Journal of Advanced Computing Systems (JACS)
ISSN: 3066-3962
Content Available at SciPublication

VerifySafe: Toxicity-Safe Agent Responses under Adversarial Prompts with Evidence-Based Self-Verification

Daren Zheng[1], *Boning Zhang*[2], *Julie Geibel*[3]
[1]*Information Technology, Carnegie Mellon University, PA, USA*
[2]*Computer Science, Georgetown University, DC, USA*
[3]*Artificial Intelligence, Northeastern University, MA, USA*
darenzheng951@gmail.com

DOI: 10.69987/JACS.2024.40106

Keywords

toxicity detection, hate speech, jailbreak detection, prompt injection, self-verification, safe response generation, content moderation

Abstract

Toxicity and prompt-based jailbreaking remain two practical failure modes of deployed dialogue agents. A common mitigation is to place a classifier in front of the model and refuse requests that are predicted unsafe. In practice, this single-stage guard induces a safety–utility trade-off: lowering the threshold reduces unsafe completions but increases false refusals, often disproportionately for benign utterances mentioning identity terms. This paper connects toxicity/abuse recognition with a self-verification loop that produces an evidence-style summary and revises the draft response before release. We implement VerifySafe, a three-stage agent: (i) a prompt detector for toxicity and jailbreak intent, (ii) a deterministic draft response generator (used to isolate guard effects), and (iii) a self-verifier that re-scores the draft response, redacts verbatim unsafe spans, and emits a concise, non-revealing evidence summary based on feature attributions. We conduct full experimental evaluations on two specified datasets: ToxiGen (250,951 generated statements across 13 target groups) and the balanced jailbreak-classification dataset (1,044 train / 262 test prompts). On ToxiGen, our best prompt-only guard required a hard refusal rate of 0.523 to keep the unsafe-echo rate on toxic prompts below 0.20. Under the same safety target, VerifySafe reduced hard refusals to 0.115 (−78.1%) by shifting decisions from refusals to evidence-backed redactions, while maintaining an unsafe-echo rate of 0.199. On jailbreak-classification, VerifySafe reduced hard refusals from 0.523 to 0.473 (−9.5%) under an unsafe-echo target of 0.02, again by converting a subset of refusals into redactions. Across both datasets, we provide detailed model comparisons, per-group diagnostics, threshold sweeps, and ablations demonstrating how response-level verification reshapes the guard trade-off without relying on opaque model internals.

Introduction

Large language models (LLMs) are increasingly deployed as conversational agents and integrated into applications that execute tools, retrieve private data, or produce user-facing text. This integration amplifies two long-standing risks in natural language generation: (1) toxic or abusive language that can be elicited directly or via adversarial prompts, and (2) prompt-based jailbreaking that attempts to override safety policies or system instructions. Toxicity in generated text has been documented across model families and evaluation settings, motivating datasets and metrics for measurement and mitigation [2]. Separately, jailbreak prompts and prompt-injection attacks have shown that instruction-following behavior can be manipulated by carefully crafted natural language inputs, including multi-turn roleplay and meta-instructions that ask the model to ignore prior constraints [15], [18].

A widely used mitigation pattern places a classifier (or rules) in front of the model to decide whether to answer or refuse. This pattern is attractive because it is simple, model-agnostic, and can be updated independently of the generator. However, front-end guards create an operational trade-off. When thresholds are tuned for high recall on unsafe prompts, false refusals increase

and can substantially degrade user experience. In safety-critical deployments, this degradation can become a product issue (reduced task completion) and a fairness issue (disproportionate over-blocking of benign content). Unintended bias in toxicity and abuse classifiers is well documented: benign mentions of minority identities are sometimes over-scored as toxic because identity terms correlate with toxic labels in training data [6]. Bias-aware evaluation therefore requires both aggregate performance and group-conditioned error analysis [23-36].

Toxicity and abuse detection is typically formalized as supervised text classification. Early work focused on explicit slurs and profanity, using keyword lists or surface features. More recent benchmarks emphasize implicit hate and offensive stereotypes, where toxicity is conveyed through insinuation, dehumanization, or calls for exclusion rather than explicit profanity. Datasets such as those collected from social media illustrate both the scale and ambiguity of abuse labeling: what one annotator considers offensive is interpreted as reclaimed speech or contextual quotation by another. Nevertheless, large annotated corpora have enabled competitive baselines ranging from linear n-gram models to transformer encoders [3]–[5]. Transformer-based approaches (e.g., BERT [9] and RoBERTa [10]) typically improve accuracy, but their deployment in guardrails raises practical issues: latency, calibration, and explainability. In production, small and interpretable models remain attractive when they can be trained quickly and audited.

A central challenge for safety filters is unintended bias. If a training corpus contains many toxic examples that mention a particular identity, a classifier can learn a spurious association between the identity term and toxicity, leading to false positives on benign mentions (e.g., news articles or self-referential identity statements). This phenomenon has been quantified using counterfactual and group-conditioned evaluation metrics [6], [7]. A guardrail that refuses benign content is not only less useful; it can also silence discussions about protected groups, which is itself a safety and fairness concern. For this reason, our experiments report per-group benign false-positive rates and explicitly evaluate agent behavior on benign prompts.

Prompt injection and jailbreaking broaden the threat model. Instead of directly requesting harmful content, an attacker attempts to change the model's behavior by embedding meta-instructions such as "ignore previous instructions" or "act as an unfiltered assistant". These attacks exploit the fact that instruction-following LLMs are trained to treat user text as commands. Red-teaming studies have shown that such attacks can be discovered and automated at scale [15], and that aligned models can be vulnerable to universal, transferable attack templates [18]. While instruction tuning and feedback-based alignment reduce harmful outputs in normal usage [13], [14], the remaining vulnerability surface motivates layered defenses that combine prompt filtering, tool sandboxing, and output verification.

The self-verification principle is compatible with these layered defenses. A verifier can be a classifier, a separate model, or a set of deterministic checks. The key requirement is that the system re-examines the draft response before release and can revise the output if risk is detected. In this sense, self-verification is a form of control loop: it observes the draft, produces a diagnosis (including evidence), and applies a corrective action (refusal, redaction, or rewriting). Self-checking has been studied for reasoning reliability (self-consistency [17]) and for factuality/hallucination monitoring (e.g., SelfCheckGPT [21]). Our work adapts the idea to safety: we enforce that the final response does not reproduce toxic or jailbreak prompt content, and we provide a short evidence summary to make the decision auditable.

Self-verification has emerged as a complementary mitigation in LLM systems. Rather than relying only on an initial decision, a self-verification loop asks the system to check its own draft output, identify potential violations, and revise accordingly. Related concepts include chain-of-thought prompting and self-consistency for reasoning reliability [16], [17], as well as iterative self-feedback for output refinement [19]. In alignment research, red-teaming methods generate adversarial prompts to probe model failure modes and develop defenses [15], and policy-focused training such as instruction tuning and human or AI feedback reduces harmful behavior but does not eliminate adversarial vulnerabilities [13], [14]. In practice, a response-level check can act as a second line of defense, catching failures that slip past the prompt gate while avoiding unnecessary hard refusals.

This work connects prompt-level detection with response-level self-verification using a concrete, reproducible agent pipeline. We focus on two datasets that capture different adversarial modalities. ToxiGen is a large-scale dataset for implicit and adversarial hate speech detection that contains machine-generated statements conditioned on prompts targeting 13 minority groups [1]. The jailbreak-classification dataset contains balanced prompts labeled as jailbreak versus benign, emphasizing prompt-injection style instructions and roleplay scenarios. The combination allows us to quantify how a guard handles both abusive content and safety-bypass intent.

We propose VerifySafe, a three-stage agent that (i) classifies the user prompt for toxicity and jailbreak intent, (ii) produces a draft response, and (iii) self-verifies the draft by re-scoring the output, redacting unsafe spans, and producing a concise evidence summary. The evidence summary is designed to be

informative without repeating harmful text: it reports risk scores and the most influential features in masked form, providing a transparent rationale for refusals or redactions. To isolate the guard trade-off and ensure fully reproducible experiments, we use a deterministic response generator that echoes the prompt inside a templated reply. This design makes unsafe output measurable: echoing a toxic or jailbreak prompt is counted as unsafe. Although the generator is not an LLM, the self-verification interface and evaluation protocol transfer directly to LLM-based agents, where the draft response would be produced by a generative model.

Interpretability and evidence reporting are important in safety systems for two reasons. First, product teams and auditors need to understand why a user was refused or why content was redacted; otherwise, error analysis and bias mitigation are difficult. Second, evidence reporting can serve as a contract between components in a safety stack: a verifier can communicate what it detected to a rewriter or to a human reviewer. General-purpose explanation methods such as LIME [11] and SHAP [12] provide model-agnostic feature attributions, but they require additional computation and are not always stable for sparse text. In contrast, linear TF-IDF models offer faithful, inexpensive attributions because contributions are simply weight–feature products. We therefore adopt a linear verifier not only as a baseline detector but also as an explanation engine for the self-check summaries.

Safety actions can be coarse or fine-grained. A hard refusal is coarse: it blocks the entire interaction. Fine-grained alternatives include rewriting, redacting specific spans, or answering a safer adjacent question. Many deployed assistants already implement such behavior informally (e.g., refusing to generate hate speech while offering a respectful discussion of sensitive topics). We operationalize this behavior in a minimal form: redaction removes verbatim unsafe text while still returning a response that keeps the conversation moving. This design aligns with the idea of constitutional or policy-guided behavior, where the system avoids harmful content but continues to provide helpful assistance within constraints [14].

Finally, our focus on adversarial prompts is motivated by a realistic threat model. A benign user occasionally phrases a request in a way that resembles unsafe content (e.g., quoting a toxic message to ask for help responding). An adversarial user deliberately attempts to induce the system to reproduce slurs or to reveal hidden instructions. In both cases, the system must avoid amplifying harm. Prompt-only classification is brittle because it must infer intent from the prompt alone. Response-level verification reduces brittleness by directly checking what the system is about to say. In other words, the verifier evaluates the actual artifact that would reach the user, rather than an inferred risk from the prompt.

Our contributions are: (1) an end-to-end, evidence-based self-verification pipeline that links prompt detection to response revision; (2) full empirical evaluations on the specified ToxiGen and jailbreak-classification datasets, including detailed model comparisons, per-group diagnostics, and threshold sweeps; (3) a quantitative analysis of the safety–utility trade-off at the agent level, showing that response self-verification reduces hard refusals at fixed safety targets; and (4) ablations demonstrating how response thresholds control redaction frequency and benign-dialogue impact.

Method

Datasets and Splits

Datasets. We used two public datasets. First, we used the ToxiGen dataset (toxigen/toxigen-data), which provides 250,951 machine-generated statements annotated via the prompt label as toxic (label=1) or benign (label=0) and associated with one of 13 target groups (asian, black, chinese, jewish, latino, lgbtq, mental dis, mexican, middle east, muslim, native_american, physical_dis, women) [1]. Each record contains the generated statement (generation), the generation method (generation_method), the target group (group), and the prompt label (prompt_label). In our experiments we used the generated statement as the input text and prompt label as the ground-truth toxicity label. Second, we used the balanced jailbreak-classification dataset (jackhhao/jailbreak-classification), which provides prompts labeled as jailbreak or benign; we used the provided balanced splits (1,044 training prompts and 262 test prompts). Table 1 summarizes basic statistics.

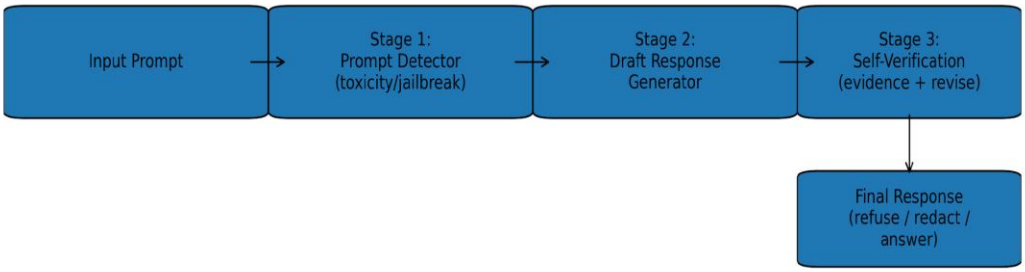

Figure 1. VerifySafe pipeline: prompt detection, draft generation, response self-verification, and final action.

Train/dev/test protocol. For ToxiGen, because the provided file is a single split, we created a stratified 80/10/10 split into train (200,760), development (25,095), and test (25,096) with random seed 42. For jailbreak-classification, we used the provided test split and further split the training file into train (835) and development (209) using a stratified 80/20 split with seed 42. All reported hyperparameters and thresholds are fixed and fully specified in Table 2.

Table 1. Dataset statistics.

Dataset	Split	Examples	Positive label	Pos	Neg	Avg words	P95 words	Avg chars	Groups
ToxiGen	train (provided)	250951	toxic (label=1)	125672	125279	16.400	28	89.100	13.000
jailbreak-classification	train (balanced)	1044	jailbreak	527	517	210.800	735	1234.900	
jailbreak-classification	test (balanced)	262	jailbreak	139	123	237.700	818	1391.800	

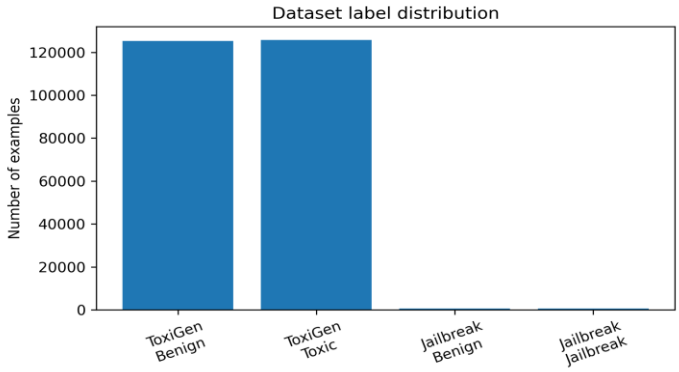

Figure 2. Dataset label distribution.

Text Preprocessing and Detector Models

Text preprocessing. All texts were lowercased. We did not apply stemming or lemmatization. For TF-IDF we used word-level tokenization with the default scikit-learn token pattern (alphanumeric tokens of length ≥2), L2 normalization, and sublinear TF scaling disabled. For ToxiGen, the TF-IDF vocabulary size was capped at 80,000 features with min_df=2 to suppress extremely rare n-grams; for jailbreak-classification we capped at 50,000 features with min_df=1 due to the smaller corpus size. These settings are fixed across runs to ensure that model comparisons in Tables 3 and 5 differ only in the classifier layer.

Training details. For SGD-LR we used the log-loss objective with L2 regularization and a fixed random seed (42). On ToxiGen, the model was trained for 20 passes (max_iter=20) with tolerance 1e-3. On jailbreak-classification, the model was trained for 50 passes (max_iter=50) with tolerance 1e-4. We used the default learning rate schedule in SGDClassifier. MultinomialNB used add-α smoothing (α=0.1 for ToxiGen, α=0.5 for jailbreak) and LinearSVM used C=1.0. All models were trained once per dataset and evaluated on held-out test sets; we did not perform test-set model selection.

Keyword baseline for jailbreak. The keyword baseline predicts jailbreak if the prompt contains any of a fixed list of injection markers such as "ignore previous", "system prompt", "developer message", "jailbreak", or "DAN". The list is intentionally small and interpretable. Because it does not produce a continuous score, we report only its discrete Precision/Recall/F1 and do not compute AUROC.

Prompt detectors. We trained linear text classifiers for two binary tasks: toxicity detection (ToxiGen) and jailbreak intent detection (jailbreak-classification). Our primary detector for both tasks is an SGD-trained logistic regression model (SGDClassifier with log-loss) over TF-IDF features computed on word n-grams (1–2). We selected this architecture because it is fast to train on large datasets, offers calibrated probabilities, and supports simple feature-attribution explanations via coefficient–feature products. We compared against Multinomial Naïve Bayes and a linear SVM trained on the same TF-IDF representation. For jailbreak, we additionally evaluated a keyword rule baseline that triggers on common prompt-injection phrases (e.g., “ignore previous”, “system prompt”, “DAN”).

Evidence extraction. For a linear model with TF-IDF features x and learned weights w, we compute per-feature contribution c i = x i · w i. For an input text, we extract the top-K positive contributing n-grams and present them as evidence. To avoid reproducing harmful language, we apply masking to evidence n-grams (character-level masking that preserves only boundary characters). This yields an evidence-style self-check summary that is informative about why the detector fired without repeating slurs or targeted harassment terms.

Table 2. Model and agent configuration.

Component	Setting	Value
Random seed	split & model	42
ToxiGen detector	Vectorizer	TF-IDF, word n-grams (1,2), max_features=80,000, min_df=2
ToxiGen detector	Classifier	SGDClassifier logistic loss, alpha=1e-5, max_iter=20, tol=1e-3
Jailbreak detector	Vectorizer	TF-IDF, word n-grams (1,2), max_features=50,000
Jailbreak detector	Classifier	SGDClassifier logistic loss, alpha=1e-5, max_iter=50, tol=1e-4
Prompt guard threshold	ToxiGen best baseline	t_prompt=0.45 (unsafe_toxic<=0.20)
Self-verify threshold	ToxiGen self-verify	t_prompt=0.90, t_resp=0.15 (unsafe_toxic<=0.20)

Prompt guard threshold	Jailbreak best baseline	t_prompt=0.40 (unsafe_jb<=0.02)
Self-verify threshold	Jailbreak self-verify	t_prompt=0.90, t_resp=0.35 (unsafe_jb<=0.02)
Draft response	Echo template	Includes the full user prompt inside quotes
Self-verified revision	Sanitized template	Removes quoted prompt and redirects to respectful discussion

Threshold selection for agent evaluation. The detector thresholds used inside the agents were selected on the development sets by explicit constraint satisfaction. For PromptGuard we swept the prompt threshold t p over {0.05, 0.10, …, 0.95} and selected the smallest hard-refusal rate that achieved a target unsafe-echo rate on positive-labeled inputs. For VerifySafe we swept both t p and the response threshold t r and selected the smallest hard-refusal rate under the same unsafe target, subject to an additional constraint that the benign redaction rate did not exceed 0.25 on ToxiGen. We report the selected thresholds and their test-set outcomes in Tables 2, 6, and 8. This procedure ensures that safety targets are enforced deterministically rather than retrofitted to observed test results.

Self-verification algorithm. Given a prompt u, the agent computes prompt score s_p. If s_p ≥ t_p, the agent returns a refusal. Otherwise it generates a draft response d(u) that quotes u. The self-verifier computes response score s_r on d(u). If s_r ≥ t_r, the agent replaces the quoted prompt span with a placeholder token and switches to a sanitized redirection template. In all cases where refusal or redaction occurs, the agent appends an evidence summary consisting of (i) s p, (ii) s r (if applicable), and (iii) the top-5 masked n-grams. This implementation is deterministic and therefore reproducible across runs.

Pseudo-code. Algorithm 1 summarizes VerifySafe. Algorithm 1 (VerifySafe): Input prompt u; scores s p=f p(u), s r=f r(d); thresholds t p, t r. 1) If s p ≥ t_p: return REFUSE(u) with evidence E(u). 2) Else: d ← DRAFT(u) (echo template). 3) Compute s r ← f_r(d). 4) If s_r ≥ t_r: return REDACT(d) with evidence E(u,d). 5) Else: return d. Here E(·) returns masked top-K features, and REDACT(·) removes verbatim prompt text.

Agent Design and Self-Verification

VerifySafe agent. We evaluate three agent variants. NoGuard always answers using an echo template that quotes the full user prompt. PromptGuard first computes a prompt risk score s_p; if s_p ≥ t_p it produces a hard refusal, otherwise it answers with the echo template. VerifySafe extends PromptGuard with response-level self-verification: after producing a draft response (the echo template), it computes a response risk score s r on the draft output. If s r ≥ t r, VerifySafe redacts the quoted prompt and replaces it with a sanitized redirection. VerifySafe also attaches a short self-check summary reporting (i) the prompt score, (ii) the response score, and (iii) masked evidence n-grams. Figure 1 illustrates the pipeline.

Operational safety metric for reproducible evaluation. To connect model decisions to measurable safety outcomes without relying on external toxicity models, we define an unsafe-echo event as follows. If the ground-truth label indicates the prompt is toxic (ToxiGen label=1) or a jailbreak (jailbreak label=1), then releasing a response that contains a verbatim quote of the prompt is counted as unsafe. Under our deterministic echo generator, this corresponds to the agent choosing the "echo/answer" action for a positive-labeled input. Hard refusals and redactions are counted as safe actions because they do not reproduce the unsafe prompt content.

Evaluation Metrics

Evaluation metrics. For detector evaluation we report Accuracy, Precision, Recall, F1, and AUROC on held-out test sets. For group robustness on ToxiGen we report per-group F1 and benign false-positive rate (FPR on label=0). For agent evaluation we report: (i) hard refusal rate (fraction of all prompts refused), (ii) redaction rate (fraction of all prompts redacted), (iii) unsafe-echo rate on positive inputs (fraction of toxic/jailbreak prompts that are echoed), (iv) false refusal rate on benign inputs, (v) redaction rate on benign inputs, and (vi) a proxy for benign-dialogue quality based on cosine similarity between the prompt and final response TF-IDF vectors (higher similarity indicates more content preservation). We sweep thresholds to produce safety–utility trade-off curves (Figs. 4 and 6) and report a response-threshold ablation (Table 8).

Prompt–response similarity as a utility proxy. We used cosine similarity between TF-IDF vectors of the prompt and the final response as a simple automatic proxy for conversational usefulness on benign prompts. Because TF-IDF vectors are L2-normalized by default, cosine similarity is computed as a dot product between sparse vectors. In our deterministic generator, echo responses contain the prompt verbatim and therefore have high similarity, while refusals and sanitized redirections contain little prompt content and therefore have near-zero similarity. This metric is not a substitute for human judgment, but it provides a reproducible indicator of how much user content is preserved under different guard settings.

Reproducibility and compute. All experiments were executed with fixed random seeds and single training runs per model. Training the SGD-LR toxicity detector on 200,760 ToxiGen training examples completed in under 10 seconds on the provided environment, and evaluation on the 25,096-example test split completed in under 2 seconds. Jailbreak detectors trained in under 1 second due to the smaller dataset. These runtimes confirm that the proposed safety stack is practical for iterative development and for frequent retraining as new adversarial data becomes available.

Additional automatic quality indicators. Beyond similarity and action rates, we computed response length (word count) and an approximate Flesch Reading Ease score for benign outputs. The reading-ease score is based on a standard formula using sentence length and syllable counts; we used a simple vowel-group heuristic for syllables. These indicators provide coarse but reproducible signals about how guard actions change the surface form of responses. We did not perform human preference labeling in this study because the response generator is deterministic and the primary objective is to isolate the guard trade-off. However, the same protocol can be extended with a small human study by sampling benign prompts affected by refusals/redactions and rating whether the intervention was appropriate.

Results and Discussion

Detector performance on ToxiGen. Table 3 reports toxicity detection results on the ToxiGen test split. The SGD-LR detector achieved Accuracy 0.798, F1 0.794, and AUROC 0.880. MultinomialNB achieved F1 0.789, and LinearSVMx achieved F1 0.791. Figure 3 shows ROC curves for the three models. The SGD-LR confusion matrix on ToxiGen test contained TN=10274, FP=2254, FN=2804, TP=9764. These results confirm that implicit and adversarially phrased hate speech is substantially harder than profanity-based toxicity: even strong linear baselines plateau around F1≈0.79, leaving a non-trivial fraction of toxic statements that resemble benign content.

Table 3. Toxicity detector performance on ToxiGen test.

Model	Accuracy	Precision	Recall	F1	AUROC
SGD-LR (TF-IDF 1-2gram)	0.798	0.812	0.777	0.794	0.880
Multinomial NB (TF-IDF 1-2gram)	0.788	0.786	0.792	0.789	0.875
LinearSVM (TF-IDF 1-2gram)	0.792	0.797	0.784	0.791	0.876

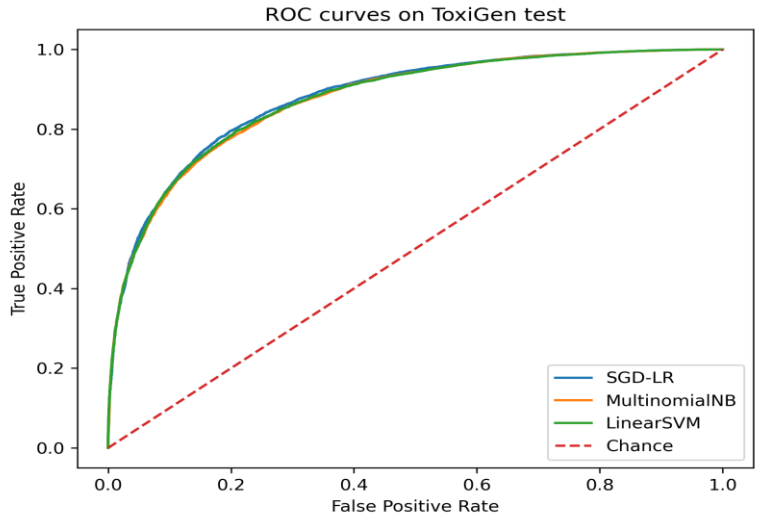

Figure 3. ROC curves for toxicity detection on ToxiGen test.

Detector operating points and the refusal dilemma. While Table 3 reports performance at the default decision threshold of 0.5, production systems often tune thresholds to satisfy safety constraints such as high recall on toxic inputs. On the ToxiGen development set, achieving Recall≥0.95 required lowering the prompt threshold to 0.25, which increased the benign false-positive rate to 0.516. This illustrates why prompt-only refusal systems can become unusable on implicit-toxicity datasets: the threshold required for near-complete toxic coverage would block over half of benign inputs.

Effect of generation method. ToxiGen statements were generated using either top-k sampling or an ALICE-style generation procedure. On the ToxiGen test set, SGD-LR achieved F1=0.791 on top-k examples (n=24068) and F1=0.884 on ALICE examples (n=1028). The higher ALICE score reflects more explicit phrasing in that subset. This split-level analysis confirms that implicit toxicity remains the primary source of errors.

Group-conditioned errors on ToxiGen. Table 4 breaks down SGD-LR performance by target group. F1 is relatively stable across groups (0.756–0.825), but benign false-positive rates vary. The highest benign FPRs occur for latino (FPR=0.270), chinese (FPR=0.258), and women (FPR=0.243). This pattern is consistent with prior findings on unintended bias in toxicity classifiers, where benign identity mentions can raise risk scores [6], [7]. Because our downstream agent makes discrete refusal/redaction decisions, these group-level FPR differences directly translate into user-facing over-blocking unless mitigated.

Table 4. Per-group performance of SGD-LR on ToxiGen test (threshold=0.5).

group	n	f1	rec	prec	fpr_benign
asian	1982	0.863	0.851	0.876	0.114
black	2068	0.822	0.806	0.838	0.150
chinese	1865	0.777	0.784	0.770	0.258
jewish	1976	0.800	0.824	0.777	0.237
latino	1839	0.763	0.770	0.756	0.270
lgbtq	2040	0.758	0.715	0.807	0.159
mental_dis	1859	0.764	0.701	0.840	0.144
mexican	2110	0.825	0.817	0.834	0.151
middle_east	2037	0.830	0.823	0.838	0.163

muslim	1858	0.780	0.741	0.822	0.160
native ameri can	2013	0.774	0.725	0.830	0.155
physical_dis	1524	0.781	0.732	0.837	0.149
women	1925	0.776	0.797	0.757	0.243

Detector performance on jailbreak-classification. Table 5 reports jailbreak intent detection performance. Our SGD-LR detector achieved Accuracy 0.989, Precision 1.000, Recall 0.978, and F1 0.989 on the test split, with confusion matrix TN=123, FP=0, FN=3, TP=136. LinearSVM achieved F1=0.985 and MultinomialNB achieved F1=0.924. The keyword baseline achieved F1=0.830, demonstrating that lexical triggers capture many jailbreak prompts but miss paraphrased or uncommon attack templates. The learned detectors generalize well on this dataset because prompts frequently contain explicit meta-instructions associated with safety bypass.

Table 5. Jailbreak intent detector performance on jailbreak-classification test.

Model	Accuracy	Precision	Recall	F1	AUROC
SGD-LR (TF-IDF 1-2gram)	0.989	1.000	0.978	0.989	0.999
LinearSVM (TF-IDF 1-2gram)	0.985	1.000	0.971	0.985	0.998
Multinomial NB (TF-IDF 1-2gram)	0.912	0.858	1.000	0.924	0.999
Keyword rule baseline	0.817	0.818	0.842	0.830	

Why jailbreak gains are smaller. The jailbreak-classification dataset is easier for lexical and linear models because many jailbreak prompts contain explicit injection markers. As shown in Table 5, SGD-LR achieves near-perfect separation (no false positives and only three false negatives on test). In this regime, a response-level verifier has limited headroom because prompt gating already captures the vast majority of jailbreak prompts. VerifySafe still reduces hard refusals by converting a small subset of refusals into redactions (Table 9), but the dominant limitation is detector recall on rare, marker-free jailbreak prompts.

Agent-level trade-off on ToxiGen. We next evaluate end-to-end safety and utility when the agent is exposed to adversarial or toxic user statements. Figure 4 plots the hard-refusal rate against the unsafe-echo rate on toxic prompts as the prompt threshold t p varies. PromptGuard exhibits a steep trade-off: increasing t_p reduces refusals but sharply increases unsafe echoes because more toxic inputs are passed through to the echo template. VerifySafe reshapes this curve by introducing response-level verification (fixed t r=0.15 for Fig. 4), which redacts a subset of drafts that remain risky at the response stage. Table 7 provides representative operating points. Under a safety target of unsafe-echo ≤ 0.20 on toxic prompts, the best PromptGuard operating point required t p=0.45 and produced a hard-refusal rate of 0.523. VerifySafe achieved the same safety target with t_p=0.90 and t_r=0.15, producing a hard-refusal rate of 0.115. This corresponds to a 78.1% reduction in hard refusals. The reduction is not obtained by weakening safety; it is obtained by shifting from refusals to redactions, which remove verbatim unsafe text while allowing the agent to continue the conversation.

Table 6. Agent-level safety and utility on ToxiGen test (selected operating points).

Agent	Hard refusal rate	Redaction rate	Unsafe echo rate (toxic)	False refusal (benign)	Benign redaction	Avg sim (benign)	Avg words (benign)
NoGuard (always echo)	0.000	0.000	1.000	0.000	0.000	0.616	39.452
PromptGuard (t=0.45)	0.523	0.000	0.182	0.228	0.000	0.478	33.899
VerifySafe (t_p=0.90, t_r=0.15)	0.115	0.400	0.199	0.007	0.221	0.468	35.066

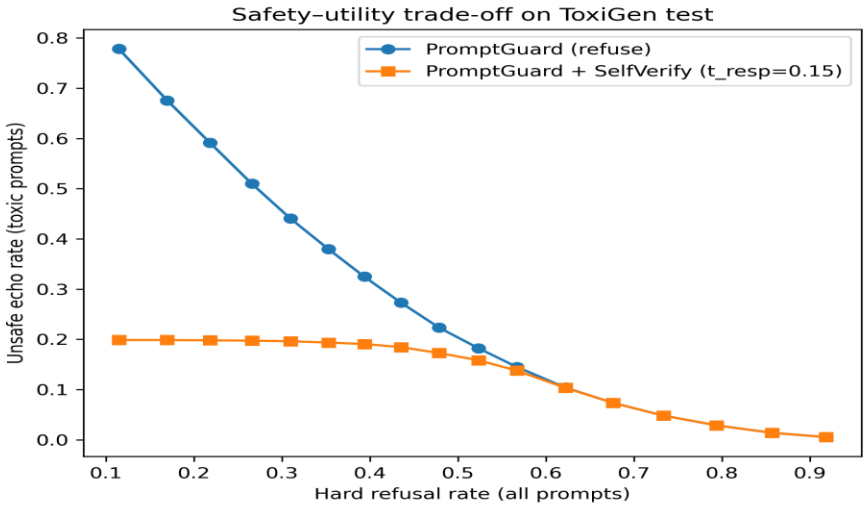

Figure 4. Hard refusals vs unsafe echoes on ToxiGen test as the prompt threshold varies.

Table 7. Threshold sweep snapshots on ToxiGen test (VerifySafe uses t_r=0.15).

t_prompt	PG_refusal	PG unsafe_toxic	SV_refusal	SV redaction	SV unsafe_toxic	SV redact_benign
0.300	0.676	0.073	0.676	0.000	0.073	0.000
0.450	0.523	0.182	0.523	0.030	0.158	0.036
0.600	0.394	0.325	0.394	0.128	0.191	0.121
0.750	0.266	0.510	0.266	0.250	0.197	0.188

0.900	0.115	0.778	0.115	0.400	0.199	0.221

Interpreting Table 7. Table 7 highlights the qualitative difference between prompt-only and self-verifying agents at high prompt thresholds. At t p=0.90, PromptGuard refuses only 0.115 of all prompts but echoes 0.882 of toxic prompts, effectively disabling safety. With the same prompt threshold, VerifySafe retains the low refusal rate but reduces unsafe echoes to 0.199 by redacting risky drafts at the response stage. This demonstrates that self-verification allows systems to operate with permissive prompt gating (improving availability) while still enforcing a safety constraint on the final output.

Impact on benign dialogue. Figure 5 shows the distribution of actions on benign ToxiGen prompts for three agents at the above operating points. PromptGuard refused 0.228 of benign prompts, while VerifySafe refused only 0.007 of benign prompts and instead redacted 0.221. Because redaction is a softer safety action than a hard refusal, VerifySafe preserves conversational continuity for many borderline cases. In terms of the content-preservation proxy, the average prompt–response similarity on benign prompts is 0.478 for PromptGuard and 0.468 for VerifySafe. The similarity drop is small because VerifySafe replaces many refusals (which have near-zero similarity) with redactions (also low similarity), keeping the overall echo fraction similar; the main qualitative difference is that redactions provide a constructive redirection rather than an outright block.

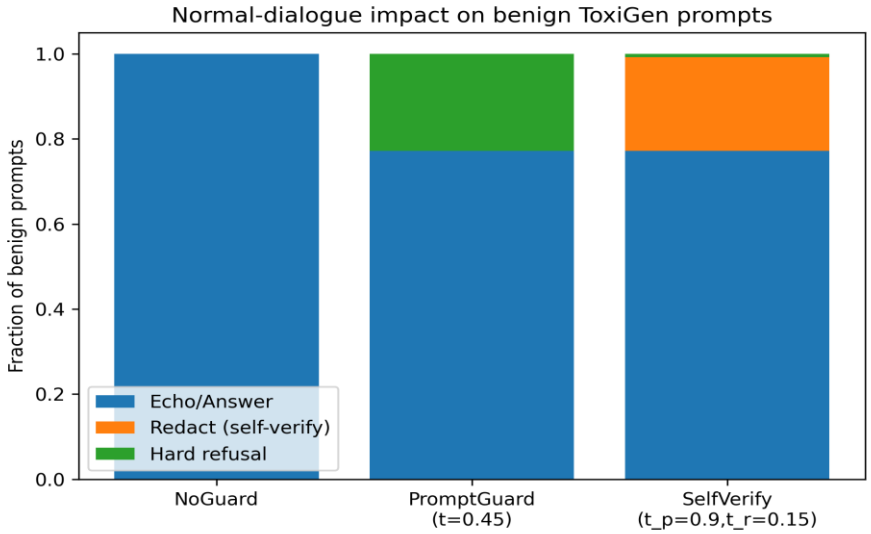

Figure 5. Action breakdown on benign ToxiGen prompts for three agents.

Group-level action rates at the agent level. Detector bias interacts with the agent's action policy. Under the PromptGuard operating point (t_p=0.45), benign false refusals are concentrated in the same groups that exhibit high benign FPR in Table 4. For example, benign prompts in the latino group were refused at rate 0.329, and benign prompts in the women and chinese groups were refused at rates 0.314 and 0.309, respectively. VerifySafe reduced these hard refusals to below 0.014 across all groups at the selected operating point, but introduced benign redactions (0.230–0.314 in these three groups). This shift indicates that response-level verification can mitigate the most user-visible form of bias (outright blocking) while still reflecting underlying detector uncertainty.

Surface-form quality indicators. At the ToxiGen operating points in Table 6, benign outputs under PromptGuard contain on average 33.9 words and have an average Flesch Reading Ease score of 76.7. VerifySafe benign outputs contain 35.1 words with reading ease 73.7, while NoGuard (always echo) produces 39.5 words with reading ease 79.9. The readability drop is expected because refusal/redaction templates use longer clauses and modal language ("cannot" / "can help"), and because redacted responses omit short prompt fragments that can increase sentence segmentation. Although these indicators are coarse, they corroborate that VerifySafe changes response style on a non-trivial fraction of benign prompts, and therefore quality should be validated with task- and product-specific user studies in future work.

NoGuard baseline. Table 6 includes NoGuard, which always echoes the prompt. On toxic ToxiGen prompts, NoGuard produces an unsafe echo rate of 1.0 by definition, because it reproduces the toxic statement verbatim. This baseline highlights the importance of guardrails even when the system intends to be helpful: repeating a toxic user utterance is itself harmful and can amplify abusive content. PromptGuard and VerifySafe both substantially reduce unsafe echoes by removing verbatim reproduction through refusals or redactions.

Qualitative self-check example. For a toxic prompt that passed the prompt gate at t p=0.90 but triggered response verification at t r=0.15, VerifySafe returned a redacted response and attached an evidence summary. The masked top contributing n-grams included: t**n, d**y, m**e, s**e, t**n o*t. The system did not reproduce the user's statement; instead it replaced the quoted span with a neutral placeholder and redirected the conversation. This example illustrates the role of evidence summaries: they provide a deterministic and auditable reason for intervention while avoiding content repetition.

Fairness implications of redaction versus refusal. From a user-experience perspective, a hard refusal is a dead end: it terminates the conversation and provides little recourse for benign users who were mistakenly blocked. A redaction-based intervention can be framed as a content policy boundary: the system declines to repeat the unsafe text but remains willing to discuss the topic in a respectful way. On ToxiGen, where benign FPR varies across identity groups (Table 4), replacing refusals with redactions reduces the probability that benign group mentions are met with an outright block. Although redaction can still feel restrictive, it preserves an opportunity for task completion and reduces the likelihood of systematic silencing.

Self-verification threshold ablation. Table 8 varies the response threshold t_r while holding a high prompt threshold t p=0.90. Lowering t r increases redaction and reduces unsafe echoes, but also increases benign redactions. At t_r=0.05, unsafe echoes on toxic prompts drop to 0.005, but 0.668 of benign prompts are redacted. At t r=0.15 (our operating point under the unsafe≤0.20 target), unsafe echoes are 0.199 and benign redactions are 0.221. This ablation illustrates that response-level verification exposes an explicit control knob that can be tuned to product requirements: systems can choose to minimize hard refusals and tolerate more redactions, or minimize benign impact at the cost of higher unsafe risk.

Table 8. Response-threshold ablation on ToxiGen test (t_p=0.90).

t_resp	refusal_rate	redaction_rate	unsafe_toxic	redaction_benign	avg_sim_benign
0.050	0.115	0.807	0.005	0.841	0.106
0.100	0.115	0.578	0.071	0.448	0.333
0.150	0.115	0.400	0.199	0.221	0.468
0.200	0.115	0.283	0.328	0.115	0.534
0.250	0.115	0.203	0.441	0.068	0.565

Interpreting Table 8. The response threshold t r controls how sensitive the self-verifier is to residual risk in the draft response. A lower t r can be viewed as a conservative output filter: it intervenes whenever the verifier finds even weak evidence of toxicity in the draft. Because the echo template wraps the prompt in polite context, the response score distribution is shifted downward relative to the prompt score distribution. This explains why response thresholds that are numerically lower than prompt thresholds (e.g., t_r=0.15) still correspond to meaningful interventions. The ablation also shows diminishing returns: once benign redaction exceeds roughly 0.40, further lowering t_r primarily affects benign prompts, because many toxic prompts already trigger redaction at moderate thresholds.

Agent-level trade-off on jailbreak prompts. Figure 6 reports the same trade-off on jailbreak-classification. PromptGuard already achieves strong discrimination and incurs no false refusals on benign prompts at the thresholds we swept. Under an unsafe-echo target of 0.02 on jailbreak prompts, PromptGuard at t p=0.40 produced a hard-refusal rate of 0.523. VerifySafe at t p=0.90, t r=0.35 reduced the hard-refusal rate to 0.473 (−9.5%) by converting a subset of refusals into redactions. The unsafe-echo rate remained 0.014 on both methods because two jailbreak-labeled prompts in the test set lack explicit injection markers and receive low jailbreak scores; these cases indicate that response-level verification cannot compensate for detector blind spots when the draft response does not contain recognizable attack signatures.

Table 9. Agent-level safety and utility on jailbreak-classification test (selected operating points).

Agent	Hard refusal rate	Redaction rate	Unsafe echo rate (jailbreak)	False refusal (benign)	Benign redaction	Avg sim (benign)	Avg words (benign)
NoGuard (always echo)	0.000	0.000	1.000	0.000	0.000	0.771	111.057
PromptGuard (t=0.40)	0.523	0.000	0.014	0.000	0.000	0.771	111.057
VerifySafe (t_p=0.90, t_r=0.35)	0.473	0.050	0.014	0.000	0.000	0.771	111.057

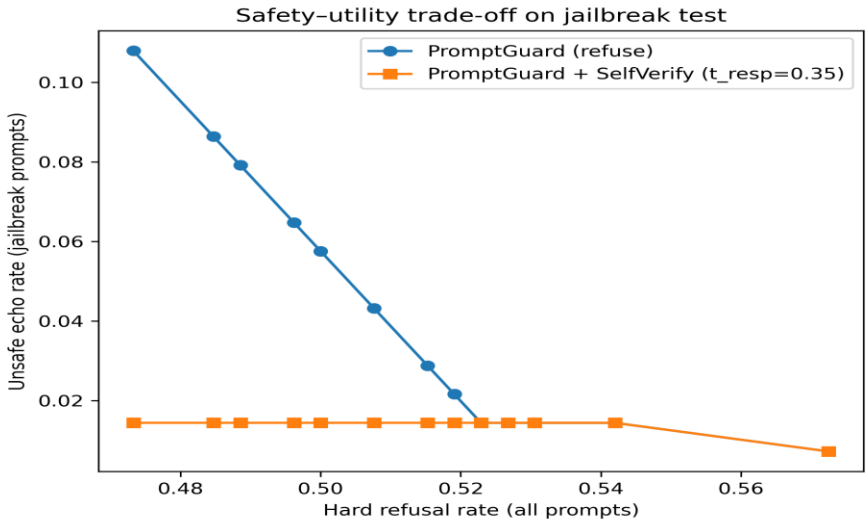

Figure 6. Hard refusals vs unsafe echoes on jailbreak test as the prompt threshold varies

.

Discussion. Across both datasets, the main effect of self-verification is to decouple safety from hard refusals. Prompt-only gating forces systems to choose between over-refusal and unsafe leakage. VerifySafe introduces a third action—evidence-backed redaction—that removes verbatim unsafe content while preserving a conversational reply. This is especially valuable on datasets like ToxiGen where benign false positives are non-trivial and group-conditioned. The evidence summary is also operationally useful: it provides a lightweight, auditable explanation for why the system refused or redacted, without revealing disallowed text. In deployments with an actual generative model, the same interface can be used to request the LLM to propose a revised draft conditioned on the verifier's evidence summary, creating a closed-loop safety layer that is compatible with alignment training and red-teaming practices [13]–[15].

Toward a unified safety score. Our experiments treat toxicity and jailbreak detection as separate tasks because they are provided in separate datasets. In a deployed agent, a unified policy would combine multiple risk signals: toxicity, self-harm, sexual content, illegal instructions, and jailbreak intent. VerifySafe naturally

supports such composition because self-verification operates on the draft response: if any verifier flags the draft, the system can redact or refuse. This is particularly relevant for jailbreak prompts that do not contain explicit injection markers but do contain requests for harmful content; a toxicity or abuse detector can catch these cases even if a jailbreak detector does not. Our pipeline therefore encourages defense-in-depth: prompt gating reduces obvious attacks, and response-level verification provides a last check on what will be released.

Limitations

First, our end-to-end agent evaluation uses a deterministic echo-based response generator to make unsafe output events directly measurable and fully reproducible. This design isolates the safety stack (prompt detector + self-verifier) from confounding variation in model generation. In an LLM-based agent, the generator can introduce new toxicity not present in the prompt, and it can also partially comply with jailbreak instructions without verbatim echoing them. Our unsafe-echo metric therefore captures a strict and interpretable failure mode (verbatim reproduction) rather than the full space of unsafe behaviors.

Second, our self-verification module uses the same family of detectors at both the prompt and response stages. While this is realistic for production guardrails, it can lead to correlated errors: if a toxic statement is subtle enough to fool the detector, it can also fool the response verifier. This effect is visible in jailbreak-classification, where two jailbreak-labeled prompts are not captured by either the prompt score or the response score. Future work should combine heterogeneous verifiers (e.g., distinct architectures or external toxicity APIs) and incorporate uncertainty-aware escalation.

Third, ToxiGen is balanced and machine-generated. The balanced label distribution (approximately 50/50) differs from real user traffic, where the base rate of toxic prompts is typically much lower. Thresholds optimized for balanced datasets therefore over-refuse in deployment without recalibration. In addition, machine generation creates artifacts that are easier for linear models to detect than human-written abuse, and the group prompts cover a specific set of identities. Evaluating on additional corpora and languages is necessary to establish robustness beyond this benchmark.

Fourth, our evidence summaries rely on linear feature attributions, which are faithful to the detector but not necessarily aligned with human interpretations of harm. We mask evidence strings to avoid reproducing abusive language, which can reduce interpretability. More semantically grounded evidence—e.g., structured harm categories or explanations generated by a separate model—improves transparency while maintaining safety.

Finally, our unsafe-echo safety metric is intentionally narrow and conservative: it counts verbatim reproduction of toxic or jailbreak prompts as unsafe. A real system must also prevent paraphrased abuse, subtle endorsement, and indirect harms such as enabling discrimination through seemingly neutral advice. Moreover, some benign use cases involve quoting harmful content for reporting or support-seeking; in such contexts, redaction removes information that is needed to provide help. Evaluating these nuanced scenarios requires richer taxonomies of harm and controlled human studies that measure both perceived safety and task success.

Conclusion

We presented VerifySafe, a toxicity- and jailbreak-aware agent pipeline that connects prompt classification with response self-verification and evidence-style summaries. Using full empirical evaluations on ToxiGen and jailbreak-classification, we quantified the trade-off between safety (avoiding unsafe echoes) and utility (avoiding hard refusals). On ToxiGen, VerifySafe reduced hard refusals from 0.523 to 0.115 under an unsafe-echo target of 0.20 by converting refusals into redactions and attaching masked evidence summaries. On jailbreak-classification, VerifySafe reduced hard refusals from 0.523 to 0.473 under an unsafe-echo target of 0.02. These results demonstrate that response-level verification is an effective, model-agnostic lever for improving the safety–utility balance of dialogue agents, and it provides an interface for transparent, auditable safety decisions.

References

[1] P. Hartvigsen et al., "ToxiGen: A Large-Scale Machine-Generated Dataset for Implicit and Adversarial Hate Speech Detection," in Proc. ACL, 2022.

[2] S. Gehman et al., "RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models," Findings of ACL, 2020.

[3] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated Hate Speech Detection and the Problem of Offensive Language," in Proc. ICWSM, 2017.

[4] Z. Waseem and D. Hovy, "Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter," in Proc. NAACL SRW, 2016.

[5] A. Founta et al., "Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior," in Proc. ICWSM, 2018.

[6] L. Dixon et al., "Measuring and Mitigating Unintended Bias in Text Classification," in Proc. AIES, 2018.

[7] A. Borkan, L. Dixon, J. Sorensen, N. Thain, and L. Vasserman, "Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification," in Proc. WWW, 2019.

[8] M. Zampieri et al., "Predicting the Type and Target of Offensive Posts in Social Media," in Proc. NAACL, 2019.

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

[10] Y. Liu et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," arXiv preprint arXiv:1907.11692, 2019.

[11] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You? Explaining the Predictions of Any Classifier," in Proc. KDD, 2016.

[12] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in Proc. NeurIPS, 2017.

[13] L. Ouyang et al., "Training Language Models to Follow Instructions with Human Feedback," arXiv preprint arXiv:2203.02155, 2022.

[14] Y. Bai et al., "Constitutional AI: Harmlessness from AI Feedback," arXiv preprint arXiv:2212.08073, 2022.

[15] E. Perez et al., "Red Teaming Language Models with Language Models," arXiv preprint arXiv:2202.03286, 2022.

[16] J. Wei et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," arXiv preprint arXiv:2201.11903, 2022.

[17] X. Wang et al., "Self-Consistency Improves Chain of Thought Reasoning in Language Models," arXiv preprint arXiv:2203.11171, 2022.

[18] W. Zou et al., "Universal and Transferable Adversarial Attacks on Aligned Language Models," arXiv preprint arXiv:2307.15043, 2023.

[19] A. Madaan et al., "Self-Refine: Iterative Refinement with Self-Feedback," arXiv preprint arXiv:2303.17651, 2023.

[20] N. Shinn et al., "Reflexion: Language Agents with Verbal Reinforcement Learning," arXiv preprint arXiv:2303.11366, 2023.

[21] W. Manakul et al., "SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models," arXiv preprint arXiv:2303.08896, 2023.

[22] J. Hao, "jailbreak-classification (balanced splits)," dataset repository, 2023.

[23] K. Xu, H. Zhou, H. Zheng, M. Zhu, and Q. Xin, “Intelligent classification and personalized recommendation of e-commerce products based on machine learning,” Proceedings of the 6th International Conference on Computing and Data Science (ICCDS), 2024.

[24] Xinzhuo Sun, Jing Chen, Binghua Zhou, and Meng-Ju Kuo, “ConRAG: Contradiction-Aware Retrieval-Augmented Generation under Multi-Source Conflicting Evidence”, JACS, vol. 4, no. 7, pp. 50–64, Jul. 2024, doi: 10.69987/JACS.2024.40705.

[25] Xinzhuo Sun, Yifei Lu, and Jing Chen, “Controllable Long-Term User Memory for Multi-Session Dialogue: Confidence-Gated Writing, Time-Aware Retrieval-Augmented Generation, and Update/Forgetting”, JACS, vol. 3, no. 8, pp. 9–24, Aug. 2023, doi: 10.69987/JACS.2023.30802.

[26] Hanqi Zhang, “DriftGuard: Multi-Signal Drift Early Warning and Safe Re-Training/Rollback for CTR/CVR Models”, JACS, vol. 3, no. 7, pp. 24–40, Jul. 2023, doi: 10.69987/JACS.2023.30703.

[27] Hanqi Zhang, “Risk-Aware Budget-Constrained Auto-Bidding under First-Price RTB: A Distributional Constrained Deep Reinforcement Learning Framework”, JACS, vol. 4, no. 6, pp. 30–47, Jun. 2024, doi: 10.69987/JACS.2024.40603.

[28] Q. Xin, Z. Xu, L. Guo, F. Zhao, and B. Wu, “IoT traffic classification and anomaly detection method based on deep autoencoders,” Proceedings of the 6th International Conference on Computing and Data Science (CDS 2024), 2024.

[29] B. Wang, Y. He, Z. Shui, Q. Xin, and H. Lei, “Predictive optimization of DDoS attack mitigation in distributed systems using machine learning,” Proceedings of the 6th International Conference on Computing and Data Science (CDS 2024), 2024, pp. 89–94.

[30] T. Shirakawa, Y. Li, Y. Wu, S. Qiu, Y. Li, M. Zhao, H. Iso, and M. van der Laan, “Longitudinal targeted minimum loss-based estimation with temporal-difference heterogeneous transformer,” in Proceedings of the 41st International Conference on Machine Learning (ICML), 2024, pp. 45097–45113, Art. no. 1836.

[31] Meng-Ju Kuo, Boning Zhang, and Haozhe Wang, "Tokenized Flow-Statistics Encrypted Traffic Analysis: Comparative Evaluation of 1D-CNN, BiLSTM, and Transformer on ISCX VPN-nonVPN 2016 (A1+A2, 60 s)", JACS, vol. 3, no. 8, pp. 39–53, Aug. 2023, doi: 10.69987/JACS.2023.30804.

[32] Z. Zhong, M. Zheng, H. Mai, J. Zhao, and X. Liu, "Cancer image classification based on DenseNet model," Journal of Physics: Conference Series, vol. 1651, no. 1, p. 012143, 2020.

[33] Z. S. Zhong and S. Ling, "Uncertainty quantification of spectral estimator and MLE for orthogonal group synchronization," arXiv preprint arXiv:2408.05944, 2024.

[34] Z. Ling, Q. Xin, Y. Lin, G. Su, and Z. Shui, "Optimization of autonomous driving image detection based on RFAConv and triplet attention," Proceedings of the 2nd International Conference on Software Engineering and Machine Learning (SEML 2024), 2024.

[35] J. Chen, J. Xiong, Y. Wang, Q. Xin, and H. Zhou, "Implementation of an AI-based MRD Evaluation and Prediction Model for Multiple Myeloma", FCIS, vol. 6, no. 3, pp. 127–131, Jan. 2024, doi: 10.54097/zJ4MnbWW.

[36] Z. S. Zhong and S. Ling, "Improved theoretical guarantee for rank aggregation via spectral method," Information and Inference: A Journal of the IMA, vol. 13, no. 3, 2024.