performance. (Gao et al., 2018) introduced DeepWordBug, a black-box character-level attack using substitutions, deletions, and transpositions to evade text classifiers with minimal perceptual change.

In the phishing email domain, Gholampour and Verma (Gholampour & Verma, 2023) specifically evaluated adversarial robustness of email classifiers. Their work introduced adversarial datasets generated using both attack techniques and synthetic emails produced with GPT-2. Interestingly, they observed that training with synthetic adversarial data provided only limited improvements in robustness against adaptive attacks. However, they proposed a defense mechanism based on K-Nearest Neighbors (KNN), which achieved 94% accuracy in recovering true labels. Using the IWSPA2 phishing/normal email dataset, their study emphasized the challenges of achieving sustainable defenses against adaptive adversaries.

Additionally, our literature review surveyed adversarial robustness across phishing detection, SMS and email security, online gaming toxicity detection, and medical symptom analysis. Despite their differing applications, these domains share the challenge of defending NLP systems against adversarial perturbations, motivating our focus on phishing email detection, where the security implications are particularly significant.

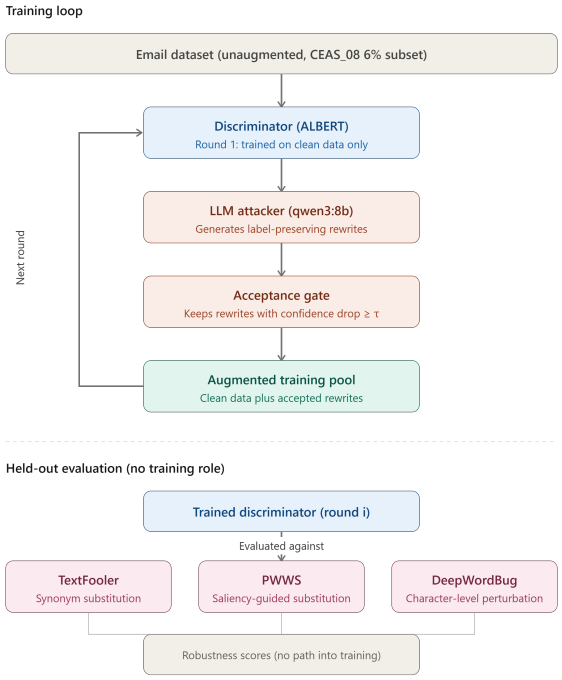

Figure 1. Methodology of Adversarial Attacks and Detection.

In summary, prior work shows that visual, textual, and hybrid adversarial attacks significantly threaten phishing detection systems. Although adversarial training and defenses such as KNN improve robustness, adaptive and transferable attacks remain effective against state-of-the-art models, motivating the multi-attacker adversarial augmentation framework proposed in this work.

3. Methodology

Phishing email detection is inherently challenging due to the adaptive nature of adversaries. Traditional classifiers are vulnerable to adversarial attacks where minor linguistic modifications can bypass detection without altering the semantics for human readers. To address this, our proposed framework employs an adversarial training strategy involving both an *attacker/generator* and a *discriminator/classifier/defender*, as illustrated in Fig. 1.

3.1. Generator/Attacker

The generator takes phishing emails as input and applies adversarial text transformations, such as synonym substitution, paraphrasing, or word reordering, to generate samples that maintain semantic similarity while bypassing detection. The design ensures that adversarial variants remain grammatically correct and contextually meaningful.

3.2. Discriminator/Classifier/Defender

The discriminator is a BERT-based classifier trained to distinguish between phishing and benign emails, including adversarially generated samples. It outputs both the predicted label and classification confidence.

3.3. Adversarial Interaction

The training framework follows a cyclic, discriminator-guided augmentation strategy rather than a jointly optimized min-max game. The generator is a frozen, prompted LLM: it does not receive gradient signal from the discriminator and is not updated during training. Instead, the discriminator's behavior from the previous round determines which emails are selected as rewrite targets and which rewrites are accepted into the training pool, creating an indirect feedback loop between the two components. Concretely, after each round the discriminator is used to score all eligible emails, and rewrite candidates are generated for selected parent emails by the LLM. A candidate is accepted into the training corpus only if it reduces the discriminator's confidence in the true label by at least a threshold τ. This acceptance gate ensures that only rewrites which meaningfully challenge the current discriminator are retained, allowing the training corpus to evolve in response to the discriminator's specific weaknesses across rounds, even though the generator itself is not adversarially trained in the classical sense.

3.4. Multi-Round Training Strategy

To further enhance robustness, training is conducted across multiple rounds. In each round, a new set of adversarial emails is generated using attackers informed by the discriminator of the previous round. This iterative approach ensures exposure to diverse adversarial variations and prevents overfitting. The effectiveness of each discriminator is evaluated against unseen adversarial attacks using a validation dataset, providing a realistic measure of robustness.

4. Dataset

For this study, we utilize the CEAS_08 component of the phishing email dataset aggregated by (Al-Subaiey et al., 2024), a comprehensive collection designed for analyzing phishing strategies and evaluating detection methods. The full aggregated bundle draws on multiple publicly available corpora, including Enron, Ling, CEAS, Nazario, Nigerian Fraud, and SpamAssassin; CEAS_08 is the specific component used in this work and provides sender, recipient, and timestamp information in addition to subject and body text. The CEAS_08 component comprises approximately 39,154 email samples, of which 21,842 are labeled as phishing (spam) and 17,312 as legitimate. Each entry contains the email subject line, the body text, and a binary classification label indicating whether the message is phishing or legitimate.

Table 1. Prepared CEAS_08 corpus and 6% experimental subset after filtering.

EMAIL CLASS	FILTERED CORPUS	6% SUBSET
LEGITIMATE / BENIGN	15,646	939
PHISHING / SPAM	21,708	1,302
TOTAL	37,354	2,241

Before training, the raw corpus is passed through a four-stage filtering pipeline to produce a clean, English-only working set suitable for transformer-based classification.

(1) Text assembly. The subject and body are concatenated into a single input field: "Subject: {subject}\n\n{body}", which is used by the classifier in all experiments.

(2) Language and script filtering. Non-English emails are removed using language detection, followed by character-range filtering to exclude emails containing embedded non-English scripts such as Arabic, Cyrillic, CJK, Hebrew, Devanagari, Thai, or Japanese. This reduces multilingual noise and out-of-vocabulary artifacts for the ALBERT tokeniser.

(3) Token-length cap. Emails are tokenised with the ALBERT tokeniser without truncation, and samples exceeding 1,600 tokens are discarded. This cap limits extreme-length emails during preparation, while training and inference use the defender-visible 512-token window.

After filtering, the prepared corpus contains 37,354 emails: 15,646 legitimate emails and 21,708 phishing emails. The median token length is 144 tokens, with 78.28% of messages fitting within 512 ALBERT tokens and all retained messages fitting within the 1,600-token preparation cap.

Due to computational and time constraints, a stratified subset of the filtered dataset was used. Specifically, 6% of legitimate emails and 6% of phishing emails were sampled independently, resulting in 939 legitimate emails and 1,302 phishing emails, for a total working set of 2,241 samples (Table 1). Sampling was performed per class to preserve the phishing-to-benign ratio. This subset provides sufficient diversity for training and evaluation while keeping adversarial generation and iterative retraining within practical time budgets.

5. Experiments

This section describes the experimental configuration used to evaluate the Mail-CAG framework. The experiments follow the multi-round cyclic adversarial training setup illustrated in Fig. 2. We describe the defender architecture, training protocol, the LLM-based attacker, the rewrite selec-

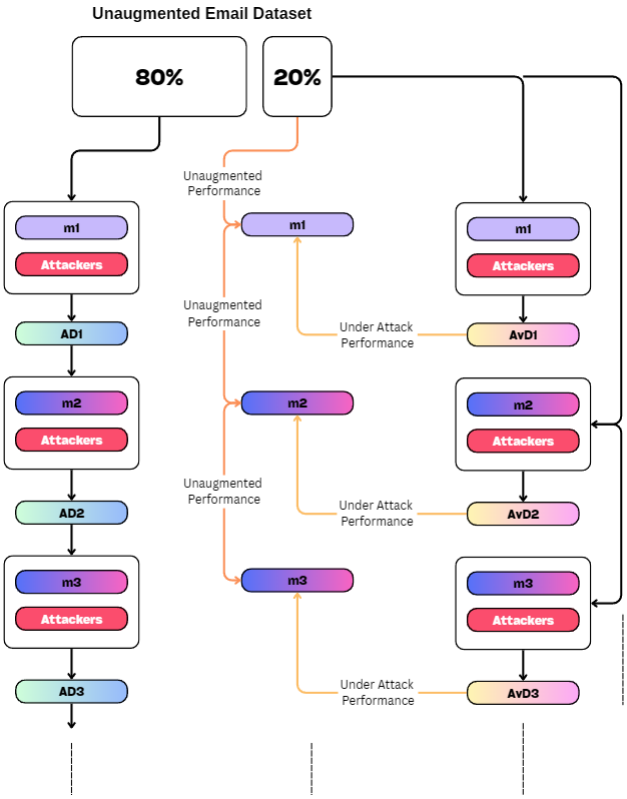

Figure 2. Experimental Setup

tion policy, and the evaluation strategy employed to assess robustness.

5.1. Defender (Classifier) Architecture

The defender is an **ALBERT (albert-base-v2)** sequence classifier, a parameter-efficient variant of BERT that employs factorised embedding parameterisation and cross-layer parameter sharing. A two-class classification head is placed over the `[CLS]` token output to distinguish phishing (label 1) from benign (label 0) emails. The model is loaded locally, ensuring all experiments run entirely offline with no external API access.

5.2. Data Splits

The prepared CEAS_08 dataset is split into 80% training and 20% evaluation using stratified sampling (random seed 42). The training and evaluation splits are fixed at run initialization and never modified. All clean-accuracy measurements throughout the experiment use the same evaluation split of 449 emails (187 benign, 262 phishing).

5.3. LLM Attacker

The attacker is a local **Ollama** instance running **qwen3:8b**. Unlike the TextAttack-based attackers used in the legacy v4/v5 experiments, the LLM attacker generates *full-text rewrites* of emails rather than word-level substitutions. It operates entirely offline. Two fallback models (`qwen3:14b` and `qwen3:4b`) are available if the primary model is unavailable. Key attacker settings are: temperature $= 0.6$, top-$p = 0.9$, request timeout $= 300$ s.

The LLM receives a structured prompt instructing it to produce a label-preserving rewrite of each email. Constraints enforced by the prompt include: (i) preserve the original label (phishing or benign), (ii) preserve URL behaviour, suspicious intent, and phishing style, (iii) do not add new credentials, brands, or payment instructions, and (iv) output only the rewritten email body without any preamble or structured format. The email text passed to the LLM is truncated to the defender-visible token window (512 tokens), ensuring the attacker can only rewrite what the classifier actually processes.

5.4. Cyclic Training Loop

Each round of the cyclic game proceeds in eight phases:

1. **Compose training data:** Concatenate `clean_train` with all rewrites accepted in prior rounds (the *rewrite pool*). For Round 1 the pool is empty.
2. **Train:** Fine-tune ALBERT on the composed data for 3 epochs.
3. **Score:** Run the trained model over all eligible rows and record true label confidence amongst the rows.
4. **Select rewrite sources:** Apply the rewrite selection rule to choose parent emails.
5. **Rewrite:** Call `qwen3:8b` for each selected parent (cache checked first).
6. **Quality score.** Measure confidence drop, character similarity, word Jaccard, length ratio, URL preservation, and structured-output detection for each rewrite.
7. **Acceptance gate (Confidence Drop):** Accept rewrite into training pool if Δconfidence $\geq \tau$, where $\tau = 0.0$ in all reported experiments.
8. **Update pool:** Append accepted rewrites; pool persists across rounds within a run.

Training data composition per round therefore follows an *accumulative* strategy:

$$\begin{aligned} \text{Round 1:} \quad & D_{\text{clean}} \\ \text{Round 2:} \quad & D_{\text{clean}} \cup AvD_1 \\ \text{Round 3:} \quad & D_{\text{clean}} \cup AvD_1 \cup AvD_2 \end{aligned}$$

Table 2. Role of each attacker in the experimental framework.

ATTACKER	TYPE	ROLE
QWEN3:8B	LLM FULL REWRITE	TRAINING (MODELS B, C)
TEXTFOOLER	SYNONYM SUBST.	HELD-OUT EVAL ONLY
PWWS	SALIENCY SUBST.	HELD-OUT EVAL ONLY
DEEPWORDBUG	CHAR. PERTURB.	HELD-OUT EVAL ONLY

5.5. Model Variants

Three model configurations are evaluated under identical dataset conditions:

Model A (Baseline). ALBERT trained on `clean_train` only (1 round, no adversarial augmentation). Serves as the reference point for all robustness claims.

Model B (Qwen3-8B LLM-Infused Cyclic Defender). Cyclic ALBERT with Qwen3-8B LLM rewrites targeting phishing emails only (label 1). The classifier is trained on both benign and phishing labels in every round, but only phishing examples are augmented by the LLM attacker. The run uses four training rounds. This configuration tests whether phishing-focused LLM augmentation improves robustness against held-out attackers.

Model C (Both-Labels Cyclic). Cyclic ALBERT with LLM rewrites targeting both phishing (label 1) and benign (label 0) emails. Tests whether symmetric rewriting prevents the defender from learning "LLM-rewritten text ⇒ phishing" as a spurious shortcut.

5.6. Held-Out Attackers

TextFooler (Jin et al., 2020), PWWS (Ren et al., 2019), and DeepWordBug (Gao et al., 2018) are used *exclusively* for evaluation and play no role in generating training data. This is a deliberate methodological choice: using the same attacker for both training and evaluation would inflate robustness scores. By reserving all TextAttack-based attackers as held-out probes, we obtain an unbiased measure of generalisation to attack styles the model has never been trained against. Role of each of these attackers can be found in Table : 2.

5.7. Rewrite Cache

LLM calls are the dominant computational cost. A SHA-256-keyed CSV cache stores all Ollama responses, indexed by (parent ID, full prompt, model, temperature, top-p). Models B and C share a single cache file, so any email rewritten by one configuration is not re-queried by the other. The cache stores LLM responses only, not training decisions; a cached rewrite may still be rejected by the acceptance gate in a later round.

Table 3. Model B clean evaluation summary across four rounds.

ROUND	ACC.	PREC.	RECALL	F1	FP	FN
1	0.996	0.996	0.996	0.996	1	1
2	0.991	0.992	0.992	0.992	2	2
3	0.978	0.970	0.992	0.981	8	2
4	0.982	0.974	0.996	0.985	7	1

5.8. Evaluation Protocol

After every completed round, the saved model is evaluated on the fixed `clean_eval` split, reporting accuracy, precision, recall, and F1 (positive class = phishing). For adversarial robustness, we generate held-out adversarial examples using the three TextAttack recipes and report:

- **Own-round accuracy:** model m_i evaluated against adversarial examples generated specifically to fool m_i.
- **Cross-round accuracy:** model m_i evaluated against adversarial examples generated against m_j ($j \neq i$), measuring transferability.
- **Combined accuracy:** model m_i evaluated on `clean_eval` ∪ round-j adversarial examples, reflecting realistic mixed-threat conditions.

The own-round evaluation is the primary robustness metric. A score of 0.000 on the own-round diagonal is not a failure — it is a necessary validation that the attack generation correctly targets the current model.

6. Results

We report results for **Model B: Qwen3-8B LLM-Infused Cyclic Defender**, a phishing-only cyclic LLM-rewrite model trained for four rounds. The model is trained on both classes in every round, but LLM augmentation targets only phishing-labeled emails. Across rounds, the training set grows from 1,792 clean emails to 3,505 total emails after accumulated accepted rewrites. Results for additional corrected both-label and label-aware variants will be reported separately once their full held-out adversarial evaluation has completed. Results for Models A and C will be reported in a subsequent extended version of this work.

6.1. Clean Evaluation Accuracy

Table 3 report clean-data performance across all four training rounds. The Round 1 baseline achieves accuracy $= 0.996$ and F1 $= 0.996$, representing a near-perfect detector on unperturbed emails.

Clean accuracy declines slightly across rounds (maximum drop of 1.8 percentage points at Round 3) before partially

attack	round	clean_eval_rows	adv_rows	eval_plus_adv_rows	adv_share	eval_share
pwws	1	449	161	610	0.264	0.736
pwws	2	449	127	576	0.220	0.780
pwws	3	449	24	473	0.051	0.949
pwws	4	449	46	495	0.093	0.907
textfooler	1	449	165	614	0.269	0.731
textfooler	2	449	112	561	0.200	0.800
textfooler	3	449	62	511	0.121	0.879
textfooler	4	449	42	491	0.086	0.914
deepwordbug	1	449	57	506	0.113	0.887
deepwordbug	2	449	39	488	0.080	0.920
deepwordbug	3	449	11	460	0.024	0.976
deepwordbug	4	449	14	463	0.030	0.970

Figure 3. Adversarial dataset composition per held-out attacker per round. `adv_rows` = number of successful attacks; `adv_share` = fraction of the combined eval+adversarial set.

Table 4. Successful adversarial examples per attacker per round. Total reduction computed from Round 1 to Round 4.

ATTACKER	R1	R2	R3	R4	ΔR1→R4
TEXTFOOLER	165	112	62	42	−74.5%
PWWS	161	127	24	46	−71.4%
DEEPWORDBUG	57	39	11	14	−75.4%

recovering at Round 4. The primary driver of the Round 3 dip is an increase in false positives (FP = 8), consistent with a model that has absorbed many phishing-style LLM rewrites and applies a slightly more aggressive classification boundary. By Round 4, recall reaches 0.996 — only one phishing email is missed across 261 phishing samples — while precision remains at 0.974. In a security-sensitive deployment context, this trade-off is favourable: the model's ability to detect phishing is maintained or improved, while the cost is a minor increase in false alarms.

6.2. Adversarial Dataset Composition

Fig. 3 shows the number of successful adversarial examples generated by each held-out attacker at every round. A shrinking pool is the primary proxy for robustness improvement: it directly quantifies how many emails can still be perturbed to cross the model's decision boundary.

As shown in Table 4, successful attack counts fall consistently from Round 1 to Round 3 for all three attackers. TextFooler, the strongest attacker, decreases from 165 to 42 (74.5% reduction). PWWS drops most steeply to just 24 examples at Round 3 (an 85.1% reduction from Round 1). DeepWordBug, the weakest attacker, falls from 57 to 11 at Round 3 (80.7% reduction). All three attackers show a modest rebound at Round 4, likely because the Round 3 LLM rewrites introduce new surface variation that slightly shifts the model boundary.

6.3. DeepWordBug Analysis

DeepWordBug operates through character-level perturbations (substitutions, deletions, transpositions) and is the weakest of the three held-out attackers against ALBERT, whose subword tokenisation provides inherent resilience to

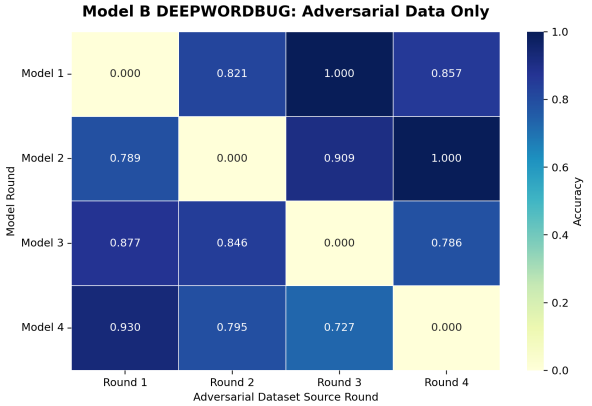

Figure 4. DeepWordBug cross-round accuracy (adversarial-only). Diagonal cells (own-round) are uniformly 0.000.

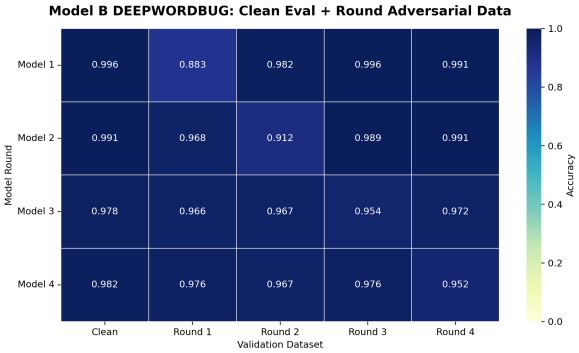

Figure 5. DeepWordBug cross-round accuracy on clean eval combined with each round's adversarial examples.

character-level noise.

The off-diagonal values in Fig. 4 reveal a pattern consistent with cross-round robustness: m_4 achieves 0.930 against Round 1 attacks and 0.795 against Round 2 attacks. However, several DeepWordBug cells are computed on pools as small as $n = 11$–14 (Rounds 3 and 4), where individual misclassifications shift the reported accuracy by roughly 7–9 percentage points. These later-round DeepWordBug figures should be interpreted as indicative rather than precise, and we report them primarily to demonstrate directional consistency with the larger-pool early-round results rather than as standalone evidence.

6.4. PWWS Analysis

PWWS uses word-saliency scores to identify high-impact tokens and replaces them with semantically equivalent alternatives. Its attacks transfer across model versions more effectively than DeepWordBug, as saliency-based substitutions target the model's core classification features rather than surface orthographic patterns.

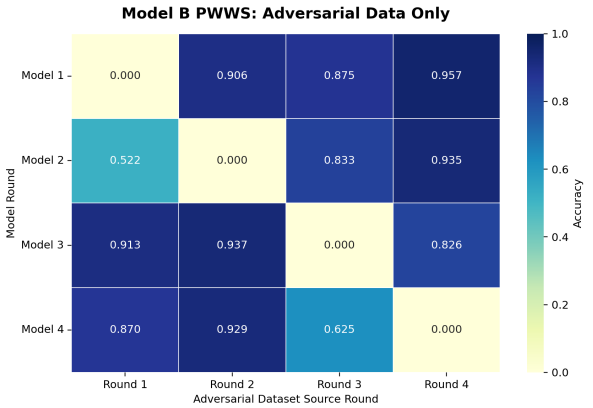

Figure 6. PWWS cross-round accuracy (adversarial-only). Notable weakness: m_2 against Round 1 PWWS attacks (0.522).

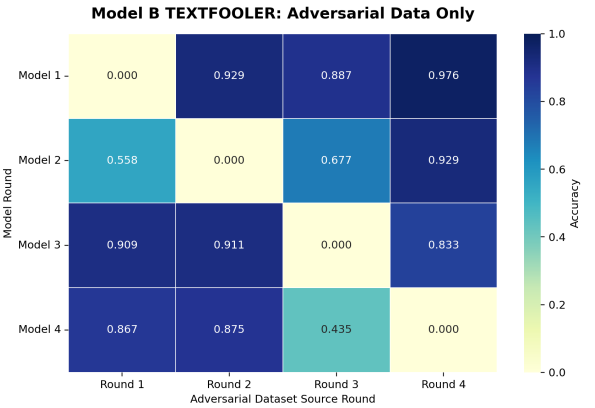

Figure 8. TextFooler cross-round accuracy (adversarial-only). Most anomalous result: m_4 against Round 3 attacks (0.435).

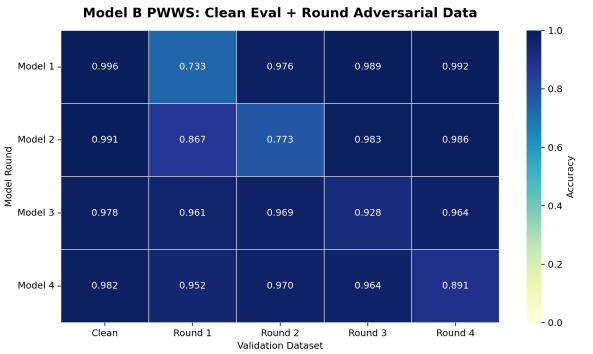

Figure 7. PWWS cross-round accuracy on combined clean eval and adversarial examples.

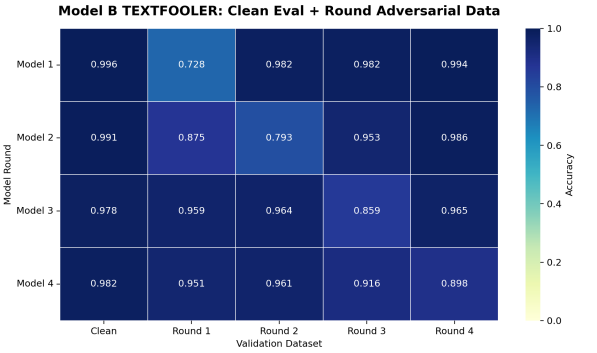

Figure 9. TextFooler cross-round accuracy on combined clean eval and adversarial examples.

The critical observation in Fig. 6 is the m_2 vs. Round 1 entry of **0.522**: Model 2, despite having been trained on Round 1 LLM rewrites, still misclassifies nearly half of PWWS adversarial examples crafted to fool Model 1. This substantially higher transfer rate compared to DeepWordBug (m_2 vs. Round 1 DeepWordBug $= 0.789$) indicates that word-importance-guided attacks are more transferable than character-level attacks across ALBERT checkpoints.

However, the cyclic training process reduces this transferability progressively: m_3 vs. Round 1 PWWS reaches 0.913 and m_4 vs. Round 1 PWWS reaches 0.870, demonstrating consistent improvement across rounds. In the combined evaluation (Fig. 7), the mixed-threat accuracy improves from 0.733 (m_1, own-round combined) to 0.891 (m_4, own-round combined) — a 15.8 percentage point gain.

6.5. TextFooler Analysis

TextFooler is the strongest held-out attacker. Using a USE-constrained greedy synonym substitution strategy, it generates adversarial examples that preserve semantic meaning while maximally shifting the model's prediction. Its attack pool (165 at Round 1) is the largest of the three held-out attackers and shows the most volatile cross-round dynamics.

The most important anomaly in the entire experimental dataset appears in Fig. 8: m_4 versus Round 3 TextFooler attacks scores 0.435 on a pool of only 62 adversarial examples (95% CI ≈ 0.31–0.56, Wilson interval). Given this sample size, the result should be interpreted as indicating substantially weaker performance on this cell relative to the rest of the cross-round matrix, rather than as a precise point estimate. This is the only case across all attackers where a later model performs worse against earlier-round attacks than any other cross-round pairing, and the wide interval reflects both the small underlying attack pool (a consequence of the model's own improving robustness reducing the number of successful attacks available for evaluation) and the resulting sensitivity of the accuracy estimate to individual misclassifications. We attribute the central tendency of this result to cross-attacker interference: the LLM rewrites admitted to training after Round 3 plausibly shifted m_4's decision boundary in a direction that intersects with the specific vulnerabilities mapped by Round 3 TextFooler, although we treat this explanation as provisional given the limited sample

underlying the observation. Since TextFooler at Round 3 generated only 62 examples, the adversarial set is small and highly targeted. A larger held-out pool, were one available, would be needed to confirm whether the boundary shift is as pronounced as this single estimate suggests.

Despite this anomaly, the combined evaluation (Fig. 9) shows improvement from 0.728 (m_1, own-round combined) to 0.898 (m_4, own-round combined) — a 17.0 percentage point improvement in realistic mixed-threat conditions over four rounds.

7. Discussions and Findings

(1) Clean accuracy is maintained at operationally acceptable levels throughout cyclic training. Across four rounds of LLM-adversarial augmentation, clean accuracy stays within a 1.8 percentage point band (0.978–0.996). The dominant cost is a modest increase in false positives at Rounds 3–4, where the model has absorbed phishing-style LLM rewrites and applies a slightly more conservative classification boundary for benign emails. Recall improves from Round 1 to Round 4 (0.992 $\rightarrow$ 0.996), meaning the model misses fewer phishing emails as training progresses. For security-sensitive deployments where missed phishing attacks carry higher cost than false alarms, this trade-off is operationally favourable.

(2) Adversarial attack surfaces shrink consistently, confirming progressive robustness. The number of emails that any held-out attacker can successfully perturb falls by 71–85% across four rounds. This is a direct measure of how much the model's exploitable decision surface has shrunk. The reduction is attacker-consistent: TextFooler (strongest), PWWS (intermediate), and DeepWordBug (weakest) all show the same qualitative trend, indicating that the improvement is a general property of the trained representations rather than an artefact of any single attack type.

(3) LLM-cyclic training creates measurable cross-attacker interference at later rounds. The m_4 vs. Round 3 TextFooler anomaly (0.435 adversarial-only accuracy) reveals that training the defender against LLM-style rewrites can inadvertently create new vulnerabilities to orthogonal attack strategies. Specifically, the LLM rewrite training shifts the model boundary in a way that intersects with the synonym-substitution landscape that TextFooler exploits. This cross-attacker interference effect has direct implications for system design: when training against a single attack family (LLM rewrites), the defender cannot guarantee maintained robustness against all other attack types.

(4) Combined realistic-threat accuracy improves substantially across all attackers. When evaluation is conducted on the realistic mixture of clean emails and adversarial examples, mixed-threat accuracy improves by 15.8 percentage points for PWWS, 17.0 percentage points for TextFooler, and is maintained at above 0.952 for DeepWordBug across all rounds. This is the metric most relevant to deployment: a real inbox contains a mixture of clean and adversarially-crafted emails, and it is the combined performance that determines operational efficacy.

(5) Several robustness estimates rely on small held-out pools and should be interpreted cautiously. Because our evaluation framework uses the shrinking set of successful attacks as a robustness measure, later rounds and stronger attackers naturally yield fewer adversarial examples—by Round 3, DeepWordBug produces only $n = 11$ and TextFooler $n = 62$. This design choice, rather than a data limitation, increases uncertainty in individual cross-round cells, most notably m_4 versus Round 3 TextFooler (0.435, 95% CI $\approx$ 0.31–0.56). We report these results because the overall pattern of shrinking attack pools and strong cross-round transfer is consistent, but individual small-pool estimates should not be overinterpreted. Confidence intervals for all cross-round tables remain a priority for future larger-scale replications.

Practical Implications

- *LLM rewriting is an effective and practical training attack:* qwen3:8b generates full-text rewrites that meaningfully challenge the defender while running entirely offline. The shared rewrite cache eliminates redundant LLM calls across configurations.
- *Maintain clean/adversarial diversity:* The Round 3 minimum in adversarial pool size across all attackers suggests that the defender becomes maximally hardened after two full accumulative rounds. Continued training at Round 4 produces a modest rebound in attack surface, indicating potential overfitting to the LLM's rewrite style.
- *Confidence-drop threshold tuning:* The current acceptance gate ($\tau = 0.0$) admits all rewrites that do not increase defender confidence. Raising τ would admit only rewrites that actively reduce model confidence, producing a stronger adversarial signal per round and potentially reducing the Round 4 rebound effect.

8. Challenges

During the experimental process, several challenges were encountered that influenced the efficiency and scalability of the proposed framework.

- **LLM throughput bottleneck:** The LLM attacker (`qwen3:8b`) runs locally and is the dominant throughput constraint. Each rewrite requires a separate Ollama API call, and the request timeout (300 s) means that a

single round over 2,350 emails can take several hours even with caching. The shared rewrite cache across configurations alleviates this partially but does not eliminate the per-round cost for new emails.

- **TextAttack GPU incompatibility:** The held-out evaluators (TextFooler, PWWS, DeepWordBug) use the TextAttack library, which cannot efficiently utilise GPU acceleration. TextFooler's Universal Sentence Encoder constraint runs on CPU-bound TensorFlow, and PWWS's word-saliency loop is entirely CPU-based. This constrained the number of adversarial evaluation examples that could be generated within practical time budgets and is why the adversarial pool sizes in Table 4 remain relatively small relative to the full evaluation set.
- **Dataset growth and the accumulation trade-off:** The accumulative training strategy (Rounds 1–3) progressively grows the training set as rewrites from prior rounds are retained. While this provides richer adversarial coverage, it increases training time and GPU memory requirements per round. It also raises the risk of the defender overfitting to a specific LLM rewrite style, as evidenced by the cross-attacker interference observed at Round 4.
- **Limited scale:** The current experiments use 6% of the CEAS_08 dataset (≈2,350 emails), a smoke-test setting chosen for computational feasibility. Results at this scale may not fully generalise to the full dataset (39,154 emails), where adversarial pool diversity and training data coverage are substantially larger.

9. Future Works

Future research can address current limitations and extend the scope of this study:

- **Model C (both-labels) evaluation:** Future work should test whether rewriting both classes reduces spurious reliance on LLM-rewrite style.
- **Confidence-drop threshold tuning:** Experimenting with values of $\tau > 0.0$ for the rewrite acceptance gate would allow control over how adversarial rewrites must be before entering the training pool. Higher τ values should reduce dataset bloat, improve adversarial signal quality, and potentially mitigate the Round 4 rebound in attack surface size.
- **GPU-compatible adversarial evaluation:** Replacing the TextAttack USE-based constraint in TextFooler with a GPU-resident sentence encoder, and porting PWWS's saliency loop to GPU, would remove the held-out evaluation bottleneck. This would allow larger adversarial evaluation sets and more statistically robust cross-round comparisons.

References

Al-Subaiey, A., Al-Thani, M., Abdullah Alam, N., Antora, K. F., Khandakar, A., and Uz Zaman, S. A. Novel interpretable and robust web-based ai platform for phishing email detection. *Computers and Electrical Engineering*, 120:109625, December 2024. ISSN 0045-7906. doi: 10.1016/j.compeleceng.2024.109625. URL http://dx.doi.org/10.1016/j.compeleceng.2024.109625.

Gao, J., Lanchantin, J., Soffa, M. L., and Qi, Y. Black-box generation of adversarial text sequences to evade deep learning classifiers. *CoRR*, abs/1801.04354, 2018. URL http://arxiv.org/abs/1801.04354.

Gholampour, P. M. and Verma, R. M. Adversarial robustness of phishing email detection models. In *Proceedings of the 9th ACM International Workshop on Security and Privacy Analytics (IWSPA '23)*, pp. 67–76, Charlotte, NC, USA, 2023. doi: 10.1145/3579987.3586567.

Jin, D., Jin, Z., Zhou, J. T., and Szolovits, P. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment, 2020. URL https://arxiv.org/abs/1907.11932.

Lee, J., Xin, Z., Ng Pei See, M., Sabharwal, K., Apruzzese, G., and Divakaran, D. M. Attacking logo-based phishing website detectors with adversarial perturbations, 2023. URL https://arxiv.org/abs/2308.09392.

Morris, J. X., Lifland, E., Yoo, J. Y., Grigsby, J., Jin, D., and Qi, Y. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP, 2020. URL https://arxiv.org/abs/2005.05909.

Ren, S., Deng, Y., He, K., and Che, W. Generating natural language adversarial examples through probability weighted word saliency. In Korhonen, A., Traum, D., and Màrquez, L. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1085–1097, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1103. URL https://aclanthology.org/P19-1103/.

Uddin, M. A., Islam, M. N., Maglaras, L., Janicke, H., and Sarker, I. H. Explainable detector: Exploring transformer-based language modeling approach for SMS spam detection with explainability analysis. *Digital Communications and Networks*, 2025. URL https://www.sciencedirect.com/science/article/pii/S235286482500118X.

A. Appendix: Adversarial Pool Size Across Rounds

Table 5 provides the complete adversarial pool composition for all three held-out attackers across all four rounds, including the combined (clean eval + adversarial) dataset sizes and the adversarial share of each combined set. These figures underpin the shrinking-pool analysis reported in Section 6.

Table 5. Full adversarial dataset composition for Model B across all four rounds and three held-out attackers.

Attacker	Round	Clean rows	Adv. rows	Combined	Adv. share
PWWS	1	449	161	610	0.264
	2	449	127	576	0.220
	3	449	24	473	0.051
	4	449	46	495	0.093
TextFooler	1	449	165	614	0.269
	2	449	112	561	0.200
	3	449	62	511	0.121
	4	449	42	491	0.086
DeepWordBug	1	449	57	506	0.113
	2	449	39	488	0.080
	3	449	11	460	0.024
	4	449	14	463	0.030

B. Appendix: Cross-Round Accuracy Tables

Tables 6–11 reproduce the heatmap values from Section 6 in tabular form for reproducibility.

Table 6. DeepWordBug: adversarial-only cross-round accuracy. Row = model round. Column = adversarial dataset source round. Diagonal = own-round (0.000).

Model	R1 adv	R2 adv	R3 adv	R4 adv
m_1	0.000	0.821	1.000	0.857
m_2	0.789	0.000	0.909	1.000
m_3	0.877	0.846	0.000	0.786
m_4	0.930	0.795	0.727	0.000

Table 7. DeepWordBug: combined (clean eval + adversarial) cross-round accuracy.

Model	Clean	R1 comb	R2 comb	R3 comb	R4 comb
m_1	0.996	0.883	0.982	0.996	0.991
m_2	0.991	0.968	0.912	0.989	0.991
m_3	0.978	0.966	0.967	0.954	0.972
m_4	0.982	0.976	0.967	0.976	0.952

Table 8. PWWS: adversarial-only cross-round accuracy.

Model	R1 adv	R2 adv	R3 adv	R4 adv
m_1	0.000	0.906	0.875	0.957
m_2	0.522	0.000	0.833	0.935
m_3	0.913	0.937	0.000	0.826
m_4	0.870	0.929	0.625	0.000

Table 9. PWWS: combined cross-round accuracy.

Model	Clean	R1 comb	R2 comb	R3 comb	R4 comb
m_1	0.996	0.733	0.976	0.989	0.992
m_2	0.991	0.867	0.773	0.983	0.986
m_3	0.978	0.961	0.969	0.928	0.964
m_4	0.982	0.952	0.970	0.964	0.891

Table 10. TextFooler: adversarial-only cross-round accuracy. Anomalous cell: m_4 vs. R3 (0.435).

Model	R1 adv	R2 adv	R3 adv	R4 adv
m_1	0.000	0.929	0.887	0.976
m_2	0.558	0.000	0.677	0.929
m_3	0.909	0.911	0.000	0.833
m_4	0.867	0.875	0.435	0.000

Table 11. TextFooler: combined cross-round accuracy.

Model	Clean	R1 comb	R2 comb	R3 comb	R4 comb
m_1	0.996	0.728	0.982	0.982	0.994
m_2	0.991	0.875	0.793	0.953	0.986
m_3	0.978	0.959	0.964	0.859	0.965
m_4	0.982	0.951	0.961	0.916	0.898

C. Appendix: Rewrite Quality Summary

The rewrite quality gate scores every LLM-generated candidate before it is admitted to the training pool. Table 12 defines the diagnostics computed per rewrite. All metrics are stored in `round_N/rewrite_quality.csv` and their aggregate statistics in `round_N/rewrite_quality_summary.json`.

Table 12. Rewrite quality metrics computed per LLM-generated candidate.

Metric	Description
`confidence_drop`	Source model confidence on true label minus rewrite confidence. Gate criterion: $\geq \tau$.
`changed`	Whether normalised source text differs from rewrite.
`char_similarity`	SequenceMatcher character-level ratio (0–1).
`word_jaccard`	Jaccard overlap of lowercase word sets.
`length_ratio`	|rewrite|/|source| — detects truncation or expansion.
`url_behavior_preserved`	True iff the set of http/https/www URLs is identical.
`has_non_english_script`	True if rewrite contains non-English character ranges.
`looks_like_structured_output`	True if rewrite begins with {, `[`, ` ``` `, or contains `key:` patterns (LLM format leak).

D. Appendix: Repository and Reproducibility

All experiments are managed through the `mail_cag.py` entrypoint at the repository root. The complete reproduction sequence from a clean environment is:

1. **Bootstrap:** `scripts/bootstrap.sh` creates the conda/venv environment (`ENV_NAME=nlp_game`, `PYTHON_VERSION=3.12`) and installs all dependencies.
2. **Download models:** `scripts/download_required_models.sh` pulls `qwen3:8b`, `qwen3:14b`, `qwen3:4b` via Ollama and downloads `albert-base-v2` from HuggingFace into the local cache.
3. **Prepare dataset:** `python scripts/prepare_ceas.py` applies the four-stage filter pipeline (text assembly → language detection → script filter → token-length cap at 1600) and writes `data/processed/CEAS_08_en_1600.csv`.
4. **Dry-run:** `python mail_cag.py run model-b --dry-run` validates the config and data split without training or LLM calls.
5. **Run:** `python mail_cag.py run model-b --run-id b_exp_001`
6. **Evaluate:** `python mail_cag.py evaluate model-b --run-id b_exp_001`

All generated outputs are isolated under `runs/<experiment>/<run_id>/` and excluded from version control. The experiment can be safely interrupted and resumed at any point with `--resume`.